Natural for Windows

Natural for Ajax

バージョン 6.3.3

October 2008

Natural

This document applies to Natural バージョン 6.3.3 and to all subsequent releases.

Specifications contained herein are subject to change and these changes will be reported in subsequent release notes or new editions.

目次

1 Natural for Ajax

This documentation is organized under the following headings:

Using Natural for Ajax		
●	**Introduction**	What is Natural for Ajax?
●	**Installation**	How to install Natural for Ajax on the supported application servers.
●	**Setting Up Your Environment**	How to set up Application Designer, your development environment for Natural, and your runtime environment for Natural.
●	**First Steps**	How to create a 「Hello World!」 application.
●	**Developing the User Interface**	How to develop the user interface using Application Designer.
●	**Developing the Application Code**	How to develop the application code using Natural Studio or Natural for Eclipse.
●	**Deploying the Application**	How to unload and install the Natural modules and user interface components.
●	**Multi Language Management**	Describes aspects to be considered for internationalization.
Application Designer Reference (adapted to Natural for Ajax)		
●	**Working with Controls**	Shows you how to work with the elements that are placed into containers - the controls.
●	**Working with Grids**	Explains what grids are and how to use them.
●	**Working with Trees**	Explains the basic types of trees and how to use them.
●	**Working with Menus**	Shows you how to arrange a number of functions in a structured way.

●	**Non-Visual Controls and Hot Keys**	Describes how to develop controls that do not have visual effects.

2 Introduction

Using Natural for Ajax, you can create rich internet applications which use the Ajax (Asynchronous JavaScript and XML) technology. This enables Natural users on Windows, UNIX and mainframe platforms to develop and use Natural applications with a browser-based user interface, similar to GUI desktop applications.

This chapter covers the following topics:

What is a Rich Internet Application?

Classical HTML- and browser-based applications suffer from known disadvantages. The server responds to each user interaction with a new page. This may lead to long response times and new rendering in the browser and thus to a discontinuous workflow for the user. The possibilities offered by DHTML overcome these disadvantages, but they are complicated to use and make it hard to build a comfortable user interface. The user interface is therefore often simpler and less comfortable than users are accustomed to from their experience with desktop applications. Although it is possible to provide complex controls and features like drag-and-drop, this is hard to implement - especially if compatibility with all commonly used browsers is required. Classical GUI applications also have the disadvantage that a client component of the application must be installed on each client machine.

Rich internet applications that use the Ajax technology overcome these disadvantages by combining the reachability of browser-based applications with the rich user interface of GUI applications. Software AG provides support for the development of rich internet applications with Application Designer. Natural for Ajax combines the user interface capabilities of Application Designer with the application development capabilities of Natural.

Rich Internet Applications with Natural

At runtime, a rich internet application with Natural has the following structure:

- A Natural host session on a Windows, UNIX or mainframe server runs the application code. Other than with a map application, the application does not deal with user interface issues. It contains only the application logic and communicates with the user interface layer by sending and receiving data. The data is displayed in page in a web browser. Events - such as button clicks - that the user raises in the web browser are passed back to the application code. Along with an event, the application code receives also the data that the user modified in the web browser. It processes the event and the data and returns modified data back to the web browser page.

- Natural for Ajax, which is running on an application server, merges the data received from the Natural application into a DHTML page and delivers the page to the web browser. In the inverse direction, Natural for Ajax forwards events that the user raised in the web browser along with the modified data to the Natural application.
- A web browser renders the DHTML page. JavaScript code on the page processes local user interaction and exchanges data with Natural for Ajax as needed. It uses Ajax technology to exchange data with the Natural application in the background without having to re-render the page as a whole.

At development time, a rich internet application is created with Natural in the following way:

- Application Designer is used to develop the user interface layout of a web page and to bind the controls on the page to data elements in the application. Application Designer is contained in the Natural for Ajax module running on the application server.
- When the user saves the page layout, a Natural module of type 「Adapter」 is generated. The adapter serves as an interface between the application code and the page layout. It contains:
  - A data structure that describes the data that the Natural application has to deliver to the application server in order to populate the web page.
  - The Natural code necessary to transfer the data structure to the user interface and to receive modified data back.
  - A code skeleton, in the form of comment lines, that contains handlers for the expected events. The application programmer can copy this code skeleton into the main program to implement the event handlers.
- Then a main program is implemented that exchanges data with the web page using the adapter and handles the events. The event handler code has no knowledge of the web page layout, but operates only on the page data that is sent and received through the adapter.
- The navigation between different pages is implemented. A rich internet application navigates between pages in the same way as a map application would navigate between maps.

Mixed Applications

With the support of Unicode, Natural has introduced the Web I/O Interface which renders Natural maps in a web browser. Typically, if you are running map-oriented applications and wish to change them to rich internet applications, you will do this gradually. In certain parts of an application, maps might be replaced by rich GUI pages, other parts will possibly be left unchanged. Therefore, Natural supports running mixed applications which consist of both maps and rich GUI pages. With maps, the application controls the page layout, and the rendering mechanism therefore respects the layout information that the application provides. With rich GUI pages, the application does not control the layout; the layout is controlled by Application Designer. However, for the users of an application the switch between maps and rich GUI pages is seamless.

3 Installation

Natural for Ajax consists of a J2EE enterprise application (*njx111.ear*) and a J2EE resource adapter (*njx111ra.rar*). Both components are to be deployed on a J2EE server. Natural for Ajax receives data from Natural applications running on a Windows, UNIX or mainframe host and delivers web pages to the user's web browser.

This chapter describes the installation of Natural for Ajax on application servers on Windows or UNIX. It does not describe the installation of the additionally required Natural components on a Windows, UNIX or mainframe host, but refers to the corresponding installation documents.

This chapter covers the following topics:

Prerequisites

The following topics are covered below:

- J2EE Server
- Apache Ant
- Natural for Windows
- Natural for UNIX
- Natural for Mainframes
- Development Servers
- Development Clients

J2EE Server

The following application servers are supported. The application servers are not delivered with Natural for Ajax. They can be obtained from the locations indicated below, according to their respective license terms.

- JBoss Application Server 4.0.5 (see ***http://www.jboss.org/***).
- Sun Java System Application Server 8.2 (see ***http://www.sun.com/***).

Apache Ant

Apache Ant 1.6.5 or above is required to perform the deployment on JBoss Application Server. This tool is freely available on ***http://ant.apache.org/***.

Natural for Windows

If you want to use Natural for Ajax with Natural for Windows, the following must be installed:

- Natural for Windows Version 6.3.3 or above, and
- the Web I/O Interface service.

For detailed information, see the following documentation which is provided for Natural for Windows:

- *Setting Up the Web I/O Interface Service* in the *Operations* documentation.

Natural for UNIX

If you want to use Natural for Ajax with Natural for UNIX, the following must be installed:

- Natural for UNIX Version 6.3.1 or above, and
- the Web I/O Interface daemon.

For detailed information, see the *Installation* documentation which is provided for Natural for UNIX, especially the following sections:

- *Installing and Setting Up Natural on UNIX*
- *Installing the Web I/O Interface Daemon on UNIX*
- *Activating the Web I/O Interface Daemon on UNIX*

Natural for Mainframes

If you want to use Natural for Ajax with Natural for Mainframes, the following must be installed:

- Natural for Mainframes Version 4.2.3 or above, and
- the Natural Web I/O Interface Server.

For detailed information, see the following documentation which is provided for Natural for Mainframes:

- *Installation* (describes how to install Natural for Mainframes)
- *Natural Web I/O Interface Server* (describes how to install the Natural Web I/O Interface server)

Development Servers

The following development servers support the remote development of Natural for Ajax applications:

- Natural Development Server for Windows Version 2.2.4 or above.
- Natural Development Server for UNIX Version 2.2.3 or above.
- Natural Development Server for Mainframes Version 2.2.3 or above.

Development Clients

The following development clients support the remote development of Natural for Ajax applications:

- Natural for Windows (Natural Studio) Version 6.3.1 or above.
- Natural for Eclipse Version 3.1.2 or above.

License Key File Handling

A valid license key file is required during the installation. The license key file is an XML file which is usually supplied along with the product. Alternatively, you can obtain a license key file from Software AG via your local distributor.

Installing Natural for Ajax on JBoss Application Server

It is assumed that `<jboss>` is the directory of your JBoss Application Server installation.

The following topics are covered below:

- Update Installation

First-time Installation

手順 3.1. To install Natural for Ajax

1 Edit the file *<jboss>/server/default/deploy/jbossjca-service.xml* and change the setting

```
<!-- Enable connection close debug monitoring -->
<attribute name="Debug">true</attribute>
```

to

```
<!-- Enable connection close debug monitoring -->
<attribute name="Debug">false</attribute>
```

2 Install Apache Ant (you need Apache Ant to deploy Natural for Ajax to the JBoss Application Server; see the ***Prerequisities*** above for the required version number):

1. Download and unzip Apache Ant (from ***http://ant.apache.org/***) into an installation directory of your choice. Avoid a directory name that contains blanks.
2. Let the environment variable `ANT_HOME` point to the directory *<ant>* (where *<ant>* is the directory of your Ant installation).
3. Add *<ant>/bin* to your `PATH` environment variable.

3 Deploy Natural for Ajax to JBoss Application Server:

1. Copy the Natural for Ajax distributables to a directory on a disk drive.
2. In the directory that contains the Natural for Ajax distributables, there is an Ant script named *jbossdeploy.xml*. Edit this script and change the setting

   ```
   <property name="jbosshome" value=""/>
   ```

to

```
<property name="jbosshome" value="<jboss>"/>
```

where *<jboss>* is your JBoss Application Server installation directory.

⚠ **Important:** Take care to use forward slashes (also on Windows) when specifying the directory path.

3. Execute the script *jbossdeploy.xml* by entering the following command:

```
ant -f jbossdeploy.xml
```

Wait for the message 「BUILD SUCCESSFUL」. This indicates that the deployment was successful.

4 Copy the license file into the directory *<jboss>/server/default/deploy/njx111.ear/cisnatural.war/cis/licensekey*.

5 Start JBoss Application Server.

Update Installation

▶手順 3.2. To update Natural for Ajax

1 Shut down JBoss Application Server.

2 Deploy Natural for Ajax to JBoss Application Server:

1. Copy the Natural for Ajax distributables to a directory on a disk drive.
2. In the directory that contains the Natural for Ajax distributables, there is an Ant script named *jbossdeploy.xml*. Edit this script and change the setting

```
<property name="jbosshome" value=""/>
```

to

```
<property name="jbosshome" value="<jboss>"/>
```

where *<jboss>* is your JBoss Application Server installation directory.

⚠ **Important:** Take care to use forward slashes (also on Windows) when specifying the directory path.

3. Execute the script *jbossdeploy.xml* by entering the following command:

```
ant -f jbossdeploy.xml redeploy
```

Wait for the message 「BUILD SUCCESSFUL」. This indicates that the deployment was successful.

3 Regenerate the HTML pages of the projects that you have created with an earlier release of Natural for Ajax. For each project to regenerate, execute the script *jbossdeploy.xml* by entering the following command:

```
ant -f jbossdeploy.xml regenerate -Dnjxproj=<projectname>
```

Wait for the message 「BUILD SUCCESSFUL」. This indicates that the generation was successful.

4 Start JBoss Application Server.

Installing Natural for Ajax on Sun Java System Application Server

Natural for Ajax is installed using the Adminstration Console of Sun Java System Application Server.

The following is assumed:

- `<host>` is the name of the machine on which the application server is installed.
- `<port>` is the name of the port where the application server is installed. In a default installation, this is port 8080.
- `<adminport>` is the name of the port where the Adminstration Console is installed. In a default installation, this is port 4848.
- `<sunas>` is the path to the directory in which the application server is installed. In a default installation on Windows, this is *C:/Sun/AppServer*.

The following topics are covered below:

- Update Installation

First-time Installation

手順 3.3. To install Natural for Ajax

1 Start the application server.

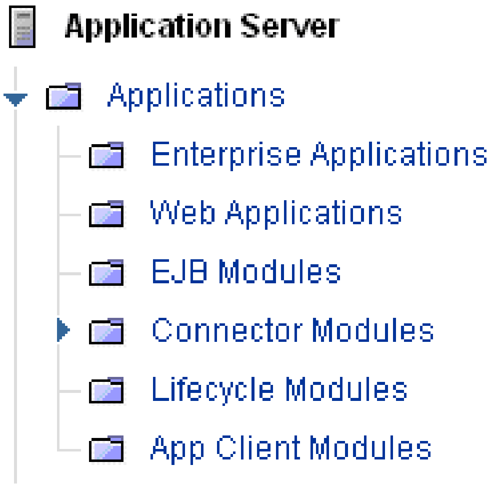
2 Open your web browser and enter the following URL:

```
http://<host>:<adminport>
```

This opens the Adminstration Console.

3 Deploy the resource adapter *njx111ra.rar*:

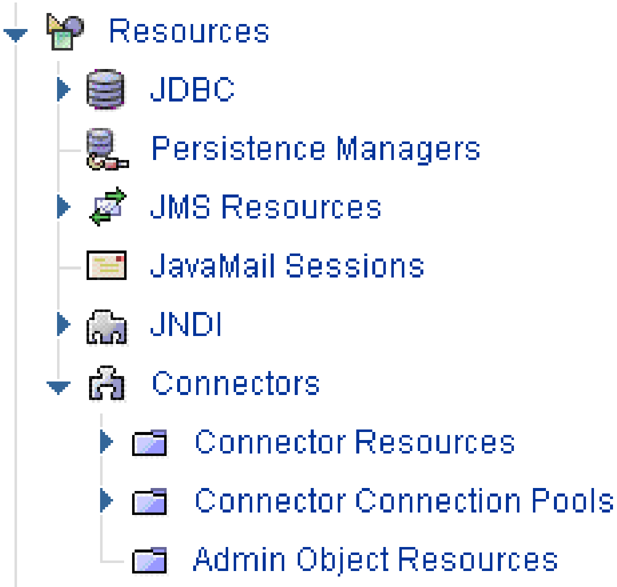
1. Open the tree node **Applications > Connector Modules**.



2. Choose **Deploy**.
3. Select *njx111ra.rar* as the package file to be uploaded to the application server.
4. Choose **Next**. "njx111ra" is automatically included as the application name.
5. Choose **Finish**.

4 Define the JNDI name for the resource adapter:

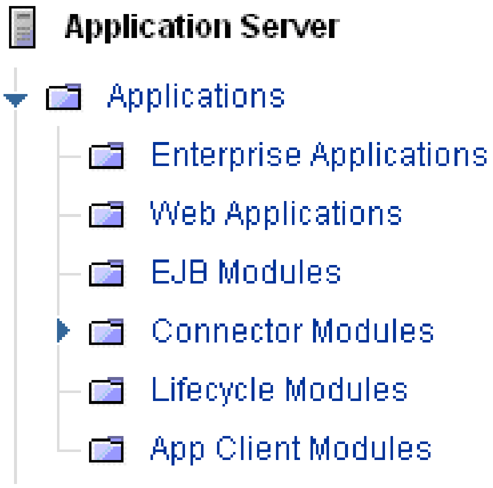
1. Open the tree node **Resources> Connectors >Connector Connection Pools**.



2. Choose **New**.
3. Enter "NatPool" (the name is arbitrary) as the name.
4. Select **njx111ra** as the resource adapter.
5. Choose **Next**.
6. Choose **Next**.
7. Choose **Finish**.
8. Open the tree node **Resources> Connectors >Connector Resources**.
9. Choose **New**.
10. Enter "eis/NaturalUnicodeRA" as the JNDI name.
11. Select **NatPool** (or whatever name you specified previously) as the pool name.
12. Choose **OK**.

5 Deploy the enterprise application *njx111.ear*:

1. Open the tree node **Applications > Enterprise Applications**.



2. Choose **Deploy**.
3. Select *njx111.ear* as the file to upload.
4. Choose **Next**.
5. Choose **OK**. The deployment may take several minutes.

6 Copy the license file into the directory <sunas>*/domains/domain1/applications/j2ee-apps/njx111/cisnatural_war/cis/licensekey*.

7 Edit the file <sunas>*/domains/domain1/config/server.policy* and add the followings settings:

```
// Allow Application Designer to create an own class loader
grant {
permission java.lang.RuntimePermission "createClassLoader";
};

// Allow Application Designer to modify its own project directories
grant {
permission java.io.FilePermission
"${com.sun.aas.instanceRoot}${/}applications${/}j2ee-apps${/}njx111${/}cisnatural_war${/}-",
"read,write,delete";
};
```

8 Restart the application server.

Update Installation

手順 3.4. To update Natural for Ajax

1 Shut down the application server.

2 Create a backup copy of your *sessions.xml* file, which is located in `<sunas>`*/domains/domain1/applications/j2ee-apps/njx111/cisnatural_war/WEB-INF*.

3 Create a backup copy of your license file, which is located in `<sunas>`*/domains/domain1/applications/j2ee-apps/njx111/cisnatural_war/cis/licensekey*.

4 Create backup copies of previously created projects, which are located in `<sunas>`*/domains/domain1/applications/j2ee-apps/njx111/cisnatural_war*.

5 Start the application server.

6 Start a web browser and enter the following URL:

```
http://<host>:<adminport>
```

This opens the Adminstration Console.

7 Undeploy the resource adapter *njx111ra.rar*.

8 Undeploy the enterprise application *njx111.ear*.

9 Deploy the new version of Natural for Ajax as in a first-time installation.

10 Shut down the application server.

11 Restore the files that you have backed up in steps 2, 3 and 4.

12 Start the application server.

13 Start a web browser and enter the following URL:

```
http://<host>:<port>/cisnatural
```

This opens the Application Designer development workplace.

14 In the **Development Tools** node of the navigation frame, choose **Layout Manager**.

15 For each application project that you have created with an earlier release of Natural for Ajax, select the layout definitions and from the **Operations on multiple Items** menu, choose **(Re)Generate HTML Pages**.

Verifying the Installation

It is assumed that *http://<host>:<port>* is the URL of your application server.

▶手順 3.5. To verify the installation

1 Enter the following URL in your web browser:

```
http://<host>:<port>/cisnatural
```

This opens Application Designer's development workplace.

2 Enter the following URL in your web browser:

```
http://<host>:<port>/cisnatural/servlet/StartCISPage?PAGEURL=/cisnatural/NatLogon.html
```

This opens the Natural logon page. The installation is now complete.

4 Setting Up Your Environment

Before you start developing and executing Natural for Ajax applications, you have to make specific definitions in your development environment.

This chapter covers the following topics:

Setting Up Application Designer

Currently, there is nothing to configure for Natural pages.

Setting Up Your Development Environment for Natural

If you are practising remote development with Natural's Single Point of Development (SPoD), a Natural Development Server must be installed and activated on the remote machine.

- **Windows**
  When your Natural Development Server is located on Windows, the **Web I/O Interface service** option, which can be set with the setup type **Custom**, must be selected when installing Natural. See the *Installation* documentation which is provided with Natural for Windows.
- **UNIX**
  When your Natural Development Server is located on UNIX, see *Activating the Natural Development Server on UNIX* in the *Installation* documentation which is provided with Natural for UNIX.
- **Mainframe**
  When your Natural Development Server is located on a mainframe, see the Natural Development Server documentation.

手順 4.1. To set up Natural Studio

1 Ask your administrator for the host name and the port number of the Natural Development Server.

2 Connect to the Natural Development Server. See *Accessing a Remote Development Environment* in the *Remote Development Using SPoD* documentation which is provided with Natural for Windows.

3 It is recommended that you create a new Natural library for each Application Designer project.

手順 4.2. To set up Natural for Eclipse

1 Ask your administrator for the host name and the port number of the Natural Development Server.

2 Create a new target in Natural for Eclipse, using this host name and port number. For further information, see the Natural for Eclipse documentation.

3 When creating a Natural project, assign this target in the project properties.

Setting Up Your Runtime Environment for Natural

The following must be installed on the remote machine where you are going to test and execute the Natural code:

- **Windows**
  When your Natural Development Server is located on Windows, the **Web I/O Interface service** option, which can be set with the setup type **Custom**, must be selected when installing Natural Runtime. See the *Installation* documentation which is provided with Natural for Windows.
- **UNIX**
  A Web I/O Interface daemon must be installed and activated. For detailed information, see the *Installation* documentation which is provided for Natural for UNIX, especially the following sections:
  - *Installing and Setting Up Natural on UNIX*
  - *Installing the Web I/O Interface Daemon on UNIX*
  - *Activating the Web I/O Interface Daemon on UNIX*
- **Mainframe**
  The Web I/O Interface Server must be installed and started. See *Natural Web I/O Interface Server* in the Natural for Mainframes documentation.

手順 4.3. To set up the runtime environment for Natural for Windows

1 Ask your administrator for the host name and the port number of the Web I/O Interface service and the name of the batch file that is used to start up Natural sessions. A sample batch file for starting up Natural (*nwo.bat*) is delivered with Natural for Windows; see also *Batch File for Starting Natural* in the *Operations* documentation.

2 Edit the configuration file `<installdir>`*/WEB-INF/sessions.xml* (the location of `<installdir>` depends on your application server environment) and add the following entry:

```
<session id="session-name" trace="false">
    <natural_server>host-name</natural_server>
    <natural_port>port-number</natural_port>
    <natural_program>batch-file-name</natural_program>
</session>
```

where:

- *`session-name`* is to be replaced with the entry that should be available for selection in the logon page, and
- *`host-name`*, *`port-number`* and *`batch-file-name`* are to be replaced with the values you received from your administrator.

3 In the configuration file, there is a preconfigured session that is intended to start the Natural for Ajax examples that are delivered with Natural in the library `SYSEXNJX`. It contains the following settings:

```
<session id="Natural for Ajax Examples" trace="false">
    <natural_server>host-name</natural_server>
    <natural_port>port-number</natural_port>
    <natural_program>batch-file-name</natural_program>
</session>
```

Enter the settings (host name, port number and the name of the Natural startup batch file) that match your environment. Then you will be able to execute the examples from the logon page.

4 Restart the application server.

手順 4.4. To set up the runtime environment for Natural for UNIX

1 Ask your administrator for the host name and the port number of the Web I/O Interface daemon and the name of the script that is used to start up Natural sessions. A sample shell script for starting up Natural (*nwo.sh*) is delivered with Natural for UNIX; see also *nwo.sh - Shell Script for Starting Natural* in the *Installation* documentation.

2 Edit the configuration file `<installdir>`*/WEB-INF/sessions.xml* (the location of `<installdir>` depends on your application server environment) and add the following entry:

```
<session id="session-name" trace="false">
    <natural_server>host-name</natural_server>
    <natural_port>port-number</natural_port>
    <natural_program>script-name</natural_program>
</session>
```

where:

- `session-name` is to be replaced with the entry that should be available for selection in the logon page, and
- `host-name`, `port-number` and `script-name` are to be replaced with the values you received from your administrator.

3 In the configuration file, there is a preconfigured session that is intended to start the Natural for Ajax examples that are delivered with Natural in the library `SYSEXNJX`. It contains the following settings:

```
<session id="Natural for Ajax Examples" trace="false">
    <natural_server>host-name</natural_server>
    <natural_port>port-number</natural_port>
    <natural_program>script-name</natural_program>
</session>
```

Enter the settings (host name, port number and the name of the Natural startup script) that match your environment. Then you will be able to execute the examples from the logon page.

4 Restart the application server.

手順 4.5. To set up the runtime environment for Natural for Mainframes

1 Ask your administrator for the host name and the port number of the Web I/O Interface Server.

2 Edit the configuration file `<installdir>`*/WEB-INF/sessions.xml* (the location of `<installdir>` depends on your application server environment) and add the following entry:

```
<session id="session-name" trace="false">
    <natural_server>host-name</natural_server>
    <natural_port>port-number</natural_port>
</session>
```

where:

- *session-name* is to be replaced with the entry that should be available for selection in the logon page, and
- *host-name* and *port-number* are to be replaced with the values you received from your administrator.

3 In the configuration file, there is a preconfigured session that is intended to start the Natural for Ajax examples that are delivered with Natural in the library SYSEXNJX. It contains the following settings:

```
<session id="Natural for Ajax Examples" trace="false">
    <natural_server>host-name</natural_server>
    <natural_port>port-number</natural_port>
    <natural_program>script-name</natural_program>
</session>
```

Enter the settings (host name and port number) that match your environment. Remove the following line:

```
<natural_program>script-name</natural_program>
```

Then you will be able to execute the examples from the logon page.

4 Restart the application server.

5 First Steps

This documentation is organized under the following headings:

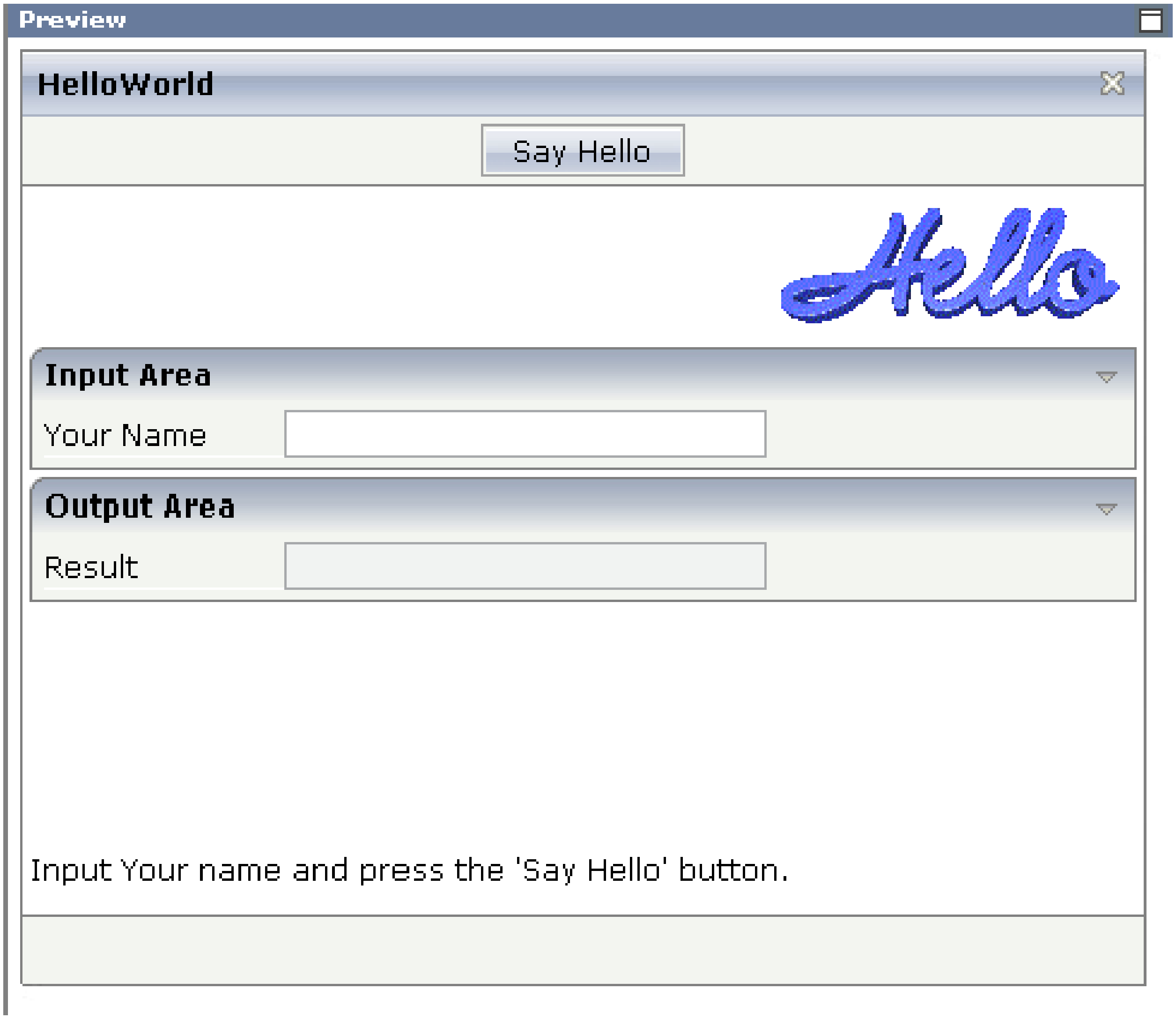
- About this Tutorial
- Starting the Development Workplace
- Creating a Project
- Getting Started with the Layout Painter
- Writing the GUI Layout
- Setting Up Your Environment
- Creating the Natural Code
- Some Background Information

It is important that you work through the exercises in the same sequence as they appear in this tutorial. Problems may occur if you skip an exercise.

6 About this Tutorial

This tutorial provides an introduction to working with Natural for Ajax. It explains how to create a 「Hello World!」 application. This covers all basic steps you have to perform when creating pages with Natural for Ajax: you create a layout file, you create an adapter and a main program, and you run the application.

When you have completed all steps of this tutorial, the page for your 「Hello World!」 application will look as follows:



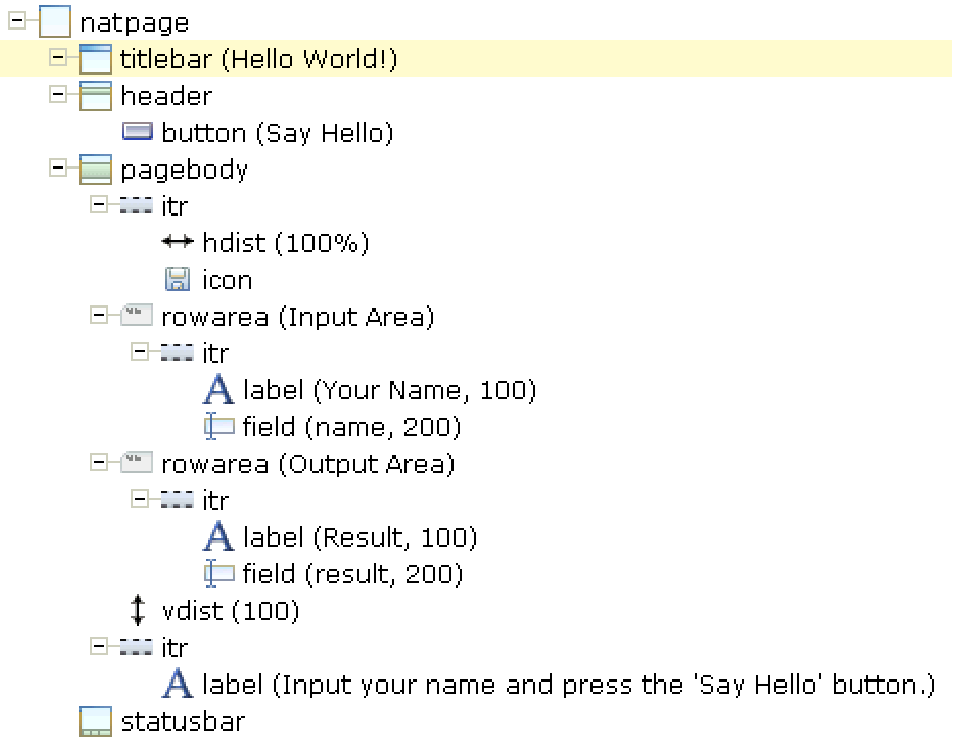
Input Your name and press the 'Say Hello' button.

Your application will act in the following way: When you enter a name in the **Your Name** field and choose the **Say Hello** button, the **Result** field displays "Hello World" and the name you have entered.

To reach this goal, you will proceed as follows:

1. You will first create a new Application Designer project.

2. You will then use Application Designer's Layout Painter to create the following layout:



```
natpage
    titlebar (Hello World!)
    header
        button (Say Hello)
    pagebody
        itr
            hdist (100%)
            icon
        rowarea (Input Area)
            itr
                label (Your Name, 100)
                field (name, 200)
        rowarea (Output Area)
            itr
                label (Result, 100)
                field (result, 200)
        vdist (100)
        itr
            label (Input your name and press the 'Say Hello' button.)
    statusbar
```

This corresponds to the following XML layout:

```
<?xml version="1.0" encoding="UTF-8"?>
<natpage natsource="HELLO-A">
    <titlebar name="Hello World!">
    </titlebar>
    <header withdistance="false">
        <button name="Say Hello" method="sayHello">
        </button>
    </header>
    <pagebody>
        <itr takefullwidth="true">
            <hdist width="100%">
            </hdist>
            <icon image="../cisdemos/images/hello.gif">
            </icon>
        </itr>
        <rowarea name="Input Area">
            <itr>
                <label name="Your name" width="100">
                </label>
                <field valueprop="name" width="200">
                </field>
            </itr>
        </rowarea>
        <rowarea name="Output Area">
            <itr>
                <label name="Result" width="100">
                </label>
                <field valueprop="result" width="200" displayonly="true">
                </field>
            </itr>
        </rowarea>
        <vdist pixelheight="100">
        </vdist>
        <itr>
            <label name="Input your name and press the &apos;Say Hello&apos;
button." asplaintext="true">
            </label>
        </itr>
    </pagebody>
    <statusbar withdistance="false">
    </statusbar>
</natpage>
```

3. When you save your layout for the first time, an intelligent HTML page and the Natural adapter for this page are generated.
4. Before you can start coding, you have to make specific definitions in your development environment (this tutorial assumes that you are using Natural Studio as your development environment).
5. You will import the generated Natural adapter into your Natural library
6. You will then create the main program which will use the adapter to display the page and which will handle the events that occur on the page, for example, when you choose the **Say Hello** button of your application.

You can now proceed with your first exercise: ***Starting the Development Workplace***.

7 Starting the Development Workplace

This tutorial assumes that you have installed Natural for Ajax as described in the *Installation* section.

▶手順 7.1. To start the development workplace

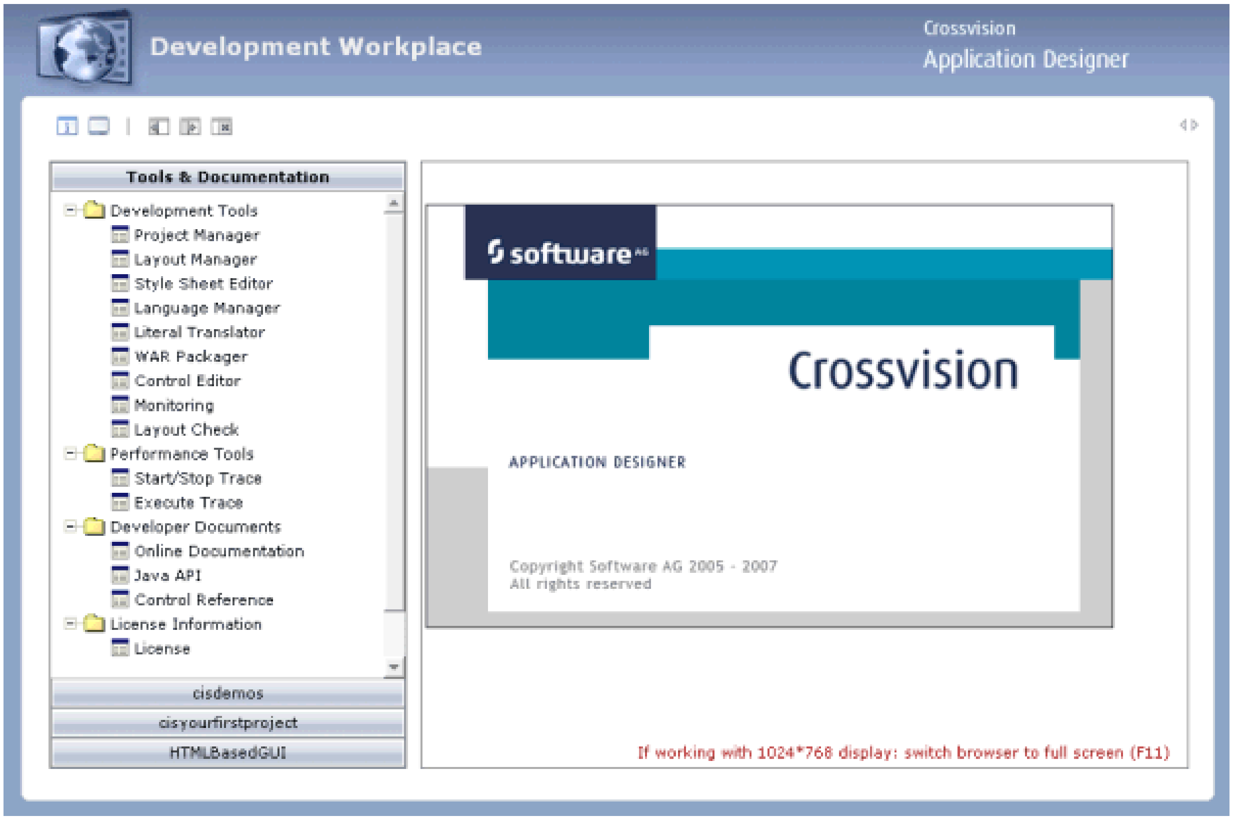
1. Make sure that your application server is running.
2. Invoke your browser and start the development workplace with the following URL:

   ```
   http://<host>:<port>/cisnatural
   ```

   where `<host>` is the name of the machine on which your application server is installed and `<port>` is the port number of your application server.

   Note: If you have not defined another port number during installation, the default port number is "8080".

The development workplace is now shown in your browser.



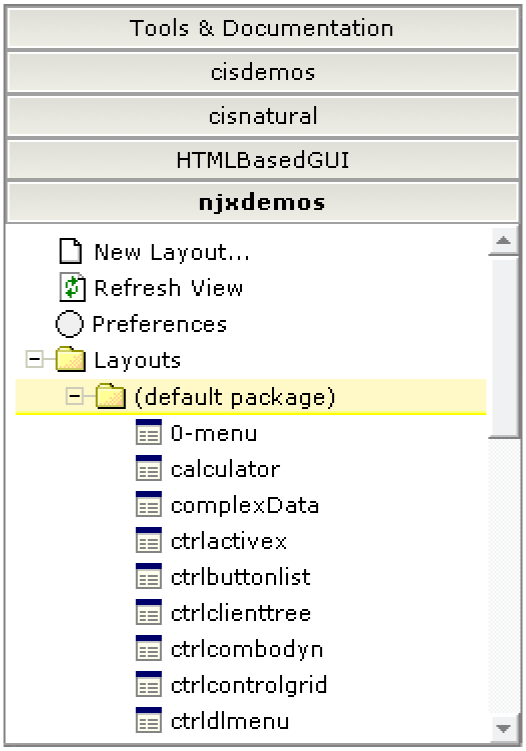
Copyright Software AG 2005 - 2007
All rights reserved

If working with 1024*768 display: switch browser to full screen (F11)

You can now proceed with the next exercise: *Creating a Project*.

8 Creating a Project

In the Application Designer environment, layouts are structured in so-called application projects. In the development workplace, you see the existing projects on the left. For each project, there is a tree of layout definitions that you can display when you choose the button containing the project name. For example:



For this tutorial, you will now create a project with the name "cisnatfirst".

手順 8.1. To create a project

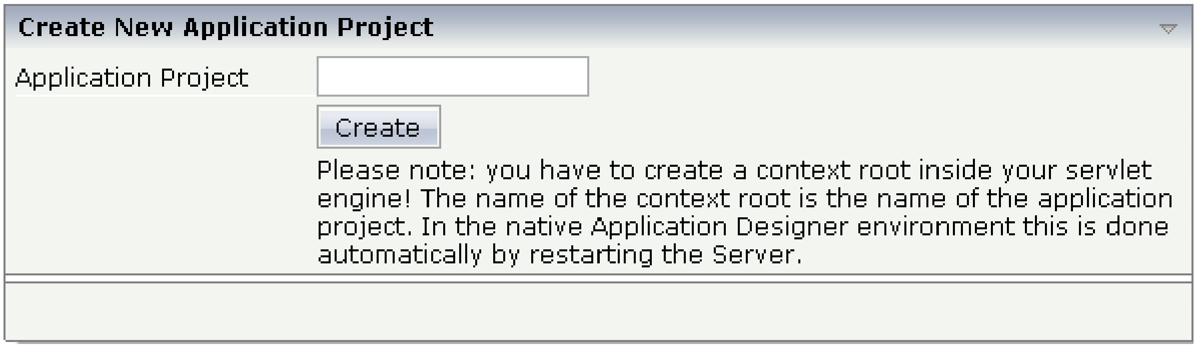
1 Choose **Tools & Documentation** to display the list of development tools.

2 Choose **Project Manager** in the tree.

A list of existing application projects is now shown on the right.

3 Choose the **New** button which is located below the list of application projects.

The following is now shown:



Please note: you have to create a context root inside your servlet engine! The name of the context root is the name of the application project. In the native Application Designer environment this is done automatically by restarting the Server.

4 Enter "cisnatfirst" as the name of your project and choose the **Create** button.

Your new project is now shown in the list of existing application projects on the right.

5 Use your browser's refresh function (F5 in Internet Explorer) to reload the development workspace.

The left side, which shows buttons for all existing projects, now also shows a button for your new project.

You can now proceed with the next exercise: *Getting Started with the Layout Painter*.

9 Getting Started with the Layout Painter

The Layout Painter, which can be accessed from the development workplace, is used to write the page layout. This is an Application Designer application itself.

This chapter contains the following exercises:

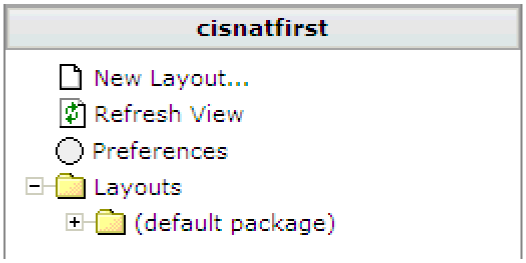
Creating a New Layout

You will now create a layout which is stored in the project you have previously created.

▶手順 9.1. To choose a layout template

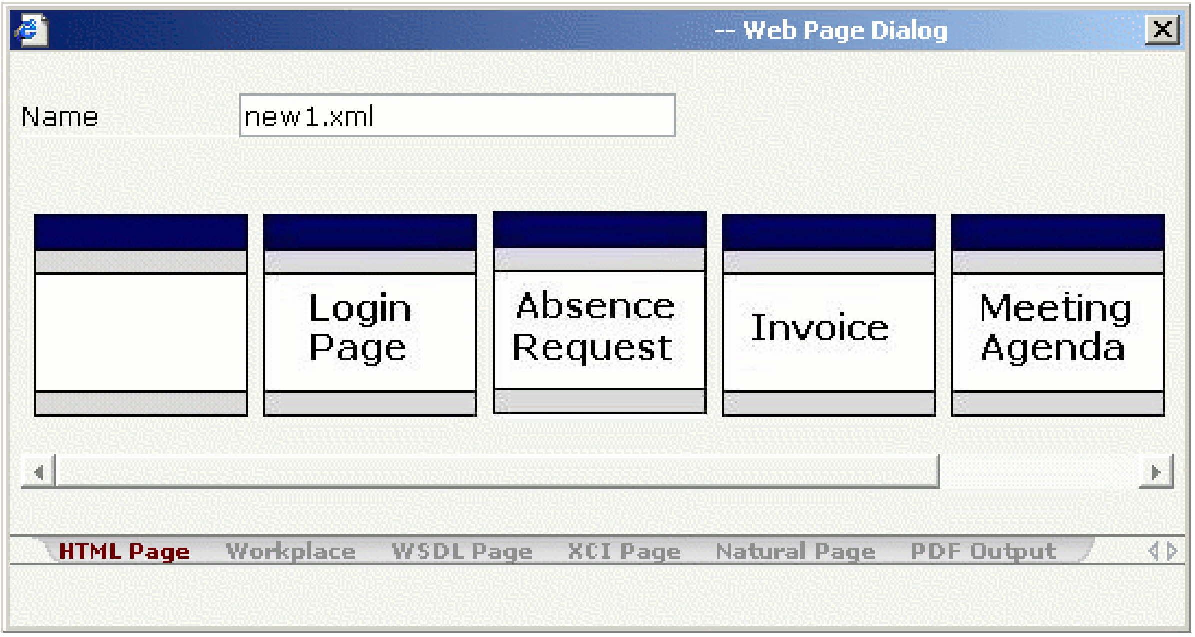
1 Choose the button for the project **cisnatfirst**.

The list of layout nodes inside the tree will be empty at the beginning:



2 Choose **New Layout...** in the tree.

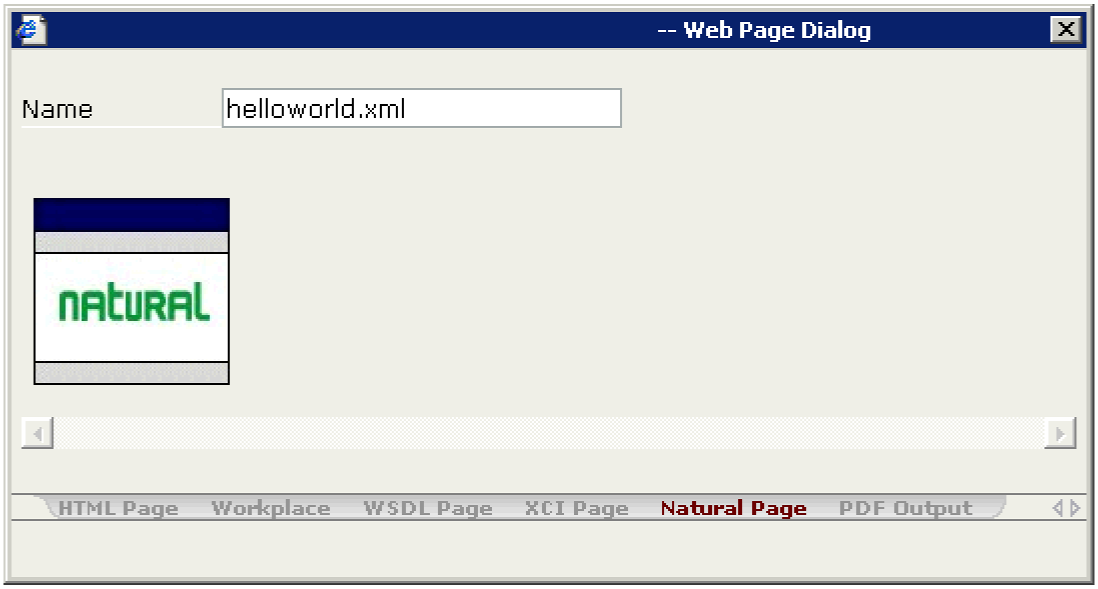
The following dialog appears.



3 Enter "helloworld.xml" in the **Name** field.

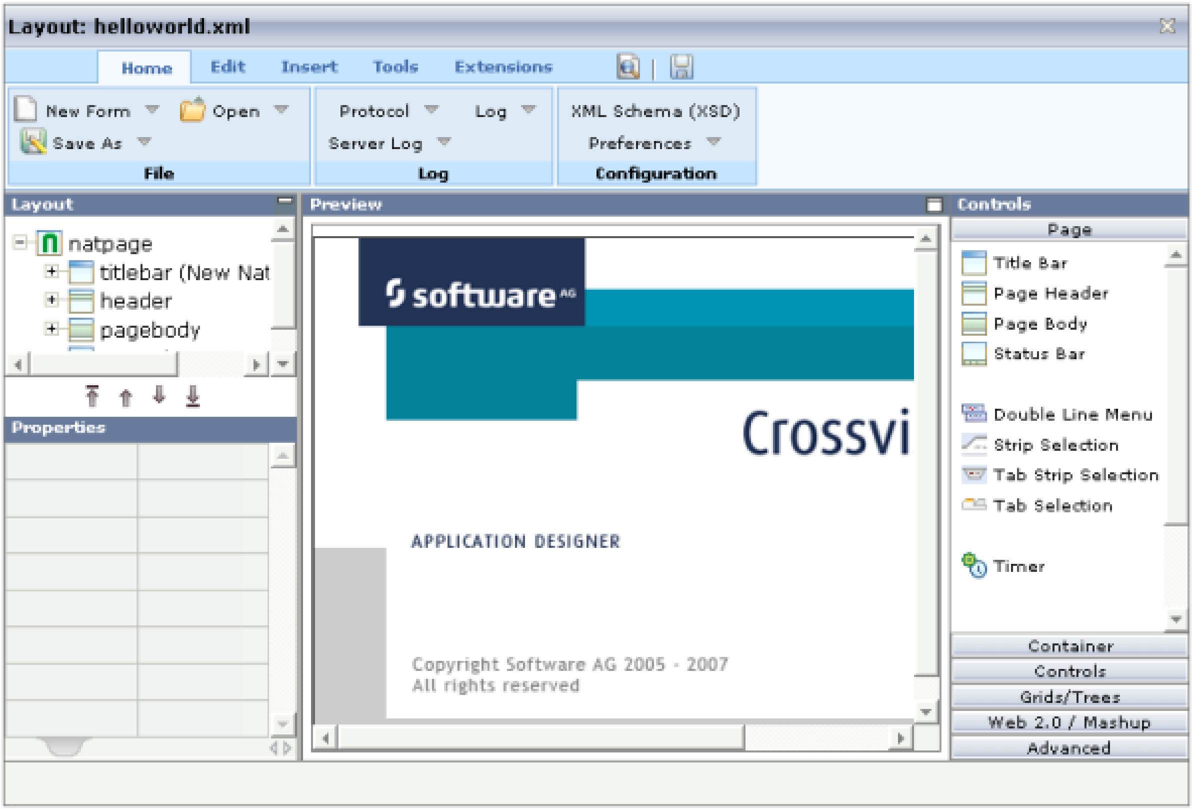
This is the name of your layout definition.

4 Select the **Natural Page** tab at the bottom of the dialog.



5 Select the template for the Natural page (when you move the mouse over this template, the tool tip "Natural Page" appears).

The main screen of the Layout Painter appears:



Note: The file *helloworld.xml* is stored in the */xml* directory of your project.

Elements of the Layout Painter Screen

The Layout Painter screen is divided into several areas:

- **Layout Area (left side)**
  This area consists of a layout tree and a properties area.

  The layout tree contains the controls that represent the XML layout definition. You drag these controls from the controls palette into the layout tree. Each node in the layout tree represents an XML tag.

  In the properties area below the layout tree, you specify the properties for the control which is currently selected in the layout tree.

- **Preview Area (middle)**
  The preview area shows the HTML page which is created using the controls in the layout area. This page is refreshed each time, you choose the preview button (see below).
- **Controls Palette (right side)**
  Each control is represented by an icon. A tool tip is also provided which appears when you move the mouse pointer over the control. This tool tip also displays the XML tag which will be used in the XML layout.

  The palette is structured into sections, where each section represents a certain type of controls.

Previewing the Layout

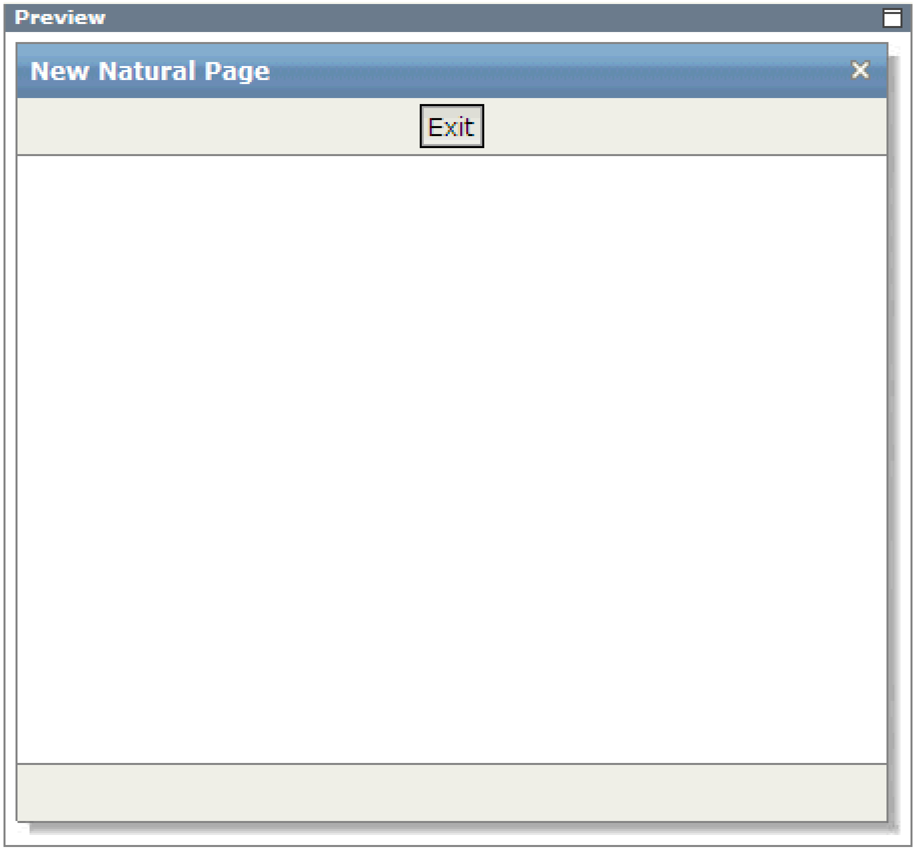
The layout tree inside the Layout Painter already contains some nodes that were copied from the template that you chose in the dialog in which you specified the name of the page. To see what the page looks like, preview the layout as described below.

Caution: If you use your browser's refresh function (F5 in Internet Explorer), the development workspace is reloaded and your latest unsaved changes are lost.

手順 9.2. To preview the layout

- Choose the following button which is shown at the top of the Layout Painter.

  The preview area is updated and you see the page. The page already contains a title bar, a header containing an **Exit** button, the page body and a status bar.



The preview area is a sensitive area. When you select a control in the preview area (for example, the title bar), this control is automatically selected in the layout tree.

Viewing the XML Code

When creating the layout, you can view the currently defined XML code.

手順 9.3. To view the XML code

- From the **Edit** tab of the Layout Painter, choose **XML**.

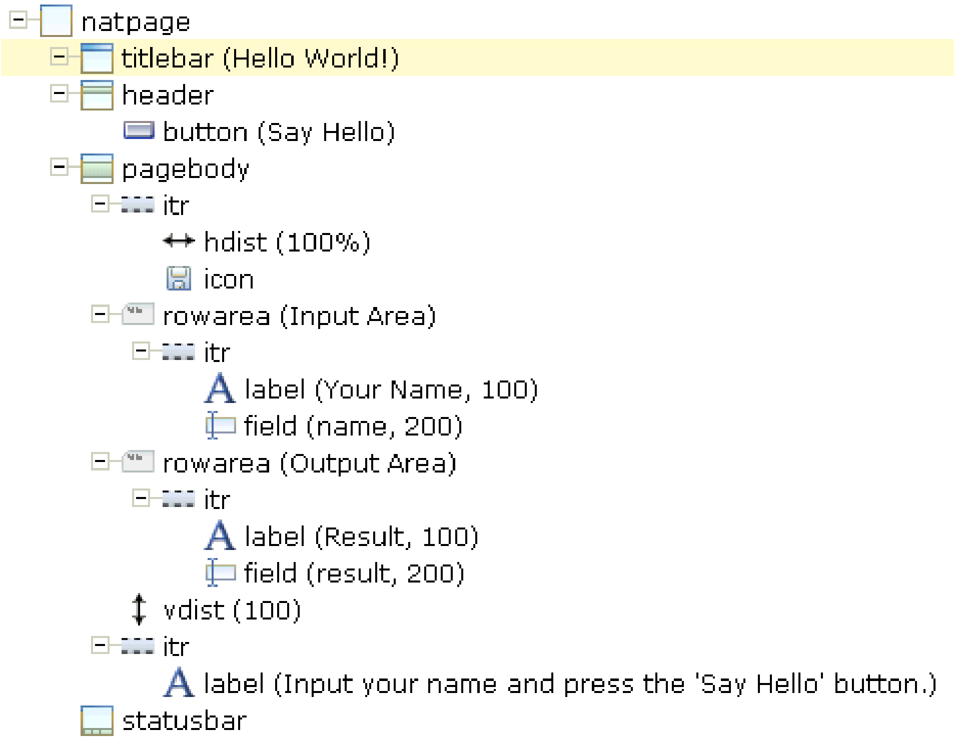
A dialog box appears. At this stage of the tutorial, it contains the following XML layout definition for the nodes which were copied from the template.

```
<natpage>
    <titlebar name="New Natural Page">
    </titlebar>
    <header withdistance="false">
        <button name="Exit" method="onExit">
        </button>
    </header>
    <pagebody>
    </pagebody>
    <statusbar withdistance="false">
    </statusbar>
</natpage>
```

You can now proceed with the next exercise: ***Writing the GUI Layout***.

10 Writing the GUI Layout

You will now create the layout for your 「Hello World!」 application. When you have completed all exercises in this chapter, the layout should look as shown below and the **XML code** should be the same as shown in the section *About this Tutorial*.



```
natpage
    titlebar (Hello World!)
    header
        button (Say Hello)
    pagebody
        itr
            hdist (100%)
            icon
        rowarea (Input Area)
            itr
                label (Your Name, 100)
                field (name, 200)
        rowarea (Output Area)
            itr
                label (Result, 100)
                field (result, 200)
        vdist (100)
        itr
            label (Input your name and press the 'Say Hello' button.)
    statusbar
```

This chapter contains the following exercises:

Tip: **Preview the layout** and **view the XML code** each time you have completed an exercise. If the system finds some wrong or missing definitions while generating the preview page, there will be a corresponding message in the status bar. From the **Home** tab of the Layout Painter, choose **Protocol** to get more information about these problems.

Specifying the Properties for the Natural Page

You will now specify the following for the Natural page:

- **Name for the Natural Adapter (`natsource`)**
  The value in the property `natsource` defines the name of the adapter. The adapter is a Natural object that your application will use to communicate with the page. It will be generated when you save the page layout.

  If you do not specify a value for `natsource`, the name that you have specified for the layout (without the extension ".xml") will be used as the name for the Natural adapter. If you want to use the adapter in a development environment other than Natural for Eclipse, you must make sure that the resulting name matches the naming conventions for Natural object names.

- **Handling of Strings (`natsinglebyte`)**
  Using the property `natsinglebyte`, you can specify how the strings displayed on this page are to be handled in the Natural application. Natural knows two types of strings: Unicode strings (format U) and code page strings (format A). By default, the strings displayed in web pages are mapped to Unicode strings in Natural. For this tutorial, you will specify that code page strings are to be used. Therefore, you will set the property `natsinglebyte` to "true".

  If you do not specify a value for `natsinglebyte` or when you set it to "false", Unicode strings will be used.

▶手順 10.1. To specify the properties for the Natural page

1. In the layout tree, select the node **natpage**.

   The properties for this control are now shown in the properties area at the bottom.

2. Specify the following properties:

Property	Value
`natsource`	HELLO-A
`natsinglebyte`	true

Specifying a Name for the Title Bar

You will now specify the string "Hello World!" which is to appear in the title bar of your application.

▶手順 10.2. To specify the name for the title bar

1. In the layout tree, select the node **titlebar (New Natural Page)**.

   The properties for this control are now shown in the properties area at the bottom. You can see the default entry "New Natural Page" for the `name` property.

2 Specify the following property:

Property	Value
`name`	Hello World!

When you click on the layout tree, the node in the layout tree changes to **titlebar (Hello World!)**.

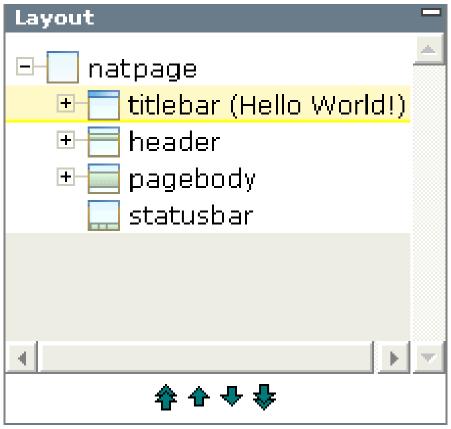
Note: Properties that are left blank are not shown in the XML code.

Using the Property Editor

You can also specify the property values using the Property Editor. In this case, you can access detailed help information on each property.

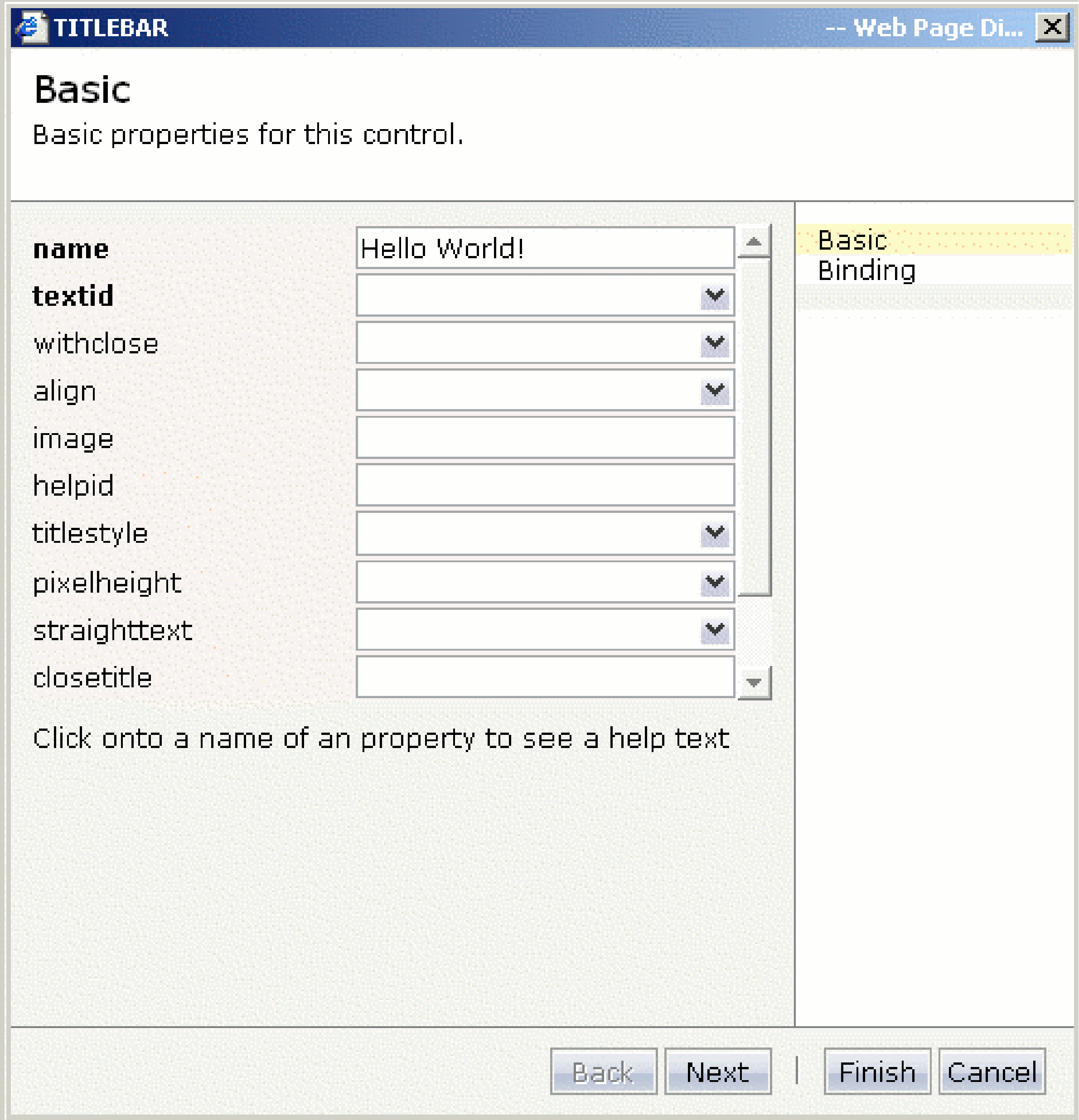
▶手順 10.3. To use the Property Editor

1 Select the control in the layout tree for which you need help, for example, the **titlebar (Hello World!)** node.



2 From the **Edit** tab of the Layout Painter, choose **Property Editor**.

The following dialog appears.



Basic

Basic properties for this control.

name	Hello World!
textid	
withclose	
align	
image	
helpid	
titlestyle	
pixelheight	
straighttext	
closetitle	

Click onto a name of an property to see a help text

The properties of the control are listed.

3 Click on the name of a property to display detailed information on this property. This information is shown below the list of properties.

4 Choose the **Finish** button to close the dialog.

Any changes you have applied in the dialog will be saved.

Specifying a Name and Method for the Button

You will now specify the string "Say Hello" which is to appear on the button. And you will specify the name of the method that is to be invoked when the user chooses this button.

▶手順 10.4. To specify the name and the method for the button

1 In the layout tree, open the **header** node.

> **Note:** By clicking the icon of a node, you hide or expand the node's subnodes.

You can now see the entry for the button with the default name "Exit".

2 Select the node **button (Exit)**.

3 Specify the following properties:

Property	Value
`name`	Say Hello
`method`	sayHello

The method needs to be programmed in the adapter. This will be explained later in this tutorial.

Adding the Input and Output Areas

The input and output areas in this tutorial are created using **Row Area** controls. These controls can be found in the **Container** section of the controls palette.

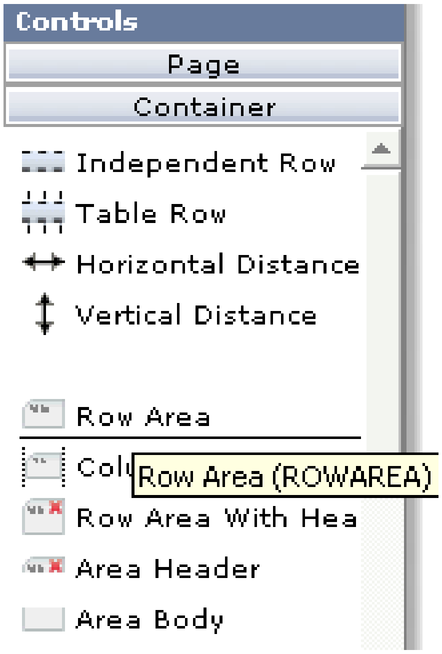
Each row area will contain an **Independent Row** control which in turn contains a **Label** and a **Field** control. These controls can be found in the **Controls** section of the controls palette.

For adding controls to your layout, you drag them from the controls palette onto the corresponding tree node in the layout tree. This is explained below.

▶手順 10.5. To create the input area

1 Open the **Container** section of the controls palette.

When you move the mouse over a control, a tool tip appears which also displays the control name which will be used in the XML layout. For example:



2 Drag the **Row Area** control from the controls palette onto the **pagebody** node in the layout tree.

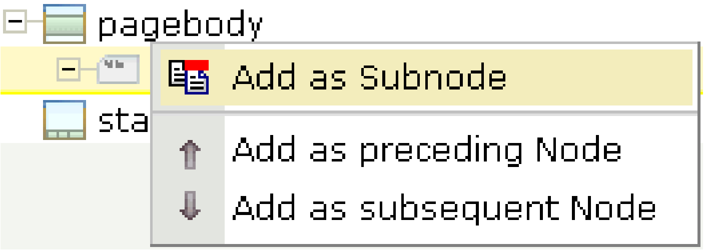
The row area is added as a subnode of the **pagebody** node. The new subnode is automatically selected so that you can maintain the properties of the row area directly in the properties area.

3 Specify the following property:

Property	Value
`name`	Input Area

4 Drag the **Independent Row** control from the controls palette onto the **rowarea (Input Area)** node in the layout tree.

When you drop information into the tree, the system will sometimes respond by offering a context menu with certain options about where to place the control. In this case, the following context menu appears.



Note: When you move the mouse outside the context menu, the context menu disappears. The control is not inserted in this case.

5 Choose the **Add as Subnode** command.

The control is now inserted below the **rowarea (Input Area)** node. The new node is shown as **itr**.

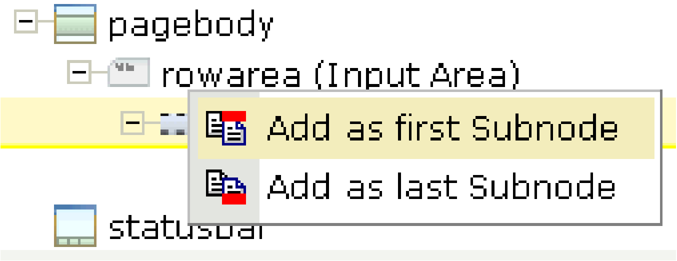
6 Open the **Controls** section of the controls palette.

7 Drag the **Label** control from the controls palette onto the **itr** node you have just inserted and specify the following properties:

Property	Value
name	Your Name
width	100

8 Drag the **Field** control from the controls palette onto the **itr** node you have just inserted.

A context menu appears and you have to specify where to place the control.



9 From the context menu, choose the **Add as last Subnode** command.

10 Specify the following properties for the field:

Property	Value
valueprop	name
width	200

手順 10.6. To create the output area

- Create the output area in the same way as the input area (add it as the last subnode of the **pagebody** node), with the following exceptions:

  Row Area

  Specify a different value for the following property:

Property	Value
name	Output Area

  Label

  Specify a different value for the following property:

Property	Value
name	Result

  Field

  Specify different values for the following properties:

Property	Value
valueprop	result
displayonly	true

  Note: To display the displayonly property, choose the **Appearance** tab at the bottom of the properties area. You can then select the required value from a drop-down list box.

Adding the Image

You will now add the image which is to be shown above the input area. To do so, you will use the **Icon** control which can be found in the **Controls** section of the controls palette.

Note: The image is provided in Application Designer's */cisdemos/images* directory.

手順 10.7. To add the image

1. Drag the **Icon** control from the controls palette onto the **pagebody** node in the layout tree.

   The icon is added as the last subnode of the **pagebody** node. It is automatically placed into an **itr** (independent row) node.

2. Specify the following property for the icon:

Property	Value
`image`	../cisdemos/images/hello.gif

3. Select the **itr** node containing the icon and choose the following button below the layout tree:

   ⤒

   The selected node is now moved up so that it appears as the first subnode of the **pagebody** node.

4. Specify the following property for the **itr** node:

Property	Value
`takefullwidth`	true

Adding a Horizontal Distance

When you preview the layout, you will see that the image you have just added appears centered.

You will now move the image to the right side of the page. To do so, you will use the **Horizontal Distance** control which can be found in both the **Controls** section and the **Container** section of the controls palette.

▶手順 10.8. To add the horizontal distance

1 Drag the **Horizontal Distance** control from the controls palette onto the **itr** node containing the icon.

2 From the resulting context menu, choose the **Add as first Subnode** command.

The node **hdist** is inserted into the tree.

3 Specify the following property:

Property	Value
`width`	100%

Adding an Instructional Text

You will now enter a text which is to appear below the output area and which tells the user what to do.

To do so, you will once again use the **Independent Row** control into which you will insert a **Label** control.

Note: The **Independent Row** control can be found in both the **Controls** section and the **Container** section of the controls palette.

▶手順 10.9. To add the independent row with the label

1 Drag the **Independent Row** control from the controls palette onto the **pagebody** node in the layout tree.

2 From the resulting context menu, choose the **Add as last Subnode** command.

The node **itr** is inserted into the tree.

3 Drag the **Label** control from the controls palette onto the **itr** node you have just created.

4 Specify the following properties for the label:

Property	Value
`name`	Input your name and press the 'Say Hello' button.
`asplaintext`	true

Note: Go to the **Appearance** tab to display the property `asplaintext`.

Adding a Vertical Distance

When you preview the layout, you will see that the text you have just added appears directly below the output area. You will now move the text 100 pixels to the bottom.

To do so, you will use the **Vertical Distance** control which can be found in both the **Controls** section and the **Container** section of the controls palette.

▶手順 10.10. To add the vertical distance

1 Drag the **Vertical Distance** control from the controls palette onto the **itr** node containing the label.

2 From the resulting context menu, choose the **Add as preceding Node** command.

The node **vdist** is inserted into the tree.

3 Specify the following property:

Properties	Value
`height`	100

Saving Your Layout

If you have not already done so, you should now save your layout.

When you save a layout for the first time, an HTML file is generated (in addition to the XML file) which is placed into the root directory of your application project. This HTML file is updated each time you save the layout.

The Natural adapter is also created when you save your layout for the first time. Later in this tutorial, you will import this adapter into your Natural library. Your application program will use the adapter to communicate with the page.

手順 10.11. To save the layout

- Choose the following button which is shown at the top of the Layout Painter.

  💾

You can now proceed with the next exercise: ***Setting Up Your Environment***.

11 Setting Up Your Environment

Before you start coding, you have to make specific definitions in your development environment.

This chapter contains the following exercises:

Setting Up Application Designer

Currently, there is nothing to configure for Natural pages.

Setting Up Your Development Environment for Natural

On the remote machine where you are going to develop the Natural code, a Natural Development Server must be installed and activated.

▶手順 11.1. To set up Natural Studio

1 Ask your administrator for the host name and the port number of the Natural Development Server.

2 Connect to the Natural Development Server. See *Accessing a Remote Development Environment* in the *Remote Development Using SPoD* documentation which is provided with Natural for Windows.

3 For this tutorial, create a new Natural library with the name `CISHELLO`.

> **Note:** It is recommended that you create a new Natural library for each Application Designer project.

Setting Up Your Runtime Environment for Natural

On the remote machine where you are going to test and execute the Natural code, a Web I/O Interface daemon (UNIX) or the Web I/O Interface Server (mainframe) must be installed and activated.

▶手順 11.2. To set up the runtime environment for Natural for UNIX

1 Ask your administrator for the host name and the port number of the Web I/O Interface daemon and the name of the script that is used to start up Natural sessions.

2. Edit the configuration file `<installdir>`*/WEB-INF/sessions.xml* (the location of `<installdir>` depends on your application server environment) and add the following entry:

   ```
   <session id="Execute samples" trace="false">
       <natural_server>host-name</natural_server>
       <natural_port>port-number</natural_port>
       <natural_program>script-name</natural_program>
   </session>
   ```

   where `host-name`, `port-number` and `script-name` are to be replaced with the values you received from your administrator.

   "Execute samples" is the entry that will later be available for selection in the logon page.

3. Restart the application server.

手順 11.3. To set up the runtime environment for Natural for Mainframes

1. Ask your administrator for the host name and the port number of the Web I/O Interface Server.
2. Edit the configuration file `<installdir>`*/WEB-INF/sessions.xml* (the location of `<installdir>` depends on your application server environment) and add the following entry:

   ```
   <session id="Execute samples" trace="false">
       <natural_server>host-name</natural_server>
       <natural_port>port-number</natural_port>
   </session>
   ```

   where `host-name` and `port-number` are to be replaced with the values you received from your administrator.

   "Execute samples" is the entry that will later be available for selection in the logon page.

3. Restart the application server.

You can now proceed with the next exercise: *Creating the Natural Code*.

12 Creating the Natural Code

This chapter contains the following exercises:

Importing the Adapter into Natural

You will now import the generated adapter into Natural to make it available to your application.

When you saved your page layout, Application Designer created the Natural adapter `HELLO-A` for your page. This is the name that you have specified earlier in this tutorial. Your application program will use the adapter to communicate with the page. The adapter has been generated into the following directory:

<installdir>/cisnatfirst/nat

Note: The location of *<installdir>* depends on your application server environment.

手順 12.1. To import the adapter

1 Import the adapter source into the Natural library `CISHELLO` which you have created earlier in this tutorial. To do so, use either drag-and-drop or the import function of the `SYSMAIN` utility.

The adapter looks as follows:

```
* PAGE1: PROTOTYPE       --- CREATED BY Application Designer --- /*<RO>>
* PROCESS PAGE USING 'XXXXXXXX' WITH
* NAME RESULT
DEFINE DATA PARAMETER
1 NAME (U) DYNAMIC
1 RESULT (U) DYNAMIC
END-DEFINE
*
PROCESS PAGE U'/cisnatfirst/helloworld' WITH
PARAMETERS
 NAME U'name'
  VALUE NAME
 NAME U'result'
  VALUE RESULT
END-PARAMETERS
*
*  TODO: Copy to your calling program and implement.
/*/*( DEFINE EVENT HANDLER
* DECIDE ON FIRST *PAGE-EVENT
*  VALUE U'nat:page.end'
*   /* Page closed.
*   IGNORE
*  VALUE U'sayHello'
*   /* TODO: Implement event code.
*   PROCESS PAGE UPDATE FULL
*  NONE VALUE
*   /* Unhandled events.
*   PROCESS PAGE UPDATE
* END-DECIDE
/*/*) END-HANDLER
*
END /*<<RO>
```

2 Stow the adapter.

Creating the Main Program

You will now create the main program which uses the adapter to display the page and which handles its events. The name of the program will be `HELLO-P` and you will store it in the library `CISHELLO`.

This description assumes that you are working with Natural Studio.

手順 12.2. To create the main program

1 Make sure that the library `CISHELLO` is selected.

2 From the **Object** menu, choose **New > Program**.

3 Enter a `DEFINE DATA` statement:

```

DEFINE DATA LOCAL
END-DEFINE
```

4 Import the adapter interface into the `DEFINE DATA` statement:

1. Place the cursor in `END-DEFINE`.
2. From the **Program** menu, choose **Import**.
3. In the resulting dialog box, select the **Adapter** option button.
4. Select the object `HELLO-A`.
5. Select all importable data fields.
6. Choose the **Import** button.

The result is your completed DEFINE DATA statement:

```
DEFINE DATA LOCAL
1 NAME (A) DYNAMIC
1 RESULT (A) DYNAMIC
END-DEFINE
```

5 Enter the PROCESS PAGE statement. The statement uses the page adapter to display the page in the web browser and to pass data to the controls on the page:

```
DEFINE DATA LOCAL
1 NAME (A) DYNAMIC
1 RESULT (A) DYNAMIC
END-DEFINE
*
PROCESS PAGE USING 'HELLO-A'
WITH NAME RESULT
```

6 Initialize the page data. In the page layout definition, the property name has been bound to the FIELD control with the label **Your Name**. For the property name, a parameter NAME has been generated into the parameter data area of the adapter. Thus, in order to preset the FIELD control, we will preset the variable NAME with the value "Application Designer".

```
DEFINE DATA LOCAL
1 NAME (A) DYNAMIC
1 RESULT (A) DYNAMIC
END-DEFINE
*
NAME := 'Application Designer'
PROCESS PAGE USING 'HELLO-A'
WITH NAME RESULT
```

7 Handle the events that can occur on the page. A template for the event handler code has been generated as a comment block into the page adapter HELLO-A. List the adapter HELLO-A and copy this comment block into your main program and terminate the program with an END statement:

```
DEFINE DATA LOCAL
1 NAME (A) DYNAMIC
1 RESULT (A) DYNAMIC
END-DEFINE
*
NAME := 'Application Designer'
PROCESS PAGE USING 'HELLO-A'
```

```
WITH NAME RESULT
*
DECIDE ON FIRST *PAGE-EVENT
  VALUE 'nat:page.end'
  /* Page closed.
    IGNORE
  VALUE 'sayHello'
  /* TODO: Implement event code.
    PROCESS PAGE UPDATE FULL
  NONE VALUE
  /* Unhandled events.
    PROCESS PAGE UPDATE
END-DECIDE
*
END
```

After the page has been displayed, the user raises events on the page by using the controls. The name of the raised event is then contained in the system variable `*PAGE-EVENT`. Depending on the event, the program modifies the page data, resends it to browser with a `PROCESS PAGE UPDATE FULL` statement and waits for the next event to occur.

The predefined event `nat:page.end` is raised when the user closes the page. The event `sayHello` is raised when the user chooses the **Say Hello** button. Previously in this tutorial, you have bound the event `sayHello` to this button while designing the page. The `NONE VALUE` block should always be defined as above. It contains the default handling of all events that are not handled explicitly.

8. When the event `sayHello` occurs, we want to display a greeting in the FIELD control with the label **Result**. Therefore, we modify the variable `RESULT` (which is bound to the corresponding FIELD control in the page layout) accordingly before we resend the page data.

```
DEFINE DATA LOCAL
1 NAME (A) DYNAMIC
1 RESULT (A) DYNAMIC
END-DEFINE
*
NAME := 'Application Designer'
PROCESS PAGE USING 'HELLO-A'
WITH NAME RESULT
*
DECIDE ON FIRST *PAGE-EVENT
  VALUE 'nat:page.end'
  /* Page closed.
    IGNORE
  VALUE 'sayHello'
  /* TODO: Implement event code.
    COMPRESS 'Hello, ' NAME '!' TO RESULT
    PROCESS PAGE UPDATE FULL
  NONE VALUE
  /* Unhandled events.
```

```
    PROCESS PAGE UPDATE
END-DECIDE
*
END
```

The main program is now complete.

If you have not yet saved the program, save or stow it now with the name "HELLO-P".

9 Catalog all modules in the library `CISHELLO`.

Testing the Completed Application

You will now run the application in your web browser and check whether it provides the desired result.

The generated HTML file *helloworld.html* (which is updated each time you save your layout) can be found within the root of your application project, that is in `<installdir>`*/cisnatfirst*.

This HTML page has some prerequisites concerning the browser workplace in which it is running. Therefore, it is per se not usable as a directly accessible page but needs to be embedded into a frame providing a defined set of functions.

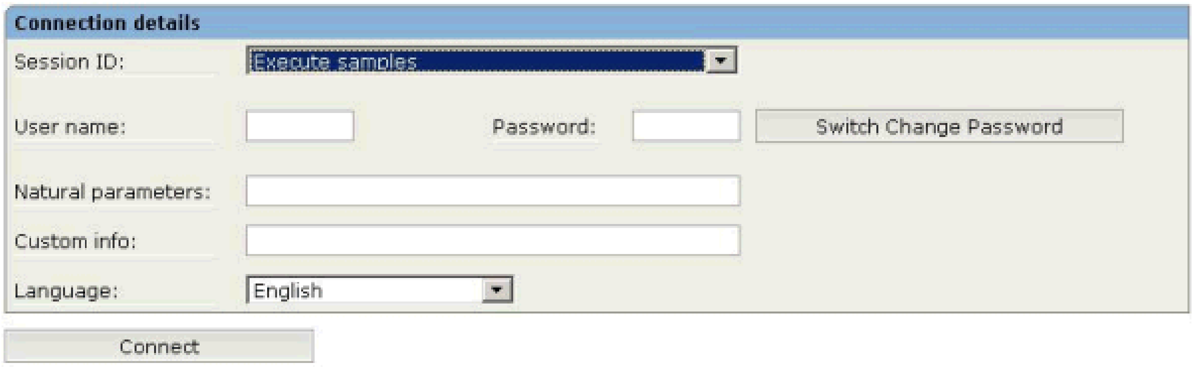
It is necessary to logon to Natural before starting an application. Therefore, Natural applications are started using a logon page.

手順 12.3. To test the application

1 Enter the following URL inside your browser:

```
http://localhost:8080/cisnatural/servlet/StartCISPage?PAGEURL=/cisnatural/NatLogon.html
```

The logon page should now appear.



Connection details

Session ID: Execute samples

Language: English

If the logon page is not displayed, check the following:

- URLs are case-sensitive. Double-check your input.
- Check whether the file *NatLogon.html* is available in the directory *cisnatural*.

2 On the logon page, select the entry **Execute samples** from the **Session ID** drop-down list box. You have prepared this entry earlier in this tutorial when you have set up the runtime environment.

3 Provide your user ID and password valid for the machine on which the Natural application will be running.

4 In the **Natural parameters** text box, enter the Natural command line which is necessary to start your application:

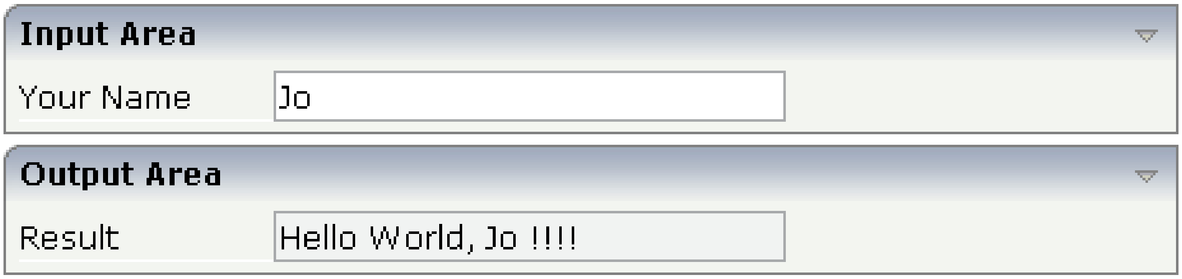
```
STACK=(LOGON CISHELLO;HELLO-P;FIN)
```

5 Choose the **Connect** button.

Your application should be started now.

6 Enter your name and choose the **Say Hello** button.

The page should now successfully 「talk」 to your adapter.



Your Name: Jo

Result: Hello World, Jo !!!!

You have now completed this tutorial. See the remaining section of these *First Steps* for **some background information**.

13 Some Background Information

This chapter covers the following topics:

Name Binding between Controls and Adapter

Which are the critical parts when building the 「Hello World!」 application?

- The NATPAGE control in the layout points to the name of the adapter object (property `natsource`).
- The FIELD control in the layout points to the property name of the adapter (property `valueprop`).
- The BUTTON control in the layout points to the event `sayHello()` of the adapter (property `method`).

There is a name binding between the layout definition and its corresponding adapter. This is the simple and effective approach of the Application Designer's development process: The adapter represents a logical abstraction of what the page displays. All layout definitions are kept in the page - all the logic is kept in the adapter. (Or better: behind the adapter. The adapter itself should only be a facade to the 「real」 application logic.)

Data Exchange at Runtime

What happens at runtime?

- When the user starts a Natural session from the logon page, the Natural program that the user specified in the command line is started.
- The Natural program executes a `PROCESS PAGE` statement, using an adapter.
- The `PROCESS PAGE` statement passes the name of the HTML page to be used and the initial page data to the browser.
- The browser displays the page. JavaScript code on the page distributes the initial data to the controls.
- The user provides some input, for example, enters the name. The content change is stored inside the page. The Natural program is not yet involved.
- The user does something which causes a flush of the changes (for example, the user chooses a button). Therefore, all registered data changes are packaged and are sent through the adapter to the Natural program, including the information which event has been raised.
- The Natural program receives the modified data.
- The system variable `*PAGE-EVENT` receives the name of the raised event.
- The event handler in the Natural program modifies the data and resends it to the page using a `PROCESS PAGE UPDATE` statement.

- And so forth.

With a standard HTTP connection, only the changed content of the screen is passed when operating on one page. The layout is kept stable in the browser. Consequently, there is no flickering of the page due to page reloading.

All steps described in the list above are done completely transparent to your adapter; i.e. you do not have to cope with session management, stream parsing, error management, building up HTML on the server, etc. You just have to provide an intelligent HTML page by defining it in the Layout Painter and an adapter object.

Files and their Locations

Have a look at the files created for your 「Hello World!」 application and take notice of the directory in which they are located.

All files are located in the directory `<installdir>`*/cisnatural/cisnatfirst*. The `<installdir>`*/cisnatural* directory is the directory of the web application instance. The `<installdir>`*/cisnatural/cisnatfirst* directory is the directory that has been created for your new project.

- The XML layout definition is kept in the `<installdir>`*/cisnatural/cisnatfirst/xml* directory.
- The generated HTML page is kept directly in the project directory. There are also some other files inside this directory that start with "ZZZZ". These files are temporary files used when previewing pages inside the Layout Painter.
- The generated Natural adapters are kept in the directory `<installdir>`*/cisnatural/cisnatfirst/nat*.
- In the directory `<installdir>`*/cisnatural/cisnatfirst/accesspath*, 「access restriction」 files are generated. If you view these files inside a normal text editor (such as Notepad), you see that one file is maintained for each page; it holds the information about which properties are accessed by the page.

14 Developing the User Interface

In the *First Steps* tutorial, you have developed a small rich internet program step by step. In this tutorial, you have already performed most of the steps required to develop a rich internet application.

The general procedure to develop a rich internet application with Natural for Ajax is as follows:

1. Use Application Designer to design the web pages that form the user interface of your application.
2. Generate a Natural adapter for each page (by saving the page). The adapter is a Natural object that forms the interface between the application code and the web page.
3. Use one of the Natural tools (Natural Studio or Natural for Eclipse) to write the Natural application programs that contain the business logic and use adapters to exchange data with the web pages.

In this chapter, the first two steps (design and adapter) are explained in more detail. Step 3 (business logic) is described in the section *Developing the Application Code* which also addresses advanced topics that are not covered in the tutorial.

This chapter covers the following topics:

For detailed information on how to use the Application Designer development workplace, see *Development Tools* in the Application Designer documentation. The latest version of the Application Designer documentation is available at *http://documentation.softwareag.com/crossvision/cit/overview.htm*. The information which is provided below describes the most important differences which pertain to Natural for Ajax.

Starting the Development Workplace

The Application Designer development workplace is the central point for starting tools for layout development.

▶手順 14.1. To start the development workplace

1 Make sure that your application server is running.

2 Invoke your browser and start the development workplace with the following URL:

```
http://<host>:<port>/cisnatural
```

where `<host>` is the name of the machine on which your application server is installed and `<port>` is the port number of your application server.

Note: If you have not defined another port number during installation, the default port number is "8080".

Creating an Application Designer Project

First you create an Application Designer project using the Project Manager. The project contains the layouts of the web pages you design, the files that are generated from the layouts and are required to run your application and additional files that make your application multi language capable and supply help information. See also *Creating a Project* in the tutorial.

Note: Detailed information on the Project Manager is provided in the Application Designer documentation at *http://documentation.softwareag.com/crossvision/cit/devtools/dt-projectmanager.htm*.

All files in your Application Designer project are stored in one directory on the application server where Natural for Ajax is installed. The name of the directory corresponds to the project name you have chosen. The location of the directory depends on the application server:

- **JBoss Application Server**
  `<installdir>`*/server/default/deploy/njx*`<nnn>`*.ear/cisnatural.war*
- **Sun Java System Application Server**
  `<installdir>`*/domains/domain1/applications/j2ee-apps/njx*`<nnn>`*.ear/cisnatural.war*

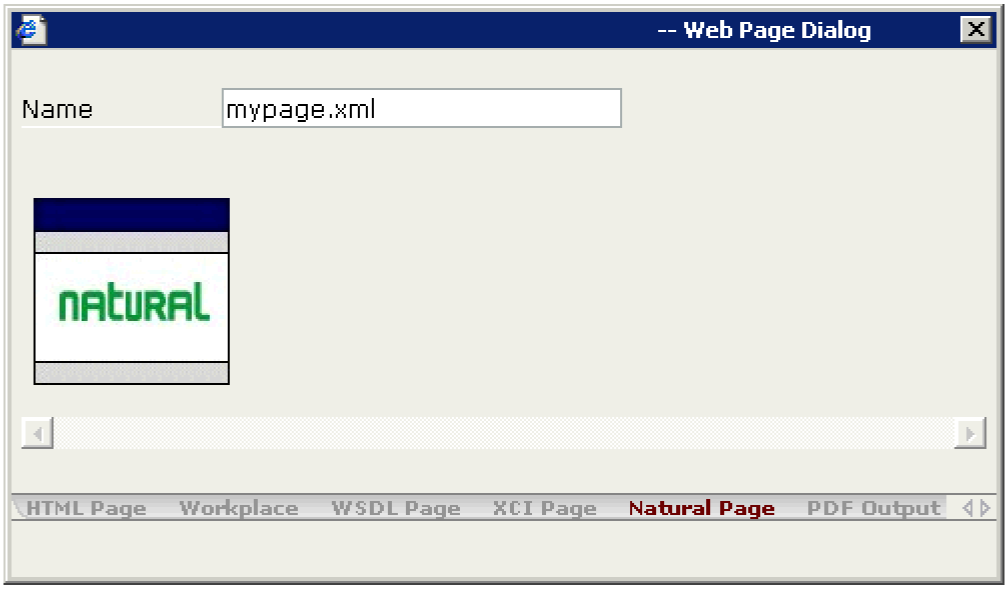
where `<installdir>` is the directory in which your application server is installed and `<nnn>` is the current Natural for Ajax version.

Creating a Natural Page

In order to create the layout of your web pages, you use Application Designer's Layout Painter.

Note: Detailed information on the Layout Painter is provided in the Application Designer documentation at *http://documentation.softwareag.com/crossvision/cit/devtools/dt-layout-over.htm*.

Add a page layout to your project as described in *Creating a New Layout* in the tutorial (select the template for the Natural page).

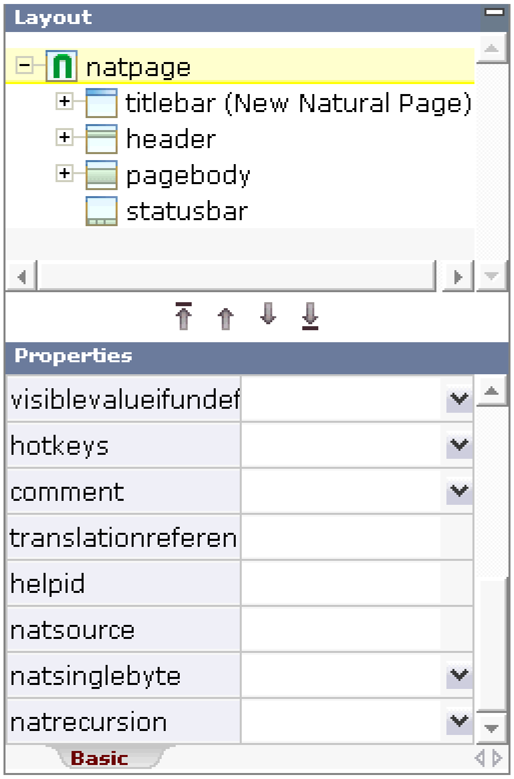


Note: More detailed information on creating a layout is provided in the Application Designer documentation at *http://documentation.softwareag.com/crossvision/cit/devtools/dt-layout-getstart.htm#dt-layout-getstart-create*.

Specifying Properties for the Natural Page

In order to specify generation options for the new page, you specify values for certain properties that are specific for Natural pages.

To define properties, you select the node **natpage** in the layout tree of the Layout Painter. The properties for this control are then shown in the properties area at the bottom. When you scroll down in the properties area, you can see the Natural-specific properties.



The following properties are available for the Natural page:

Property	Description
`natsource`	Specifies a name for the Natural adapter object that will later be generated from your page layout. During adapter generation, this name is checked to match the Natural naming conventions for objects. If you do not specify a name here, the adapter name is taken from the layout name. This might result in names that are not valid for Natural objects. These adapters can only be used in Natural for Eclipse.
`natsinglebyte`	Specifies whether string properties of the page are to be mapped to Unicode strings (U) or code page strings (A) in Natural. The value "true" means code page strings. The value "false" means Unicode strings (default).
`natrecursion`	Properties of controls used in the page might have a recursive structure. These structures are mapped to multi-dimensional arrays in the Natural adapter. Natural arrays are limited to three dimensions. Therefore, the recursion depth of these structures can be limited using this property.

Designing the Page

Design your Natural page by dragging controls and containers from the controls palette onto the corresponding node in the layout tree or to the HTML preview. This has already been explained in the section *Writing the GUI Layout* of the tutorial.

Note: More detailed information on defining the layout is provided in the Application Designer documentation at *http://documentation.softwareag.com/crossvision/cit/devtools/dt-layout-xml.htm*.

Binding Properties and Methods

Many of the controls you use on your page have properties that can be controlled by the application. Also the controls can raise events that your application may wish to handle. The next step is therefore assigning identifiers to each of these properties and events under which your application can later address them. This procedure is called 「binding」.

To get an overview which properties and events are bindable to application variables and events, it is a good idea to select a control in the layout tree and open the Event Editor as described in the Application Designer documentation at *http://documentation.softwareag.com/crossvision/cit/devtools/dt-layout-xml.htm#dt-layout-xml-attrib-eventeditor*.

The Event Editor displays only those properties of controls that can be bound to application variables and events. It indicates also which properties must be bound mandatorily. The usage and meaning of each of the properties and events is described for each control in the following sections of this Natural for Ajax documentation:

- *Working with Controls*
- *Working with Grids*
- *Working with Trees*
- *Working with Menus*
- *Non-Visual Controls and Hot Keys*

As an example for property and event binding, see the following sections in the *First Steps* tutorial:

- *Using the Property Editor*
- *Specifying a Name and Method for the Button*

Previewing the Layout

To find out how the current layout definitions are rendered on the page, preview the layout as described in the Application Designer documentation at ***http://documentation.softwareag.com/crossvision/cit/devtools/dt-layout-xml.htm#dt-layout-xml-preview***.

Viewing the Protocol

The protocol contains warnings and error messages that might occur while you design and preview your page. For further information, see the Application Designer documentation at ***http://documentation.softwareag.com/crossvision/cit/devtools/dt-layout-info.htm#dt-layout-view-protocol***.

Saving the Layout

Save the page layout as described in ***Saving Your Layout*** in the tutorial.

Other than with Java adapters (which are described in the Application Designer documentation), you do not use the Code Assistant (which is part of the Layout Painter) to generate adapter code interactively. For Natural pages the adapter code is generated completely from the page properties and the property and event bindings that you specified previously. An adapter is generated automatically when you save the layout for the first time. It is updated each time you save the layout.

Generating the Adapter

When you save the layout, a Natural adapter is generated according to the following rules:

Location	The adapter is generated into the subdirectory *nat* of your project directory. The name of the project directory corresponds to the project name. The location of the directory depends on the application server. See ***Creating an Application Designer Project***.
Name	The name of the adapter is determined by the properties you have set. See ***Specifying Properties for the Natural Page***.
Property identifiers	For each control property that has been bound to an identifier (as described in ***Binding Properties and Methods***) a parameter in the parameter data area of the adapter is

	generated.The identifier is therefore validated against the Natural naming conventions for user-defined variables and translated to upper-case. If an identifier does not comply to these rules, a warning is generated into the protocol and as a comment into the adapter code. Additionally, the name must comply to the naming conventions for XML entities. This means especially that the name must start with a character. To achieve uniqueness within 32 characters, the last four characters are (if necessary) replaced by an underscore, followed by a three-digit number.
Event identifiers	For each event that can be raised by a control on the page, an event handler skeleton is generated as a comment into the adapter. **注意:** Some controls raise events whose names are dynamically constructed at runtime. For these events, no handler skeleton can be generated. The control reference contains information about these additional events. The event identifiers are not validated.

Data Type Mapping

Several Application Designer controls have properties for which a data type can be specified. An example is the FIELD control. It has a `valueprop` property which can be restricted to a certain data type. The data type is used at runtime to validate user input. At generation time (that is, when a Natural adapter is generated for the page), the data type determines the Natural data format of the corresponding adapter parameter.

The following table lists the data types used in Application Designer and the corresponding Natural data formats.

Application Designer	Natural
`color`	`A` or `U` (depending on the NATPAGE property `natsinglebyte`). The string must contain an RGB value, for instance "#FF0000" for the color red.
`date`	`D (YYYYMMDD)`
`float`	`F4`
`int`	`I4`
`long`	`P19`
`time`	`T (HHIISS)`
`timestamp`	`T (YYYYMMDDHHIISST)`
`N` *`n.n`*	`N`*`n.n`*
`P` *`n.n`*	`P`*`n.n`*
`string` (default)	`A` or `U` dynamic (depending on the NATPAGE property `natsinglebyte`).
`string` *`n`*	`A`*`n`* or `U`*`n`* (depending on the NATPAGE property `natsinglebyte`).

Application Designer	Natural
xs:double	F8
xs:byte	I1
xs:short	I2

15 Developing the Application Code

This chapter covers the following topics:

Natural for Ajax Tools, which is an optional plug-in for Natural Studio, allows you to use some of the Natural for Ajax functionality which is described in this chapter directly from within Natural Studio. For further information, see *Natural for Ajax Tools* in the *Natural Studio Extensions* documentation which is provided for Natural for Windows.

Importing the Adapter

After having generated the adapter, the next step is making it available to your Natural development project.

As described previously, the adapter code is generated into a directory in your application server environment. The way you access the adapter depends on the Natural development tool you use.

The following topics are covered below:

- Importing the Adapter Using Natural Studio
- Importing the Adapter Using Natural for Eclipse

Importing the Adapter Using Natural Studio

It is assumed that your development library is located on a Natural development server and that you have mapped this development server in Natural Studio.

▶手順 15.1. To import the adapter from a remote environment

- Use drag-and-drop.

  Or:

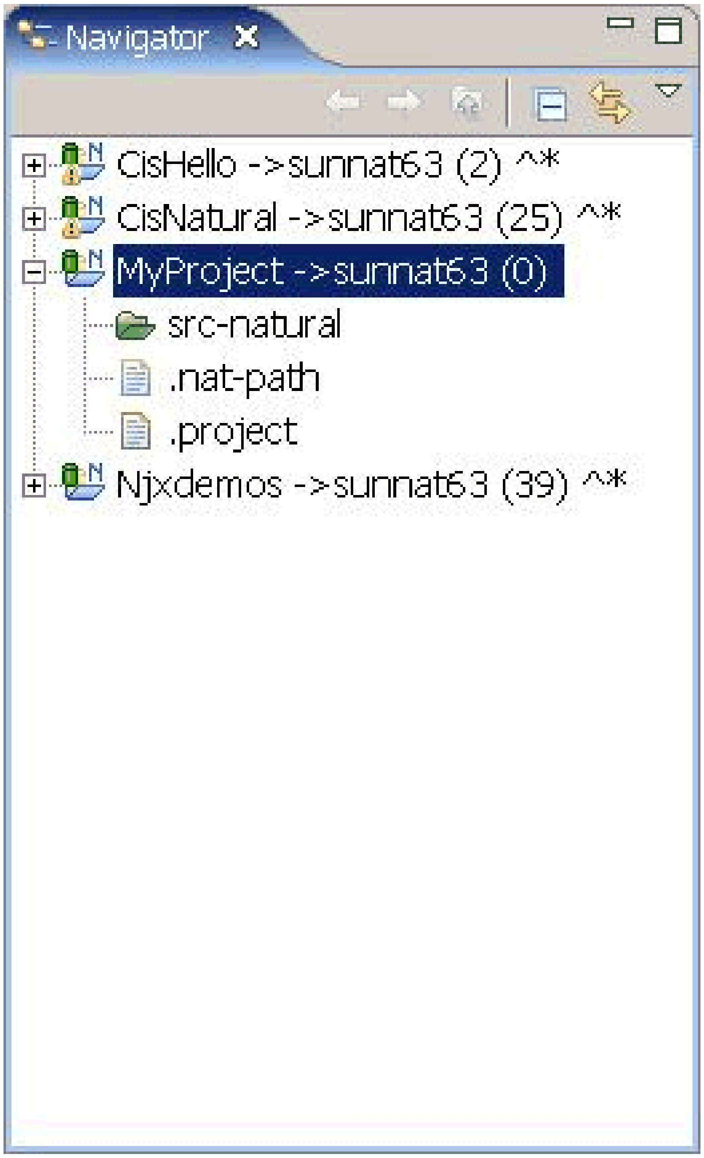
  Remote UNIX environment only: Use the import function of `SYSMAIN`.

Importing the Adapter Using Natural for Eclipse

It is assumed that you have

- installed Natural for Eclipse,
- installed Application Designer's Eclipse plug-in,
- created a Natural project in Eclipse,
- established a target for the Natural project (a Natural development server).

The Navigator view will then look similar to the following:



CisHello ->sunnat63 (2) ^*

CisNatural ->sunnat63 (25) ^*

MyProject ->sunnat63 (0)

src-natural

.nat-path

.project

Njxdemos ->sunnat63 (39) ^*

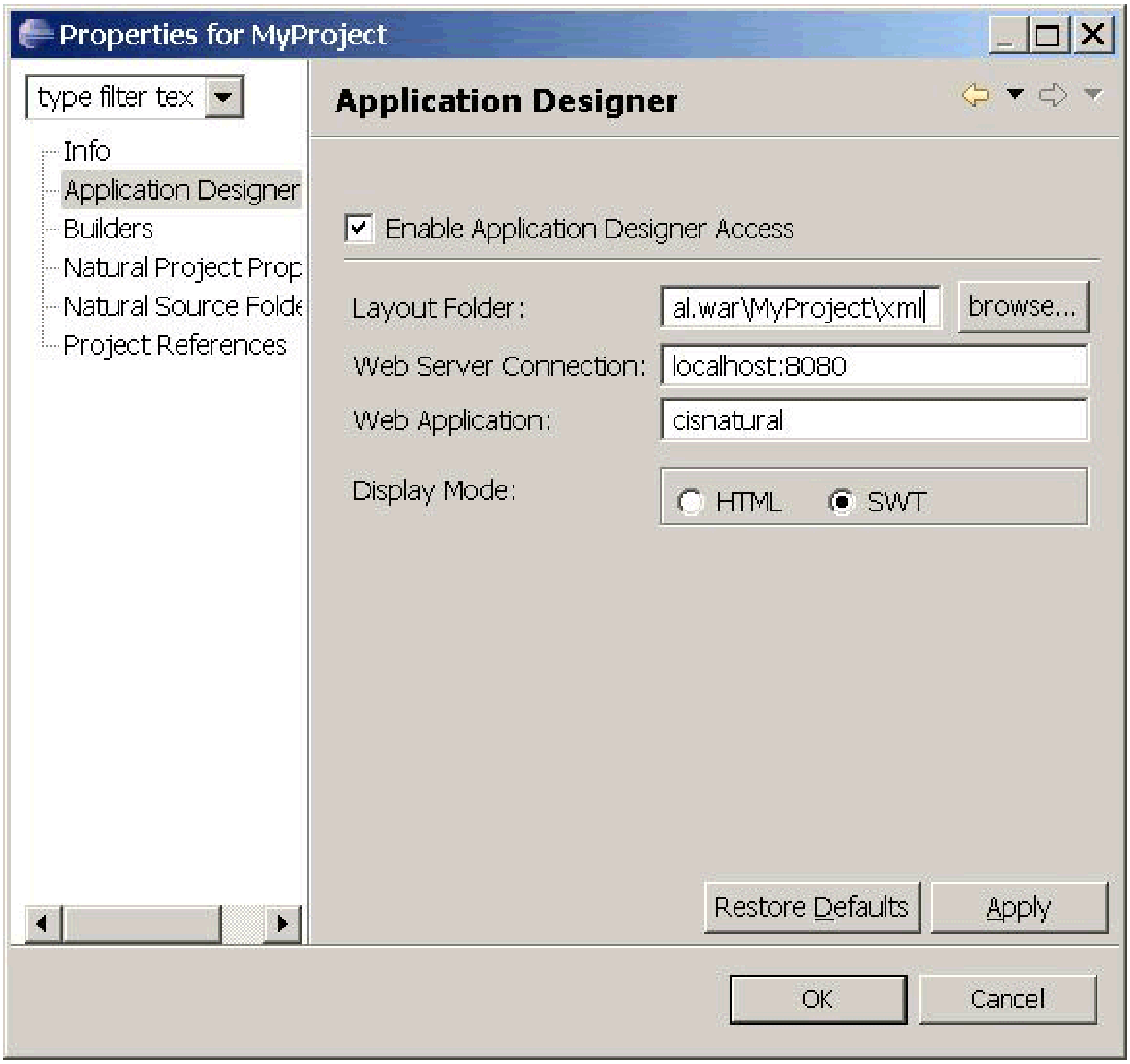
▶手順 15.2. To import the adapter from a remote environment

1 Proceed as described below to create the **Page Layouts** folder in your Natural project. This is the folder where you edit your page layouts with Application Designer.

1. Invoke the **Properties** dialog for your Natural project.

2. Set the Application Designer properties as follows:

Option	Description
Layout Folder	Specify the application server directory in which the page layouts of your project are stored.
Web Server Connection	Specify host name and port number of your application server.
Web Application	Specify "cisnatural".


Properties for MyProject

Application Designer

☑ Enable Application Designer Access

Layout Folder: al.war\MyProject\xml

Web Server Connection: localhost:8080

Web Application: cisnatural

Display Mode: ○ HTML ● SWT

2 Proceed as described below to create an additional folder in your Natural project. This is the folder in which the generated adapters are located.

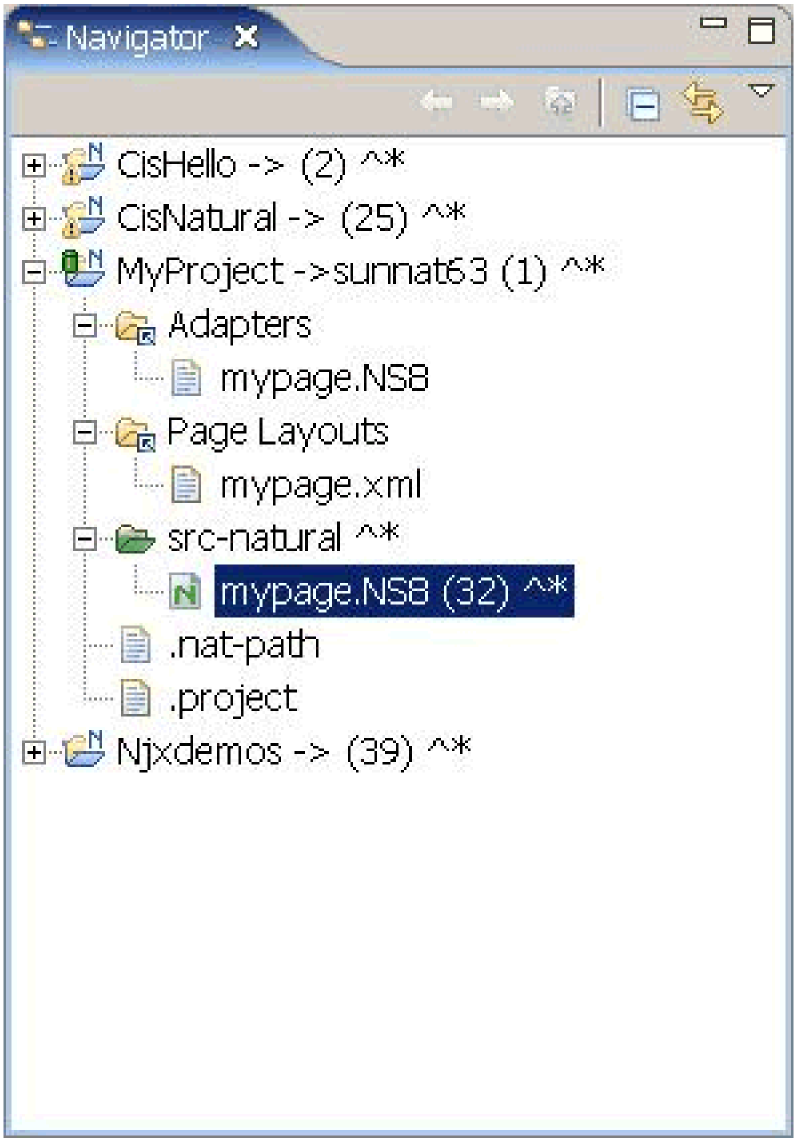
1. Select your Natural project, invoke the context menu and choose **New > Natural Folder**.

2. Expand the resulting dialog by choosing the **Advanced** button.
3. Specify a folder name of your choice (for example, "Adapters").
4. Enable the **Link to folder in the file system** check box and specify the application server directory in which the generated adapters of your project are stored.

Now you have access to your page layouts and adapters in your Natural project.

3 Copy or move the generated adapter from the new folder you have just created into your Natural source folder.

The Navigator view should now look similar to the following (with the new folders for the page layouts and adapters, and with your adapter in the Natural source folder).



```
⊞ CisHello -> (2) ^*
⊞ CisNatural -> (25) ^*
⊟ MyProject ->sunnat63 (1) ^*
    ⊟ Adapters
        mypage.NS8
    ⊟ Page Layouts
        mypage.xml
    ⊟ src-natural ^*
        mypage.NS8 (32) ^*
    .nat-path
    .project
⊞ Njxdemos -> (39) ^*
```

4 Catalog or stow the adapter in the Natural source folder. To do so, you have to upload and compile the adapter with Natural for Eclipse.

Creating the Main Program

After you have imported the adapter, you create a program that calls the adapter to display the page and handles the events that the user raises on the page. This program can be a Natural program, subprogram, subroutine or function. We use a Natural program as example.

The adapter already contains the data structure that is required to fill the page. It contains also a skeleton with the necessary event handlers. You can therefore create a program with event handlers from an adapter in a few steps.

Open or list the adapter in the development tool of your choice (Natural Studio or Natural for Eclipse).

```
* PAGE1: PROTOTYPE       --- CREATED BY Application Designer ---
* PROCESS PAGE USING 'XXXXXXXX' WITH
* FIELD1 FIELD2
DEFINE DATA PARAMETER
1 FIELD1 (U) DYNAMIC
1 FIELD2 (U) DYNAMIC
END-DEFINE
*
PROCESS PAGE U'/MyProject/mypage' WITH
PARAMETERS
 NAME U'field1'
  VALUE FIELD1
 NAME U'field2'
  VALUE FIELD2
END-PARAMETERS
*
*  TODO: Copy to your calling program and implement.
/*/*( DEFINE EVENT HANDLER
* DECIDE ON FIRST *PAGE-EVENT
*  VALUE U'nat:page.end'
*   /* Page closed.
*   IGNORE
*  VALUE U'onExit'
*   /* TODO: Implement event code.
*   PROCESS PAGE UPDATE FULL
*  NONE VALUE
*   /* Unhandled events.
*   PROCESS PAGE UPDATE
* END-DECIDE
/*/*) END-HANDLER
```

```
*
END
```

Create a new program, copy the adapter source into the program and then proceed as follows:

- Remove the comment lines in the header.
- Change `DEFINE DATA PARAMETER` into `DEFINE DATA LOCAL`.
- Replace the `PROCESS PAGE` statement with a `PROCESS PAGE USING` *`operand4`* statement, where *`operand4`* stands for the name of your adapter.
- Remove the comment lines that surround the `DECIDE` block.
- Uncomment the `DECIDE` block.

Your program should now look as follows:

```
DEFINE DATA LOCAL
1 FIELD1 (U) DYNAMIC
1 FIELD2 (U) DYNAMIC
END-DEFINE
*
PROCESS PAGE USING 'MYPAGE'
*
DECIDE ON FIRST *PAGE-EVENT
 VALUE U'nat:page.end'
  /* Page closed.
  IGNORE
 VALUE U'onExit'
  /* TODO: Implement event code.
  PROCESS PAGE UPDATE FULL
 NONE VALUE
  /* Unhandled events.
  PROCESS PAGE UPDATE
END-DECIDE
*
END
```

Stow the program with a name of your choice. The resulting program can be executed in a browser where it displays the page. However, it does not yet do anything useful, because it handles the incoming events only in a default way and contains no real application logic.

Structure of the Main Program

The main program that displays the page and handles its events has the following general structure:

- A PROCESS PAGE USING statement with the page adapter. The PROCESS PAGE statement displays the page in the user's web browser and fills it with data. Then, it waits for the user to modify the data and to raise an event.
- A DECIDE block with a VALUE clause for each event that shall be explictly handled.
- A default event handler for all events that shall not be explicitly handled.

Each event handler does the following:

- It processes the data the has been returned from the page in the user's web browser.
- It performs a PROCESS PAGE UPDATE FULL statement to re-execute the previous PROCESS PAGE USING statement with the modified data and to wait for the next event.

The default event handler does not modify the data. It does the following:

- It performs a PROCESS PAGE UPDATE statement to re-execute the previous PROCESS PAGE USING statement and to wait for the next event.

Handling Page Events

When the PROCESS PAGE statement receives an event, the data structure that was passed to the adapter is filled with the modified data from the page and the system variable *PAGE-EVENT is filled with the name of the event. Now, the corresponding VALUE clause in the DECIDE statement is met and the code in the clause is executed.

The application handles the event by processing and modifying the data and resending it to the page with a PROCESS PAGE UPDATE FULL statement. Alternatively, it uses the PROCESS PAGE UPDATE statement without the FULL clause in order to resend the original (not modified) data.

Built-in Events and User-defined Events

There are built-in events and user-defined events.

Built-in Events

The following built-in events can be received from the page:

nat:page.end
This event is raised when the user closes the page with the Close button in the upper right corner of the page, opens another page or closes the web browser.

nat:page.default
This event is sent if the Natural for Ajax client needs to synchronize the data displayed on the page with the data held in the application. It is usually handled in the default event handler and just responded with a `PROCESS PAGE UPDATE`.

Other built-in events can be sent by specific controls. These events are described in the control reference.

User-defined Events

User-defined events are those events that the user has assigned to controls while designing the page layout with the Layout Painter. The names of these events are freely chosen by the user. The meaning of the events is described in the control reference.

Sending Events to the User Interface

The `PROCESS PAGE UPDATE` statement can be accompanied by a `SEND EVENT` clause. With the `SEND EVENT` clause, the application can trigger certain events on the page when resending the modified data.

The following events can be sent to the page:

nat:page.message

This event is sent to display a text in the status bar of the page. It has the following parameters:

Name	Format	Value
type	A or U	Sets the icon in the status bar ("S"=success icon, "W"=warning icon, "E"=error icon).
short	A or U	Short text.
long	A or U	Long text.

nat:page.valueList

This event is sent to pass values to a FIELD control with value help on request (see also the description of the FIELD control in the control reference). It has the following parameters:

Name	Format	Value
id	A or U	A list of unique text identifiers displayed in the FIELD control with value help. The list must be separated by semicolon characters.
text	A or U	A list of texts displayed in the FIELD control with value help. The list must be separated by semicolon characters.

nat:page.xmlDataMode

This event is sent to switch several properties of controls on the page in one call to a predefined state. The state must be defined in an XML file that is expected at a specific place. See the information on XML property binding in the Application Designer documentation for further information.

Name	Format	Value
data	A or U	Name of the property file to be used.

Using Pop-Up Windows

A rich GUI page can be displayed as a modal pop-up in a separate browser window. A modal pop-up window can open another modal pop-up window, thus building a window hierarchy. If a `PROCESS PAGE` statement and its corresponding event handlers are enclosed within a `PROCESS PAGE MODAL` block, the corresponding page is opened as a modal pop-up window.

The application can check the current modal pop-up window level with the system variable `*PAGE-LEVEL`. `*PAGE-LEVEL = 0` indicates that the application code is currently dealing with the main browser window. `*PAGE-LEVEL > 0` indicates that the application code is dealing with a modal pop-up window and indicates the number of currently stacked pop-up windows.

In order to modularize the application code, it makes sense to place the code for the handling of a modal pop-up window and the enclosing PROCESS PAGE MODAL block in a separate Natural module, for instance, a subprogram. Then the pop-up window can be opened with a CALLNAT statement and can thus be reused in several places in the application.

Example program MYPAGE-P:

```
DEFINE DATA LOCAL
1 FIELD1 (U) DYNAMIC
1 FIELD2 (U) DYNAMIC
END-DEFINE
*
PROCESS PAGE USING 'MYPAGE-A'
*
DECIDE ON FIRST *PAGE-EVENT
 VALUE U'nat:page.end'
  /* Page closed.
  IGNORE
 VALUE U'onPopup'
  /* Open a pop-up window with the same fields.
  CALLNAT 'MYPOP-N' FIELD1 FIELD2
  PROCESS PAGE UPDATE FULL
 NONE VALUE
  /* Unhandled events.
  PROCESS PAGE UPDATE
END-DECIDE
*
END
```

Example subprogram MYPOP-N:

```
DEFINE DATA PARAMETER
1 FIELD1 (U) DYNAMIC
1 FIELD2 (U) DYNAMIC
END-DEFINE
*
/* The following page will be opened as pop-up.
PROCESS PAGE MODAL
*
 PROCESS PAGE USING 'MYPOP-A'
*
 DECIDE ON FIRST *PAGE-EVENT
  VALUE U'nat:page.end'
   /* Page closed.
   IGNORE
  NONE VALUE
   /* Unhandled events.
   PROCESS PAGE UPDATE
 END-DECIDE
*
END-PROCESS
*
END
```

Using Natural Maps

Rich internet applications written with Natural for Ajax need not only consist of rich GUI pages, but may also use classical maps. This is especially useful when an application that was originally written with maps shall only be partly changed to provide a rich GUI. In this case the application can run under Natural for Ajax from the very beginning and can then be 「GUIfied」 step by step.

Navigating between Pages and Maps

Due to the similar structure of programs that use maps and programs that use adapters, it is easy for an application to leave a page and open a map, and vice versa. For each rich GUI page, you write a program that displays the page and handles its events. For each map, you write a program that displays the map and handles its events. In an event handler of the page, you call the program that handles the map. In an 「event handler」 of the map, you call the program that handles the page.

Example for program `MYPAGE-P`:

```
DEFINE DATA LOCAL
1 FIELD1 (U20)
1 FIELD2 (U20)
END-DEFINE
*
PROCESS PAGE USING 'MYPAGE'
*
DECIDE ON FIRST *PAGE-EVENT
 VALUE U'nat:page.end'
  /* Page closed.
  IGNORE
 VALUE U'onDisplayMap'
  /* Display a Map.
  FETCH 'MYMAP-P'
 NONE VALUE
  /* Unhandled events.
  PROCESS PAGE UPDATE
END-DECIDE
```

```
*
END
```

Example for program MYMAP-P:

```
DEFINE DATA LOCAL
1 FIELD1 (U20)
1 FIELD2 (U20)
END-DEFINE
*
SET KEY ALL
INPUT USING MAP 'MYMAP'
*
DECIDE ON FIRST *PF-KEY
 VALUE 'PF1'
  /* Display a rich GUI page.
  FETCH 'MYPAGE-P'
 NONE VALUE
  REINPUT WITH TEXT
  'Press PF1 to display rich GUI page.'
END-DECIDE
*
END
```

Using Pages and Maps Alternatively

An application can also decide at runtime whether to use maps or rich GUI pages, depending on the capabilities of the user interface. The system variable *BROWSER-IO lets the application decide if it is running in a web browser at all. If this is the case, the system variable tells whether the application has been started under Natural for Ajax and may thus use both maps and pages, or whether it has been started under the Natural Web I/O Interface and may thus use only maps.

Example:

```
DEFINE DATA LOCAL
1 FIELD1 (U20)
1 FIELD2 (U20)
END-DEFINE
*
IF *BROWSER-IO = 'RICHGUI'
  /* If we are running under Natural for Ajax,
  /* we display a rich GUI page.
  PROCESS PAGE USING 'MYPAGE'
  DECIDE ON FIRST *PAGE-EVENT
    VALUE U'nat:page.end'
```

```
      /* Page closed.
      IGNORE
    NONE VALUE
      /* Unhandled events.
      PROCESS PAGE UPDATE
  END-DECIDE
ELSE
  /* Otherwise we display a map.
  SET KEY ALL
  INPUT USING MAP 'MYMAP'
  DECIDE ON FIRST *PF-KEY
    VALUE 'PF1'
      /* Map closed.
      IGNORE
    NONE VALUE
      REINPUT WITH TEXT
      'Press PF1 to terminate.'
END-DECIDE
END-IF
*
END
```

Starting a Natural Application from the Logon Page

In order to start a Natural application from the logon page, you proceed as decribed below.

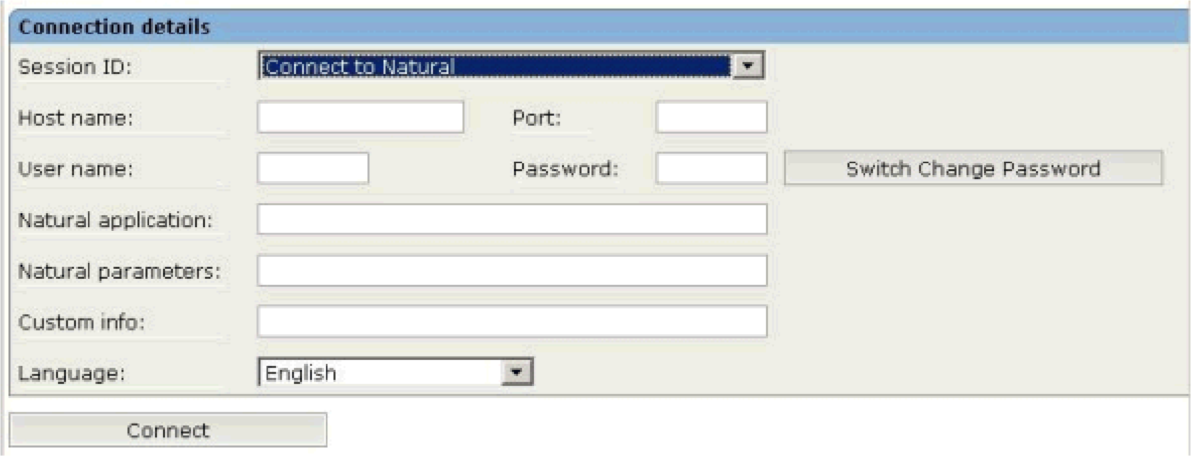
手順 15.3. To start a Natural application from the logon page

1 Enter the following URL inside your browser:

```
http://<host>:<port>/cisnatural/servlet/StartCISPage?PAGEURL=/cisnatural/NatLogon.html
```

where *<host>* and *<port>* are the host name and port number of your application server.

The logon page should now appear.



Connection details

Session ID: Connect to Natural

Host name:

Port:

User name:

Password:

Natural application:

Natural parameters:

Custom info:

Language: English

2 Enter the following connection parameters for the selected session:

Parameter	Description
Host name	The name of the machine on which the Web I/O Interface daemon (Natural for UNIX), the Web I/O Interface service (Natural for Windows), or the Web I/O Interface server (Natural for Mainframes) is running.
Port	The port number on which the Web I/O Interface daemon (Natural for UNIX), the Web I/O Interface service (Natural for Windows), or the Web I/O Interface server (Natural for Mainframes) is running.
User name	Your user ID on the machine on which the Web I/O Interface daemon (Natural for UNIX), the Web I/O Interface service (Natural for Windows), or the Web I/O Interface server (Natural for Mainframes) is running.
Password	Your password on the machine on which the Web I/O Interface daemon (Natural for UNIX), the Web I/O Interface service (Natural for Windows), or the Web I/O Interface server (Natural for Mainframes) is running.
Natural application	UNIX: The name of the script that the Web I/O Interface daemon will use to start a Natural session. See also *Installing the Web I/O Interface Daemon on UNIX* in the *Installation* documentation which is provided for Natural for UNIX. Windows: The name of the batch file that the Web I/O Interface service will use to start a Natural session. See also *Batch File for Starting Natural* in the *Operations* documentation which is provided for Natural for Windows.
Natural parameters	Additional Natural startup parameters that will be passed to the script and the Natural session. You have to specify at least the library and the name of the Natural program that is to be started, for example: `STACK=(LOGON MYPROJ;MYPROG-P;FIN)`
Custom info	An additional parameter that is to be passed to the Natural startup script. It is up to the script to evaluate this parameter or not.

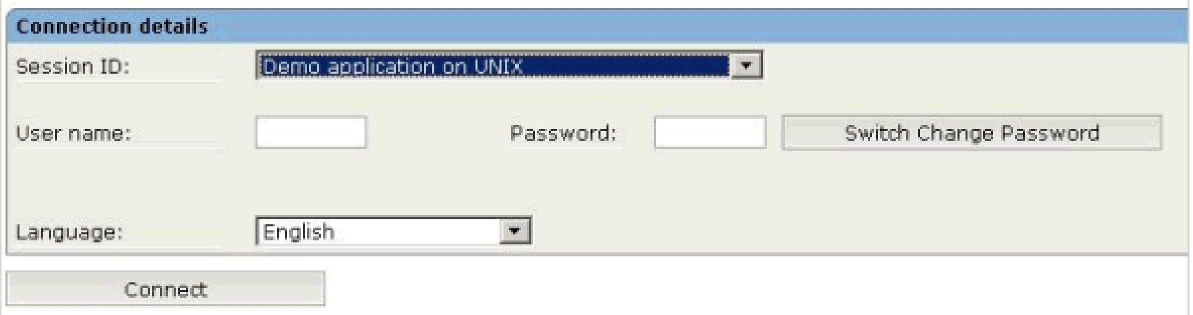
For the convenience of the end users of the application, all of the above parameters can be preconfigured in the configuration file for the session. See also:

- *Defining the Natural Sessions in the Configuration File* in the *Operations* documentation which is provided with Natural for Windows and Natural for UNIX, or
- *Defining the Natural Sessions in the Configuration File* in the *Natural Web I/O Interface Server* documentation which is provided with Natural for Mainframes.

This configuration file is located in the directory `<installdir>`*/WEB-INF*.

For each session definition that has been configured in the configuration file, an entry appears on the logon page. If the user selects the corresponding entry, only those parameters that were not pre-configured in the file need to be specified in the logon page in order to start the application. Usually, you will specify all connection parameters except **User name** and **Password** in the configuration file for the session.

The following shows an example of a session for which the connection parameters have been preconfigured in the configuration file:



Connection details

Session ID: Demo application on UNIX

User name:

Password:

Language: English

An additional parameter that can be specified on the logon page is the language code. For detailed information on multi-lingual applications, see ***Multi Language Management***.

3 Choose the **Connect** button.

Starting a Natural Application with a URL

The connection parameters available in the configuration file for the session (see above) and on the logon page can also be specified as URL parameters of the logon page URL. This allows bookmarking the startup URL of a Natural application or starting an application by clicking a hyperlink in a document.

The following URL parameters are available for the logon page:

URL Parameter	Description
`xciParameters.natsession`	A session ID as defined in the configuration file for the session.
`xciParameters.natserver`	The parameter **Host name** from the logon page.
`xciParameters.natport`	The parameter **Port** from the logon page.
`xciParameters.natuser`	The parameter **User name** from the logon page.
`xciParameters.natprog`	The parameter **Natural application** from the logon page.
`xciParameters.natparam`	The parameter **Natural parameters** from the logon page.

Example

In order to start the Natural program `MENU-NJX` from the library `SYSEXNJX`, while your application server is running on *myappserver:4711*, your Web I/O Interface daemon is running on *mywebio:4712*, and the name of the Natural startup script is *nwo.sh*, you can use the following URL:

```
http://myappserver:4711/cisnatural/servlet/StartCISPage?PAGEURL=%2Fcisnatural%2FNatLogon.html&xciParameters.natserver=mywebio&xciParameters.natprog=nwo.sh&xciParameters.natport=4712&xciParameters.natparam=stack%3D%28logon+SYSEXNJX%3BMENU-NJX%3Bfin%29
```

⚠ **Important:** All parameters must be URL-encoded.

16 Deploying the Application

This chapter covers the following topics:

Components of a Natural for Ajax Application

A Natural for Ajax application consists of two parts that are usually installed on two different machines.

On one hand, there are Natural modules (adapters, programs, subprograms and other Natural objects) that are installed on a Natural server. On the other hand, there are page layouts of rich GUI pages and related files that are installed in a Natural for Ajax environment on an application server.

Unloading Natural Modules

The Natural modules that belong to your application are contained in one or several Natural libraries in your Natural development environment. Unload them into a file, using the Object Handler.

Unloading the User Interface Components

The user interface components of your application are contained in one or several Application Designer projects in your Natural for Ajax development environment on your development application server.

All files in your Application Designer project are stored in one directory on the application server on which Natural for Ajax is installed. The name of the directory corresponds to the project name you have chosen. The location of the directory depends on the application server:

- **JBoss Application Server**
  <installdir>/server/default/deploy/njx<nnn>.ear/cisnatural.war
- **Sun Java System Application Server**
  <installdir>/domains/domain1/applications/j2ee-apps/njx<nnn>.ear/cisnatural.war

where *<installdir>* is the directory in which your application server is installed and *<nnn>* is the current Natural for Ajax version.

The project directory contains a number of subdirectories, only some of which need to be deployed to the production environment. *<projectdir>* in the table below stands for the name of your project directory. Pack the following files and subdirectories into an archive, using an archiving tool like WinZip or tar.

File	Description
<projectdir>/*.html	Generated HTML pages.
<projectdir>/xml/*.*	Page layouts.
<projectdir>/wsdl/*.*	Page data schemas.
<projectdir>/accesspath/*.*	Page data access definitions.
<projectdir>/multilanguage/*.*	Language-dependent strings.
<projectdir>/help/*.*	Language-dependent help texts.

Installing the Natural Modules

In order to install the Natural modules in the production environment, load them with the Object Handler.

Installing the User Interface Components

In order to install the user interface components, unpack the previously created archive into a corresponding project directory in your Natural for Ajax production environment on your production application server.

17 Multi Language Management

The multi language management is responsible for changing the text IDs into strings that are presented to the user.

There are two translation aspects:

- All literals in the GUI definitions of a layout are replaced by strings which are language-specific. This is based on the multi language management of Application Designer.

  Note: Detailed information on the multi language management is provided in the Application Designer documentation at *http://documentation.softwareag.com/crossvision/cit/i18n/multilanguagemanagement.htm*.

- Literals that are contained in your application code are handled with the language management of Natural.

In a Natural for Ajax application, both language management systems are related by common language codes. The language codes used are those that are defined for the Natural profile parameter ULANG and the system variable *LANGUAGE.

The Application Designer documentation describes how the text files containing the language-dependent texts are created and maintained (see the information on writing multi language layouts at the above URL). For a multi-lingual Natural for Ajax application, the names of the directories that contain the text files should be chosen according to the Natural language codes, for instance */multilanguage/4* for Spanish texts.

When an application is started from the Natural logon page, the user can select the language to be used. Depending on the selected language, the same (Natural) language code is set up both in Application Designer and in the Natural session, so that both language management systems are then configured to use the same language.

For compatibility with the predefined multi language directories in Application Designer, the English and German texts need not be stored in */multilanguage/1* and */multilanguage/2*, but can be contained in */multilanguage/en* and */multilanguage/de*.

18 Working with Controls

Controls are the elements that are placed inside containers. This part first gives some common rules that are valid for all controls, then describes the controls in more detail.

The information provided in this part is organized under the following headings:

- NEWSFEED
- RADIOBUTTON
- SCHEDULELINE
- SLIDER
- STRIPSEL
- SUBPAGE
- TABSEL
- TABSTRIP2
- TAGCLOUD
- TEXT
- TEXTOUT
- TOGGLE

Special Controls:

- ACTIVEX
- GOOGLEMAP2
- NETMEETING
- SKYPECALL

19 Some Common Rules for all Controls

This chapter covers the following topics:

Name and Text ID

Every time a control needs a static text definition (the name of a button or the name of a label), there are always two possibilities to define this text:

- Specify a name directly.
- Specify a text ID. This is a literal replaced with a string that is determined inside the multi language management at runtime.

Table, Row, Column, Control

Most controls that allow dynamic sizing offer the following properties:

- `colspan` - number of columns occupied by the control.
- `rowspan` - number of rows occupied by the control.
- `width` - width.
- `height` - height.

These properties influence the way how controls are placed into container rows.

Explicit Alignment

Controls are put into table columns. If the column is wider or higher than the control itself, then you can explicitly control the vertical and horizontal alignment of the control inside the columns.

Most controls offer two properties:

- `valign`
  Specifies the vertical alignment. Valid values are "top", "middle", "bottom". "middle" is the default value.
- `align`
  Specifies the horizontal alignment. Valid values are "left", "center", "right". The default value depends on the control. For example, labels are aligned "left" by default, the default for radio buttons is "center".

Pay attention: `valign` and `align` only affect the position of the control inside the column in which it is positioned if the column is larger than the control. If the column is exactly as wide and high as the control itself, which is the typical case, then they do not have any visual effects - and also need not be defined.

`align`/`valign` do not affect the control's internal alignment.

Binding to Adapter Parameters

Most controls provide properties to specify the binding to the adapter processing. There is a naming convention, which is:

- The names of the properties which specify the binding to an adapter parameter end with "prop".
- The names of the properties which specify the binding to an event end with "method".

The type of the adapter parameter which is referenced by a control depends on the control itself:

- Most controls directly bind to scalar adapter parameters.
- More complex controls bind to an array of group structures.

The type of adapter parameter is described with each control.

Directly Influencing the Control Style

All controls that incorporate textual information - such as labels, buttons or fields - offer the possibility to influence directly the style that is used for displaying the information.

The normal style is derived from the definition inside a cascading style definition file (file *layout.css* inside the *html/general* directory of the server). Overwrite or enhance this style information for your controls by passing the style information inside the corresponding style properties.

The properties specifying the style information end with the suffix "style", e.g. there is a property `labelstyle` for the label tag. The value of the property can be any kind of a valid HTML style specification. If you want to change the display style of a label to be large and blue, define the label in the following way:

```
<label name="Test" width="150" labelstyle="font-size: 24pt; color: #0000FF">
</label>
```

Dynamically Controlling the Visibility and the Display Status of Controls

It is possible to influence the visibility of all input controls (FIELD, BUTTON, etc.) by adapter parameters.

For some of these controls there is a property `visibleprop`, specifying a Boolean adapter parameter. By this, you can control whether you want to display the control within the client or not.

For all other controls - and for more complex manipulations of what is visible and not - use the possibility to be able to control the visibility of rows (ITR, TR) or containers (ROWAREA, ROWTABLE0): these controls provide for a visibility parameter and consequently can be switched on and off.

There is an extended management of what the control status "INVISIBLE" means. Most input controls (FIELD, CHECKBOX, etc.) supporting a `statusprop` or a `visibleprop` also support a property `invisiblemode`. The allowed values of `invisiblemode` are:

- **invisible**
  The corresponding control is completely removed. The horizontal space it occupied before is taken out.
- **cleared**
  The corrresponding control is not visible but still occupies its horizontal space.
- **disabled**
  The corresponding control is displayed with a disabled state. This state is only allowed with a certain number of controls (e.g. button and icon).

Focus Management

Sometimes you want to control the keyboard focus inside a page. Here are the internal rules how a page finds out where to put the focus on.

The default reaction is - if a page is displayed for the first time - to put the focus on the first input control (FIELD, CHECKBOX, RADIOBUTTON, etc.) that is available inside a page. After that, you can navigate through the input controls - and the focus is kept stable when interacting with the server.

With `statusprop` - as mentioned in the previous section - you can interrupt this default reaction; there are two possibilities:

- If an input control is set to status "ERROR", it requests the focus automatically. The purpose is to guide the user automatically to those fields that are not correctly entered.
- If an input control is set to status "FOCUS", it is editable - just as normal - and also requests the focus.

If several input controls are requesting the focus at the same time, the focus is put on the first corresponding input control.

Flushing of Inputs

Most input controls (FIELD, CHECKBOX, RADIOBUTTON, COMBOFIX, etc.) support a property named `flush`. This property controls whether data input from a user causes an immediate synchronisation with the server or whehter data input from a user is stored internally within the client and is synchronized with the next flushing event (e.g. when choosing a button).

There are three different values that can be specified with the `flush` property:

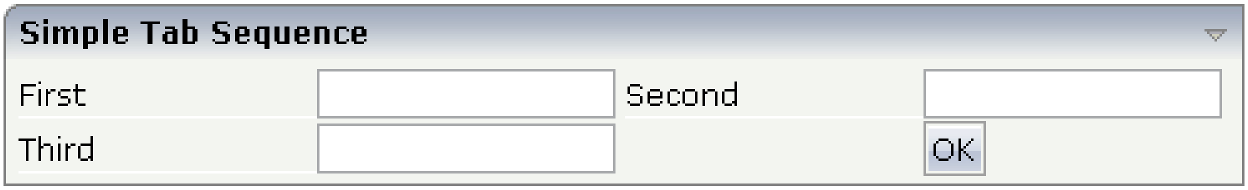
- **""(blank)**
  The data is not synchroized after leaving the control. This is the default.
- **server**
  The data is synchronized with the server immediately when the data has been entered, i.e. when the user has left the corresponding input field.
- **screen**
  The data is synchronized within the controls of the screen. This means - if you have two fields displaying the same property - you can synchronize the fields immediately, without interacting with the server.

Tip: On the one hand, it is useful to flush information in a very fine granular way; you can react on wrong entered data immediately - on the other hand, you have to remember that each flush causes network traffic. The screen's data is sent to the server side processing and the screen waits for the response of the server. During this time, the page is blocked for input and the user sees an hour glass popping up in the left top corner of the screen.

Tab Sequence

By default, the tab sequence of the controls of a page is defined by the order of the controls inside the page's XML layout definition. Using the property `tabindex`, this order can be overridden and the order of the tab index can be explicitly defined.

The following example shows a page with three fields and one button with an explicitly defined tab sequence:



The XML layout definition is:

```
<rowarea name="Simple Tab Sequence">
    <itr takefullwidth="true">
        <coltable0 width="50%">
            <itr>
                <label name="First" width="120">
                </label>
                <field valueprop="first" width="120" tabindex="1">
                </field>
            </itr>
            <itr>
                <label name="Third" width="120">
                </label>
                <field valueprop="third" width="120" tabindex="3">
                </field>
            </itr>
        </coltable0>
        <coltable0 width="50%">
            <itr>
                <label name="Second" width="120">
                </label>
                <field valueprop="second" width="120" tabindex="2">
                </field>
            </itr>
            <itr>
                <hdist width="120">
                </hdist>
                <button name="OK" method="onOK" tabindex="4">
                </button>
            </itr>
        </coltable0>
    </itr>
</rowarea>
```

According to the sequence of controls inside the layout definition, the default tab sequence would be: field **First**, field **Third**, field **Second** and button **OK**.

Due to explicitly defining the `tabindex` property for the fields and the button, the tab sequence is now correct: field **First**, field **Second**, field **Third** and button **OK**.

Pay attention:

- Once having started to explicitly set the tab index in a page, you must consequently continue with all controls of the page. Adding new controls without tab index, is internally interpreted as if these controls were defined with tab index "0".
- Equal tab indices in controls are allowed. In this case, the sequence of the controls inside the layout definition defines the tab sequence among the controls with an equal index.
- Moving controls from one location to the other within a page typically means that you have to adapt the tab sequence accordingly.

The tab index usually is a positive integer value. You may define tab index "-1" for excluding certain controls from the tab sequence at all. In this case, the corresponding controls may only be reached by mouse clicking.

Conclusion:

- In typical pages, you do not have to take care of the tab sequence at all because the default (tab sequence by order of controls in page layout) is adequate to the user's experience.
- Only use the explicit definition of the tab sequence if really it is required - the effort for maintaing each tab index with each control should not be underestimated.

Tooltips

Tooltips can be applied to many controls. If the user hovers with the mouse cursor over a control for some seconds, a small yellow box appears showing some more detailed explanation.

The corresponding controls offer two properties:

- `title`
  Here you can specify a hard-coded text that is used as the tooltip.

- `titletextid`
  Here you specify a text ID that is passed to the multi language management..

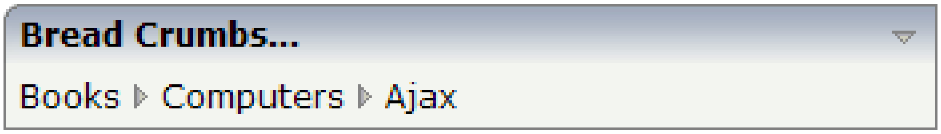
20 BREADCRUMB

The BREADCRUMB control represents a horizontal list of links. The number of links and the name of each link is dynamically controlled by the application.

The control always occupies 100% of the given width.

The following topics are covered below:

Example



Bread Crumbs...

Books ▸ Computers ▸ Ajax

The XML layout definition is:

```
<rowarea name="Bread Crumbs...">
    <breadcrumb breadcrumbprop="items">
    </breadcrumb>
</rowarea>
```

Adapter Interface

```
DEFINE DATA PARAMETER
1 ITEMS (1:*)
2 STYLE (U) DYNAMIC
2 TEXT (U) DYNAMIC
2 TOOLTIP (U) DYNAMIC
1 ITEMSINFO
2 SELECTEDITEM (I4)
END-DEFINE
```

Built-in Events

`value-of-breadcrumbprop`.onSelect

Properties

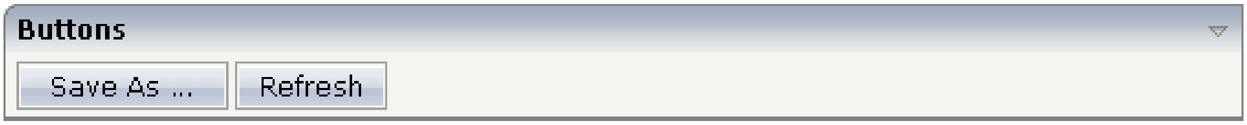
Basic			
breadcrumbprop	Name of the adapter parameter that represents the control in the adapter.	Obligatory	
breadcrumbstyle	CSS style definition that is directly passed into this control. With the style you can individually influence the rendering of the control. You can specify any style sheet expressions. Examples are: border: 1px solid #FF0000 background-color: #808080 You can combine expressions by appending and separating them with a semicolon. Sometimes it is useful to have a look into the generated HTML code in order to know where direct style definitions are applied. Press right mouse-button in your browser and select the "View source" or "View frame's source" function.	Optional	background-color: #FF0000 color: #0000FF font-weight: bold
pixeldistance	Pixel distance between the links that are rendered.	Optional	1 2 3 int-value
comment	Comment without any effect on rendering and behaviour. The comment is shown in the layout editor's tree view.	Optional	

21 BUTTON

The BUTTON control represents a button. Within the definition, specify an event that is sent to the adapter when choosing the button.

The following topics are covered below:

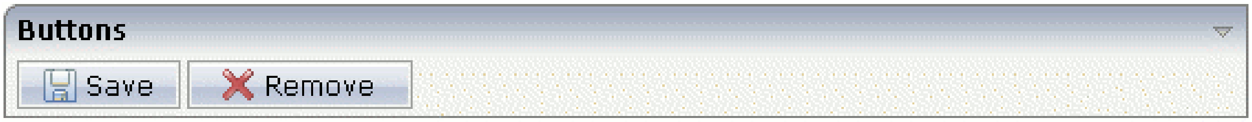
Example: Simple Button



The XML layout definition is:

```
<rowarea name="Buttons">
    <itr>
        <button name="Save As ..." method="saveAs">
        </button>
        <hdist>
        </hdist>
        <button name="Refresh" method="refresh">
        </button>
    </itr>
</rowarea>
```

Example: Button with Image



The XML layout definition is:

```
<rowarea name="Buttons">
    <itr>
       <button name="Save" method="onSave" image="../HTMLBasedGUI/images/save.gif">
        </button>
        <hdist>
        </hdist>
        <button name="Remove" method="onRemove"
image="../HTMLBasedGUI/images/remove.gif">
        </button>
    </itr>
</rowarea>
```

Hiding and Disabling Buttons

Buttons (like many other controls) can be dynamically hidden by using the `visibleprop` property - and referencing to a server side property that decides whether to hide a button or not.

There are two modes of hiding that can be controlled by using the property `invisiblemode`:

- If set to "disabled", the button is grayed and is not selectable anymore.
- If set to "invisible", the button is hidden.

Properties

Basic			
name	Text that is displayed inside the control. Please do not specify the name when using the multi language management - but specify a "textid" instead.	Sometimes obligatory	
textid	Multi language dependent text that is displayed inside the control. The "textid" is translated into a corresponding string at runtime. Do not specify a "name" inside the control if specifying a "textid".	Sometimes obligatory	
method	Name of the event that is sent to the adapter when the user presses the button.	Obligatory	
comment	Comment without any effect on rendering and behaviour. The comment is shown in the layout editor's tree view.	Optional	
Appearance			
name	(already explained above)		
textid	(already explained above)		
image	URL of image that is displayed inside the control. Any image type (.gif, .jpg, ...) that your browser does understand is valid. Use the following options to specify the URL: (A) Define the URL relative to your page. Your page is generated directly into your project's folder. Specifiying "images/xyz.gif" will point into a directory parallel to your page. Specifying "../HTMLBasedGUI/images/new.gif" will point to an image of a neighbour project. (B) Define a complete URL, like "http://www.softwareag.com/images/logo.gif".	Optional	gif jpg jpeg
invisiblemode	This property has three possible values: (1) "invisible": the button is not visible without occupying any space. (2) "disabled": the button is deactivated: it is "grayed" and does not show any roll over effects any more. (3)"cleared": the button is not visible but it still occupies space.	Optional	invisible disabled cleared
width	Width of the control.	Optional	100

	There are three possibilities to define the width: (A) You do not define a width at all. In this case the width of the control will either be a default width or - in case of container controls - it will follow the width that is occupied by its content. (B) Pixel sizing: just input a number value (e.g. "100"). (C) Percentage sizing: input a percantage value (e.g. "50%"). Pay attention: percentage sizing will only bring up correct results if the parent element of the control properly defines a width this control can reference. If you specify this control to have a width of 50% then the parent element (e.g. an ITR-row) may itself define a width of "100%". If the parent element does not specify a width then the rendering result may not represent what you expect.		120 140 160 180 200 50% 100%
height	Height of the control. There are three possibilities to define the height: (A) You do not define a height at all. As consequence the control will be rendered with its default height. If the control is a container control (containing) other controls then the height of the control will follow the height of its content. (B) Pixel sizing: just input a number value (e.g. "20"). (C) Percentage sizing: input a percantage value (e.g. "50%"). Pay attention: percentage sizing will only bring up correct results if the parent element of the control properly defines a height this control can reference. If you specify this control to have a height of 50% then the parent element (e.g. an ITR-row) may itself define a height of "100%". If the parent element does not specify a width then the rendering result may not represent what you expect.	Optional	100 150 200 250 300 250 400 50% 100%
imageheight	Pixel height of image inside button.	Optional	
imagewidth	Pixel width of image inside button.	Optional	
textstyle	CSS style definition that is directly passed into the text of this control. With the style you can individually influence the text of the button. You can specify any style sheet expressions. Examples are: font-weight: bold color: #FF0000	Optional	background-color: #FF0000 color: #0000FF font-weight: bold

buttonstyle	CSS style definition that is directly passed into this control. With the style you can individually influence the rendering of the control. You can specify any style sheet expressions. Examples are: border: 1px solid #FF0000 background-color: #808080 You can combine expressions by appending and separating them with a semicolon. Sometimes it is useful to have a look into the generated HTML code in order to know where direct style definitions are applied. Press right mouse-button in your browser and select the "View source" or "View frame's source" function.	Optional	background-color: #FF0000 color: #0000FF font-weight: bold
stylevariant	Some controls offer the possibility to define style variants. By this style variant you can address different styles inside your style sheet definition file (.css). If not defined "normal" styles are chosen, if defined (e.g. "VAR1") then other style definitions (xxxVAR1xxx) are chosen. Purpose: you can set up style variants in the style sheet defintion and use them multiple times by addressing them via the "stylevariant" property. CIS currently offerst two variants "VAR1" and "VAR2" but does not predefine any semantics behind - this is up to you!	Optional	VAR1 VAR2
align	Horizontal alignment of control in its column. Each control is "packaged" into a column. The column itself is part of a row (e.g. ITR or TR). Sometimes the size of the column is bigger than the size of the control itself. In this case the "align" property specifies the position of the control inside the column. In most cases you do not require the align control to be explicitly defined because the size of the column around the controls exactly is sized in the same way as the contained control. If you want to directly control the alignment of text: in most text based controls there is an explicit property "textalign" in which you align the control's contained text.	Optional	left center right
valign	Vertical alignment of control in its column. Each control is "packaged" into a column. The column itself is part of a row (e.g. ITR or TR). Sometimtes the size of the column is bigger than the size of the control. In this case the "align" property specify the position of the control inside the column.	Optional	top middle bottom
colspan	Column spanning of control.	Optional	1

	If you use TR table rows then you may sometimes want to control the number of columns your control occupies. By default it is "1" - but you may want to define the control to span over more than one columns. The property only makes sense in table rows that are snychronized within one container (i.e. TR, STR table rows). It does not make sense in ITR rows, because these rows are explicitly not synched.		2 3 4 5 50 int-value
rowspan	Row spanning of control. If you use TR table rows then you may sometimes want to control the number of rows your control occupies. By default it is "1" - but you may want to define the control two span over more than one columns. The property only makes sense in table rows that are snychronized within one container (i.e. TR, STR table rows). It does not make sense in ITR rows, because these rows are explicitly not synched.	Optional	1 2 3 4 5 50 int-value
imagedisabled	URL of image that is displayed if the control is disabled. Use properties VISIBLEPROP and INVISIBLEMODE to disable the control.	Optional	gif jpg jpeg
submitbutton	Set this property to true and the button will work as an 'Submitbutton', that is neccessary if you want to transfer and/or save form values. i.e. password and username or complete search forms Default value is false. You should only use a 'Submitbutton' if the withformtag option of the pagebody tag is set true.	Optional	true false
tabindex	Index that defines the tab order of the control. Controls are selected in increasing index order and in source order to resolve duplicates.	Optional	-1 0 1 2 5 10

			32767
Binding			
method	(already explained above)		
visibleprop	Name of the adapter parameter that provides the information if this control is displayed or not. As consequence you can control the visibility of the control dynamically.	Optional	
nameprop	Name of an adapter parameter that provides the text to be displayed inside the button. Typically buttons have static texts either defined by the property "name" or "textid". Via "nameprop" you can dynamically set the button's text by your application. Use the nameprop in cases the button's text should change dependent on your logic. Example: you may want to define the button's text to reflect the next status the user can set to a business object.	Optional	
titleprop	Name of the adapter parameter that dynamically defines the title of the control. The title is displayed as tool tip when ther user moves the mouse onto the control.	Optional	
Online help			
title	Text that is shown as tooltip for the control. Either specify the text "hard" by using this TITLE property - or use the TITLETEXTID in order to define a language dependent literal.	Optional	
titletextid	Text ID that is passed to the multi lanaguage management - representing the tooltip text that is used for the control.	Optional	
titleprop	(already explained above)		
Miscellaneous			
testtoolid	Use this attribute to assign a fixed control identifier that can be later on used within your test tool in order to do the object identification	Optional	

22 BUTTONLIST

The button list represents a vertical arrangement of buttons. The number of buttons and the name on each button are dynamically controlled by the application.

The controls always occupy 100% of the given width and occupy the height required by the buttons.

The following topics are covered below:

Adapter Interface

```
DEFINE DATA PARAMETER
1 BUTTONLIST (1:*)
2 IMAGEURL (U) DYNAMIC
2 METHOD (U) DYNAMIC
2 STYLE (U) DYNAMIC
2 TEXT (U) DYNAMIC
END-DEFINE
```

Properties

Basic			
buttonlistprop	Name of the adapter parameter that represents the control in the adapter.	Obligatory	
pixeldistance	Pixel distance between the buttons that are rendered.	Optional	1 2 3 int-value
buttonstyle	CSS style definition that is directly passed into this control. With the style you can individually influence the rendering of the control. You can specify any style sheet expressions. Examples are: border: 1px solid #FF0000 background-color: #808080 You can combine expressions by appending and separating them with a semicolon.	Optional	background-color: #FF0000 color: #0000FF font-weight: bold

	Sometimes it is useful to have a look into the generated HTML code in order to know where direct style definitions are applied. Press right mouse-button in your browser and select the "View source" or "View frame's source" function.		
imageheight	Pixel height of image inside button.	Optional	
imagewidth	Pixel width of image inside button.	Optional	
tabindex	Index that defines the tab order of the control. Controls are selected in increasing index order and in source order to resolve duplicates.	Optional	-1 0 1 2 5 10 32767
comment	Comment without any effect on rendering and behaviour. The comment is shown in the layout editor's tree view.	Optional	

23 CHECKBOX

The CHECKBOX control displays a check box. It represents a boolean value in the application.

The following topics are covered below:

Properties

Basic			
valueprop	Name of the adapter parameter that represents the control in the adapter.	Obligatory	
comment	Comment without any effect on rendering and behaviour. The comment is shown in the layout editor's tree view.	Optional	
Appearance			
width	Width of the control. There are three possibilities to define the width: (A) You do not define a width at all. In this case the width of the control will either be a default width or - in case of container controls - it will follow the width that is occupied by its content. (B) Pixel sizing: just input a number value (e.g. "100"). (C) Percentage sizing: input a percantage value (e.g. "50%"). Pay attention: percentage sizing will only bring up correct results if the parent element of the control properly defines a width this control can reference. If you specify this control to have a width of 50% then the parent element (e.g. an ITR-row) may itself define a width of "100%". If the parent element does not specify a width then the rendering result may not represent what you expect.	Optional	100 120 140 160 180 200 50% 100%
displayonly	If set to true, the FIELD will not be accessible for input. It is just used as an output field.	Optional	true false
align	Horizontal alignment of control in its column. Each control is "packaged" into a column. The column itself is part of a row (e.g. ITR or TR). Sometimes the size of the column is bigger than the size of the control itself. In this case the "align" property specifies the position of the control inside the column. In most cases you do not require the align control to be explicitly defined because the size of the column around the controls exactly is sized in the same way as the contained control.	Optional	left center right

	If you want to directly control the alignment of text: in most text based controls there is an explicit property "textalign" in which you align the control's contained text.		
valign	Vertical alignment of control in its column. Each control is "packaged" into a column. The column itself is part of a row (e.g. ITR or TR). Sometimtes the size of the column is bigger than the size of the control. In this case the "align" property specify the position of the control inside the column.	Optional	top middle bottom
colspan	Column spanning of control. If you use TR table rows then you may sometimes want to control the number of columns your control occupies. By default it is "1" - but you may want to define the control to span over more than one columns. The property only makes sense in table rows that are snychronized within one container (i.e. TR, STR table rows). It does not make sense in ITR rows, because these rows are explicitly not synched.	Optional	1 2 3 4 5 50 int-value
rowspan	Row spanning of control. If you use TR table rows then you may sometimes want to control the number of rows your control occupies. By default it is "1" - but you may want to define the control two span over more than one columns. The property only makes sense in table rows that are snychronized within one container (i.e. TR, STR table rows). It does not make sense in ITR rows, because these rows are explicitly not synched.	Optional	1 2 3 4 5 50 int-value
invisiblemode	If the visibility of the control is determined dynamically by an adapter property then there are two rendering modes if the visibility is "false": (1) "invisible": the control is not visible. (2) "disabled": the control is deactivated: it is "grayed" and does not show any roll over effects any more.	Optional	invisible cleared
tabindex	Index that defines the tab order of the control. Controls are selected in increasing index order and in source order to resolve duplicates.	Optional	-1 0 1

			2 5 10 32767
Label			
name	Text that is displayed inside the control. Please do not specify the name when using the multi language management - but specify a "textid" instead.	Optional	
textid	Multi language dependent text that is displayed inside the control. The "textid" is translated into a corresponding string at runtime. Do not specify a "name" inside the control if specifying a "textid".	Optional	
hdistpixelwidth	Witdh of the distance between checkbox and label in pixel.	Optional	
labelstyle	CSS style definition that is directly passed into this control. With the style you can individually influence the rendering of the control. You can specify any style sheet expressions. Examples are: border: 1px solid #FF0000 background-color: #808080 You can combine expressions by appending and separating them with a semicolon. Sometimes it is useful to have a look into the generated HTML code in order to know where direct style definitions are applied. Press right mouse-button in your browser and select the "View source" or "View frame's source" function.	Optional	background-color: #FF0000 color: #0000FF font-weight: bold
Binding			
valueprop	(already explained above)		
statusprop	Name of the adapter parameter that dynamically passes information how the control should be rendered and how it should act.	Optional	
flush	Flushing behaviour of the input control. By default an input into the control is registered within the browser client - and communicated to the server adapter object when a user e.g. presses a button. By using the FLUSH property you can change this behaviour.	Optional	screen server

	Setting FLUSH to "server" means that directly after changing the input a synchronization with the server adapter is triggered. As consequence you directly can react inside your adapter logic onto the change of the corresponding value. - Please be aware of that during the synchronization always all changed properties - also the ones that were changed before - are transferred to the adapter object, not only the one that triggered the synchonization. Setting FLUSH to "screen" means that the changed value is populated inside the page. You use this option if you have redundant usage of the same property inside one page and if you want to pass one changed value to all its representaion directly after changing the value.		
flushmethod	$en/popupwizard/njx__attr_flushmethod$	Optional	
Online Help			
helpid	Help id that is passed to the online help management in case the user presses F1 on the control.	Optional	
title	Text that is shown as tooltip for the control. Either specify the text "hard" by using this TITLE property - or use the TITLETEXTID in order to define a language dependent literal.	Optional	
titletextid	Text ID that is passed to the multi lanaguage management - representing the tooltip text that is used for the control.	Optional	
Miscellaneous			
testtoolid	Use this attribute to assign a fixed control identifier that can be later on used within your test tool in order to do the object identification	Optional	

Typically, the CHECKBOX is followed by a LABEL control naming the displayed check box. In the LABEL definition, set the property `asplaintext` to "true".

24 COMBODYN2

The COMBODYN control is the dynamic counterpart of the COMBOFIX control. Whereas the selection options inside the COMBOFIX control are defined in a fixed way inside the page definition, the COMBODYN2 control offers the possibility to control the selection options dynamically in the application.

The following topics are covered below:

Adapter Interface

```
DEFINE DATA PARAMETER
1 COSTCENTER (U) DYNAMIC
1 VALIDCOSTCENTERS (1:*)
2 ID (U) DYNAMIC
2 NAME (U) DYNAMIC
END-DEFINE
```

Properties

Basic			
valueprop	Name of the adapter parameter that provides the content of the control.	Obligatory	
validvaluesprop	Name of the adapter parameter that provides the valid values that are available as selectable options.	Obligatory	
width	Width of the control. There are three possibilities to define the width: (A) You do not define a width at all. In this case the width of the control will either be a default width or - in case of container controls - it will follow the width that is occupied by its content. (B) Pixel sizing: just input a number value (e.g. "100"). (C) Percentage sizing: input a percantage value (e.g. "50%"). Pay attention: percentage sizing will only bring up correct results if the parent element of the control properly defines a width this control can reference. If you specify this control to have a width of 50% then the parent element (e.g. an ITR-row) may itself define a width of "100%". If the parent element does not specify a width then the rendering result may not represent what you expect.	Sometimes obligatory	100 120 140 160 180 200 50% 100%
comment	Comment without any effect on rendering and behaviour. The comment is shown in the layout editor's tree view.	Optional	

Appearance			
width	(already explained above)		
size	Number of rows that are displayed inside the control. If specified as "1" (default) then the control is rendered as combo box - if ">1" then the control is rendered as multi line selection.	Optional	
displayonly	If set to true, the FIELD will not be accessible for input. It is just used as an output field.	Optional	
align	Horizontal alignment of control in its column. Each control is "packaged" into a column. The column itself is part of a row (e.g. ITR or TR). Sometimes the size of the column is bigger than the size of the control itself. In this case the "align" property specifies the position of the control inside the column. In most cases you do not require the align control to be explicitly defined because the size of the column around the controls exactly is sized in the same way as the contained control. If you want to directly control the alignment of text: in most text based controls there is an explicit property "textalign" in which you align the control's contained text.	Optional	left center right
valign	Vertical alignment of control in its column. Each control is "packaged" into a column. The column itself is part of a row (e.g. ITR or TR). Sometimtes the size of the column is bigger than the size of the control. In this case the "align" property specify the position of the control inside the column.	Optional	top middle bottom
colspan	Column spanning of control. If you use TR table rows then you may sometimes want to control the number of columns your control occupies. By default it is "1" - but you may want to define the control to span over more than one columns. The property only makes sense in table rows that are snychronized within one container (i.e. TR, STR table rows). It does not make sense in ITR rows, because these rows are explicitly not synched.	Optional	1 2 3 4 5 50 int-value
rowspan	Row spanning of control. If you use TR table rows then you may sometimes want to control the number of rows your control occupies. By default it is "1" - but you may want to define the control two span over more than one columns.	Optional	1 2 3 4

	The property only makes sense in table rows that are snychronized within one container (i.e. TR, STR table rows). It does not make sense in ITR rows, because these rows are explicitly not synched.		5 50 int-value
renderasfield	If set to "true" then the combo box is rendered like a FIELD control that offers valid value support. Default is "false". The normal translation of COMBODYN2 into HTML renders an HTML-select control. This control has certain limitations inside Internet Explorer: it only offers a very reduced set of styles to manipulate its look and feel and - much worse: it always occupies z-index "0" i.e. if you other areas overlapping the COMBODYN2 area then COMBODYN2 is always on the top. This is quite ugly if e.g. a menu is opened and parts of the menu overlap a COMBODYN2 control.	Optional	true false
combostyle	CSS style definition that is directly passed into this control. With the style you can individually influence the rendering of the control. You can specify any style sheet expressions. Examples are: border: 1px solid #FF0000 background-color: #808080 You can combine expressions by appending and separating them with a semicolon. Sometimes it is useful to have a look into the generated HTML code in order to know where direct style definitions are applied. Press right mouse-button in your browser and select the "View source" or "View frame's source" function.	Optional	
invisiblemode	If the visibility of the control is determined dynamically by an adapter property then there are two rendering modes if the visibility is "false": (1) "invisible": the control is not visible. (2) "disabled": the control is deactivated: it is "grayed" and does not show any roll over effects any more.	Optional	invisible cleared
Binding			
valueprop	(already explained above)		
validvaluesprop	(already explained above)		
statusprop	Name of the adapter parameter that dynamically passes information how the control should be rendered and how it should act.	Optional	

titleprop	Name of the adapter parameter that dynamically defines the title of the control. The title is displayed as tool tip when ther user moves the mouse onto the control.	Optional	
flush	Flushing behaviour of the input control. By default an input into the control is registered within the browser client - and communicated to the server adapter object when a user e.g. presses a button. By using the FLUSH property you can change this behaviour. Setting FLUSH to "server" means that directly after changing the input a synchronization with the server adapter is triggered. As consequence you directly can react inside your adapter logic onto the change of the corresponding value. - Please be aware of that during the synchronization always all changed properties - also the ones that were changed before - are transferred to the adapter object, not only the one that triggered the synchonization. Setting FLUSH to "screen" means that the changed value is populated inside the page. You use this option if you have redundant usage of the same property inside one page and if you want to pass one changed value to all its representaion directly after changing the value.	Optional	screen server
flushmethod	$en/popupwizard/njx__attr_flushmethod$	Optional	
Online Help			
helpid	Help id that is passed to the online help management in case the user presses F1 on the control.	Optional	
titleprop	(already explained above)		
Miscellaneous			
testtoolid	Use this attribute to assign a fixed control identifier that can be later on used within your test tool in order to do the object identification	Optional	

25 COMBOFIX

The COMBOFIX control is a selection control. Depending on its configuration, it is either displayed as a combo box or as a selection list.

The COMBOFIX control allows specifying a defined set of values which can be selected. This set of values is defined as part of the layout definition - it cannot be controlled dynamically by the application.

Note: If you want to use dynamic selection, there are two possibilities. Either use the COMBODYN control which has the same look and feel as the COMBOFIX control, but where the selectable values are not specified as part of the page definition and are controlled by the application. Or use the value help popup dialogs.

The following topics are covered below:

COMBOFIX Properties

Basic			
valueprop	Name of the adapter parameter that provides the content of the control.	Obligatory	
width	Width of the control. There are three possibilities to define the width: (A) You do not define a width at all. In this case the width of the control will either be a default width or - in case of container controls - it will follow the width that is occupied by its content. (B) Pixel sizing: just input a number value (e.g. "100"). (C) Percentage sizing: input a percantage value (e.g. "50%"). Pay attention: percentage sizing will only bring up correct results if the parent element of the control properly defines a width this control can reference. If you specify this control to have a width of 50% then the parent element (e.g. an ITR-row) may itself define a width of "100%". If the parent element does not specify a width then the rendering result may not represent what you expect.	Optional	100 120 140 160 180 200 50% 100%
comment	Comment without any effect on rendering and behaviour. The comment is shown in the layout editor's tree view.	Optional	
Appearance			
width	(already explained above)		
size	Number of rows that are displayed inside the control. If specified as "1" (default) then the control is rendered as combo box - if ">1" then the control is rendered as multi line selection.	Optional	

displayonly	If set to true, the FIELD will not be accessible for input. It is just used as an output field.	Optional	
align	Horizontal alignment of control in its column. Each control is "packaged" into a column. The column itself is part of a row (e.g. ITR or TR). Sometimes the size of the column is bigger than the size of the control itself. In this case the "align" property specifies the position of the control inside the column. In most cases you do not require the align control to be explicitly defined because the size of the column around the controls exactly is sized in the same way as the contained control. If you want to directly control the alignment of text: in most text based controls there is an explicit property "textalign" in which you align the control's contained text.	Optional	left center right
valign	Vertical alignment of control in its column. Each control is "packaged" into a column. The column itself is part of a row (e.g. ITR or TR). Sometimtes the size of the column is bigger than the size of the control. In this case the "align" property specify the position of the control inside the column.	Optional	top middle bottom
colspan	Column spanning of control. If you use TR table rows then you may sometimes want to control the number of columns your control occupies. By default it is "1" - but you may want to define the control to span over more than one columns. The property only makes sense in table rows that are snychronized within one container (i.e. TR, STR table rows). It does not make sense in ITR rows, because these rows are explicitly not synched.	Optional	1 2 3 4 5 50 int-value
rowspan	Row spanning of control. If you use TR table rows then you may sometimes want to control the number of rows your control occupies. By default it is "1" - but you may want to define the control two span over more than one columns. The property only makes sense in table rows that are snychronized within one container (i.e. TR, STR table rows). It does not make sense in ITR rows, because these rows are explicitly not synched.	Optional	1 2 3 4 5 50 int-value
combostyle	CSS style definition that is directly passed into this control.	Optional	

	With the style you can individually influence the rendering of the control. You can specify any style sheet expressions. Examples are: border: 1px solid #FF0000 background-color: #808080 You can combine expressions by appending and separating them with a semicolon. Sometimes it is useful to have a look into the generated HTML code in order to know where direct style definitions are applied. Press right mouse-button in your browser and select the "View source" or "View frame's source" function.		
invisiblemode	If the visibility of the control is determined dynamically by an adapter property then there are two rendering modes if the visibility is "false": (1) "invisible": the control is not visible. (2) "disabled": the control is deactivated: it is "grayed" and does not show any roll over effects any more.	Optional	invisible cleared
tabindex	Index that defines the tab order of the control. Controls are selected in increasing index order and in source order to resolve duplicates.	Optional	-1 0 1 2 5 10 32767
Binding			
valueprop	(already explained above)		
statusprop	Name of the adapter parameter that dynamically passes information how the control should be rendered and how it should act.	Optional	
flush	Flushing behaviour of the input control. By default an input into the control is registered within the browser client - and communicated to the server adapter object when a user e.g. presses a button. By using the FLUSH property you can change this behaviour. Setting FLUSH to "server" means that directly after changing the input a synchronization with the server adapter is triggered. As consequence you directly can react inside your adapter logic onto the change of the corresponding value. - Please be aware of that during the	Optional	screen server

	synchronization always all changed properties - also the ones that were changed before - are transferred to the adapter object, not only the one that triggered the synchonization. Setting FLUSH to "screen" means that the changed value is populated inside the page. You use this option if you have redundant usage of the same property inside one page and if you want to pass one changed value to all its representaion directly after changing the value.		
flushmethod	$en/popupwizard/njx__attr_flushmethod$	Optional	
Online Help			
helpid	Help id that is passed to the online help management in case the user presses F1 on the control.	Optional	
Miscellaneous			
testtoolid	Use this attribute to assign a fixed control identifier that can be later on used within your test tool in order to do the object identification	Optional	

COMBOOPTION Properties

Basic			
name	Name that is displayed as selectable option. Either use the NAME property to specify the text in a "hard" way or use the TEXTID property to define the text in a language dependent way.	Optional	
textid	Text ID that is used for this option. The text id is passed to the mulit language management in order to find a language dependent text.	Optional	
value	Actual value of the option that is passed into the adapter property specified by VALUEPROP inside the COMBOFIX control.	Optional	
comment	Comment without any effect on rendering and behaviour. The comment is shown in the layout editor's tree view.	Optional	

26 DATEINPUT

The DATEINPUT control is used to input a date or a date with time. The input can be done both with the keyboard or by opening a popup in which the user can browse through a calendar. The calendar can be controlled by server side processing in the following way:

- You can define a valid-from and a valid-to date. Thus, the control will not allow the user to input an invalid date.
- You can explicitly control the color and the tooltip information inside the calendar. For example, you may set up a calendar in which vacation times are hightlighted in a certain way.

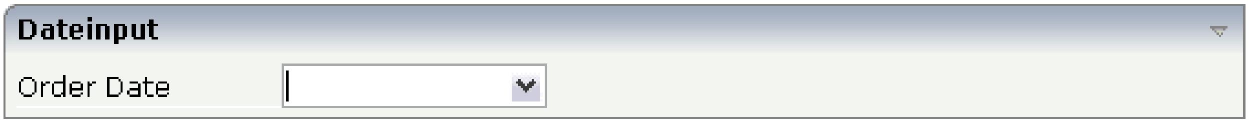
The following topics are covered below:

Example

The most simple usage scenario is to just use the DATEINPUT control in the following way:

```
<rowarea name="Dateinput">
    <itr>
        <label name="Order Date" width="120">
        </label>
        <dateinput valueprop="orderDate" width="120">
        </dateinput>
    </itr>
</rowarea>
```

The corresponding screen looks like this:



Properties

Basic			
valueprop	Name of the adapter parameter that provides the content of the control.	Optional	
width	Width of the control. There are three possibilities to define the width:	Optional	100 120 140

	(A) You do not define a width at all. In this case the width of the control will either be a default width or - in case of container controls - it will follow the width that is occupied by its content. (B) Pixel sizing: just input a number value (e.g. "100"). (C) Percentage sizing: input a percantage value (e.g. "50%"). Pay attention: percentage sizing will only bring up correct results if the parent element of the control properly defines a width this control can reference. If you specify this control to have a width of 50% then the parent element (e.g. an ITR-row) may itself define a width of "100%". If the parent element does not specify a width then the rendering result may not represent what you expect.		160 180 200 50% 100%
flush	Flushing behaviour of the input control. By default an input into the control is registered within the browser client - and communicated to the server adapter object when a user e.g. presses a button. By using the FLUSH property you can change this behaviour. Setting FLUSH to "server" means that directly after changing the input a synchronization with the server adapter is triggered. As consequence you directly can react inside your adapter logic onto the change of the corresponding value. - Please be aware of that during the synchronization always all changed properties - also the ones that were changed before - are transferred to the adapter object, not only the one that triggered the synchonization. Setting FLUSH to "screen" means that the changed value is populated inside the page. You use this option if you have redundant usage of the same property inside one page and if you want to pass one changed value to all its representaion directly after changing the value.	Optional	screen server
datatype	By default, the DATEINPUT control is managing a day. By explicitly setting a datatype you can define that the control is managing a day and time. In the first use type CDATE within your adapter program - in the second case use type CTIMESTAMP.	Optional	date datetime
comment	Comment without any effect on rendering and behaviour. The comment is shown in the layout editor's tree view.	Optional	
Binding			

valueprop	(already explained above)		
fromprop	Name of the adapter parameter that provides a lower limit for the value of the control. The value is used for client side validation of user input.	Optional	
toprop	Name of the adapter parameter that provides an upper limit for the value of the control. The value is used for client side validation of user input.	Optional	
infoprop	Name of the adapter parameter that provides style information that is used inside the date popup.	Optional	
secondsvisprop	Name of the adapter parameter that provides a boolean that indicates if to show additional seconds. This property make sense only if property DATATYPE is set to "daytime".	Optional	
statusprop	Name of the adapter parameter that dynamically passes information how the control should be rendered and how it should act.	Optional	
Appearance			
invisiblemode	If the visibility of the control is determined dynamically by an adapter property then there are two rendering modes if the visibility is "false": (1) "invisible": the control is not visible. (2) "disabled": the control is deactivated: it is "grayed" and does not show any roll over effects any more.	Optional	invisible cleared
displayonly	If set to true, the FIELD will not be accessible for input. It is just used as an output field.	Optional	true false
align	Horizontal alignment of control in its column. Each control is "packaged" into a column. The column itself is part of a row (e.g. ITR or TR). Sometimes the size of the column is bigger than the size of the control itself. In this case the "align" property specifies the position of the control inside the column. In most cases you do not require the align control to be explicitly defined because the size of the column around the controls exactly is sized in the same way as the contained control. If you want to directly control the alignment of text: in most text based controls there is an explicit property "textalign" in which you align the control's contained text.	Optional	left center right
valign	Vertical alignment of control in its column.	Optional	top

	Each control is "packaged" into a column. The column itself is part of a row (e.g. ITR or TR). Sometimtes the size of the column is bigger than the size of the control. In this case the "align" property specify the position of the control inside the column.		middle bottom
inputstyle	CSS style definition that is directly passed into this control. With the style you can individually influence the rendering of the control. You can specify any style sheet expressions. Examples are: border: 1px solid #FF0000 background-color: #808080 You can combine expressions by appending and separating them with a semicolon. Sometimes it is useful to have a look into the generated HTML code in order to know where direct style definitions are applied. Press right mouse-button in your browser and select the "View source" or "View frame's source" function.	Optional	background-color: #FF0000 color: #0000FF font-weight: bold
rowspan	Row spanning of control. If you use TR table rows then you may sometimes want to control the number of rows your control occupies. By default it is "1" - but you may want to define the control two span over more than one columns. The property only makes sense in table rows that are snychronized within one container (i.e. TR, STR table rows). It does not make sense in ITR rows, because these rows are explicitly not synched.	Optional	1 2 3 4 5 50 int-value
colspan	Column spanning of control. If you use TR table rows then you may sometimes want to control the number of columns your control occupies. By default it is "1" - but you may want to define the control to span over more than one columns. The property only makes sense in table rows that are snychronized within one container (i.e. TR, STR table rows). It does not make sense in ITR rows, because these rows are explicitly not synched.	Optional	1 2 3 4 5 50 int-value

noborder	Boolean value defining if the control has a border. Default is "false".	Optional	true false
transparentbackground	Boolean value defining if the control is rendered with a transparent background. Default is "false".	Optional	true false
tabindex	Index that defines the tab order of the control. Controls are selected in increasing index order and in source order to resolve duplicates.	Optional	-1 0 1 2 5 10 32767
Valuehelp			
popupicon	URL of image that is displayed inside the right corner of the field to indicate to the user that there is some value help available.. Any image type (.gif, .jpg, ...) that your browser does understand is valid. Use the following options to specify the URL: (A) Define the URL relative to your page. Your page is generated directly into your project's folder. Specifiying "images/xyz.gif" will point into a directory parallel to your page. Specifying "../HTMLBasedGUI/images/new.gif" will point to an image of a neighbour project. (B) Define a complete URL, like "http://www.softwareag.com/images/logo.gif".	Optional	gif jpg jpeg
popupinputonly	Boolean property that control if a field with POPUPMETHOD defined is still usable for keyboard input. If "false" (= default) then the user can input a value either directly via keyboard or by using the popupmethod's help. If set to "true" then no keyboard input is possible - but only selection from the popup-method's help.	Optional	true false
popuponalt40	Value help in a field is triggered either by clicking with the mouse or by pressing a certain key inside the field. The "traditional" keys are "cusrsor-down", "F7" or "F4". Sometimes you do not want to mix other	Optional	true false

	"cursor-down" behaviour (e.g. scrolling in lists) with the value help behaviour. In this case switch this property to "true" - and the value help will only come up anymore when "alt-cursor-down" is pressed.		
Online Help			
title	Text that is shown as tooltip for the control. Either specify the text "hard" by using this TITLE property - or use the TITLETEXTID in order to define a language dependent literal.	Optional	
titletextid	Text ID that is passed to the multi lanaguage management - representing the tooltip text that is used for the control.	Optional	
titleprop	Name of the adapter parameter that dynamically defines the title of the control. The title is displayed as tool tip when ther user moves the mouse onto the control.	Optional	
helpid	Help id that is passed to the online help management in case the user presses F1 on the control.	Optional	

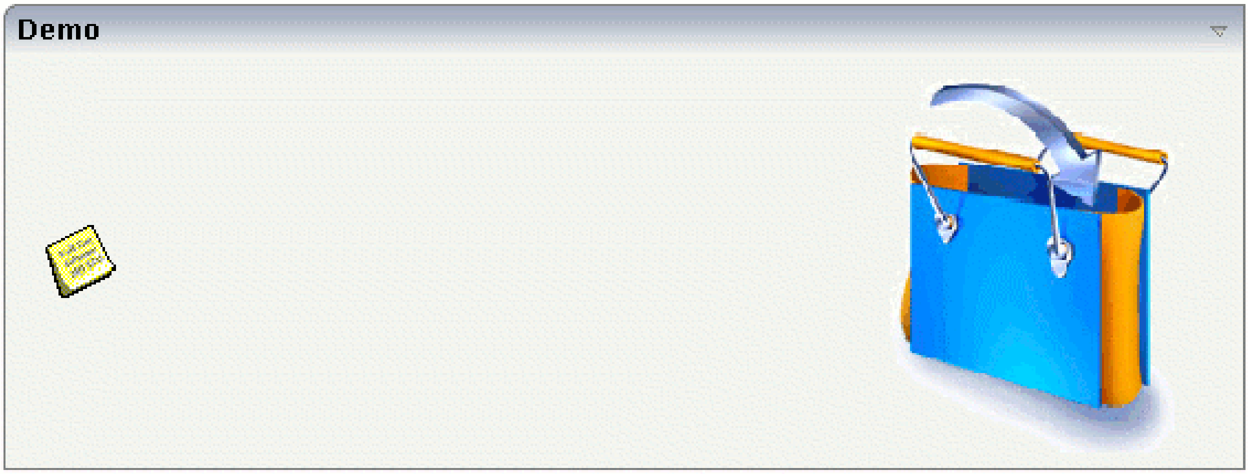
27 DROPICON

The DROPICON control is an icon that can be used in order to build drag-and-drop scenarios. A DROPICON can be defined as the starting point of a drag-and-drop operation or as the target point of a drag-and-drop operation.

The following topics are covered below:

Example

Have a look at the following screen:



The user can click the left mouse button on the left icon (drag), move the mouse to the right icon and then release the mouse button (drop).

The configuration of drag and drop is quite simple: the icon that is used for starting drag-and-drop operations leaves a certain drag information - a plain string. The receiving icon, on which the user performs the drop operation, receives both an event and the string which was left by the icon from where the operation was started.

Properties

Basic			
image	URL that points to the image that is shown as icon. The URL either is an absolute URL or a relative URL. If using a relative URL then be aware of that the generated page is located directly inside your project's directory. Example: "images/icon.gif" points to an icon in an images-folder that is parallel to the page itself. "../HTMLBasedGUI/images/new.gif" point to a URL that is located inside a different project.	Obligatory	gif jpg jpeg
draginfo	String containing any kind of application data to identify the source DROPINFO control within a drag and drop process. Use property DROPINFOPROP to return this data on runtime.	Optional	
draginfoprop	Name of the adapter parameter that provides for information that is passed to the adapter when dropping this control over another DROPICON. Do not use this property (or property DROPINFO respectively) if you do not want the user to drag this control.	Optional	
dropinfoprop	Name of the adapter parameter to that the "drag info" of the dragged DROPICON control is set. Do not use this property if this control should not accept other DROPICON controls within a drag and drop process (i.e. is not a drop target).	Optional	
dropmethod	Name of the event that is sent to the adapter when the user is dragging another DROPICON control over this control and drops it there. Do not use this parameter if this control should not accept other DROPICON controls within a drag and drop process (i.e. is not a drop target).	Sometimes obligatory	
method	Name of the event that is sent to the adapter when clicking on the control.	Sometimes obligatory	
comment	Comment without any effect on rendering and behaviour. The comment is shown in the layout editor's tree view.	Optional	
Binding			
draginfoprop	(already explained above)		
dropinfoprop	(already explained above)		
dropmethod	(already explained above)		
imageprop	Name of adapter parameter that provides as value the URL of the image that is shown inside the control.	Optional	
method	(already explained above)		

visibleprop	Name of the adapter parameter that provides the information if this control is displayed or not. As consequence you can control the visibility of the control dynamically.	Optional	
titleprop	Name of the adapter parameter that dynamically defines the title of the control. The title is displayed as tool tip when ther user moves the mouse onto the control.	Optional	
Appearance			
image	(already explained above)		
invisiblemode	If the visibility of the control is determined dynamically by an adapter property then there are two rendering modes if the visibility is "false": (1) "invisible": the control is not visible. (2) "disabled": the control is deactivated: it is "grayed" and does not show any roll over effects any more.	Optional	invisible cleared
imageinactive	If the visibility is dynamically controlled by using the INVISIBLEPROP then there are two ways the icon reacts if the corresponding property passes back "false". If you want the icon to switch into an inactive status then define inside this property the URL of the image that is the inactive counter part to the normal icon image. Maybe the image is a grayed version of the normal icon image. If you do not define a value for this property then the icon is made invisible.	Optional	
imagewidth	Pixel width of the image that is shown inside the icon. If not defined then the icon is rendered with its normal width.	Optional	
imageheight	Pixel height of the image that is shown inside the icon. If not defined then the icon is rendered with its normal height.	Optional	
withdistance	If set to "true" then 2 pixels of distance are kept on the left and on the right of the icon. Reason behing: if arranging several icons inside one table row (ITR, TR) then a certain distance is kept between the icons when this property is set to "true".	Optional	true false
align	Horizontal alignment of control in its column. Each control is "packaged" into a column. The column itself is part of a row (e.g. ITR or TR). Sometimes the size of the column is bigger than the size of the control itself. In this case the "align" property specifies the position of the control inside the column. In most cases you do not require the align control to be explicitly defined because the size of the	Optional	left center right

	column around the controls exactly is sized in the same way as the contained control. If you want to directly control the alignment of text: in most text based controls there is an explicit property "textalign" in which you align the control's contained text.		
valign	Vertical alignment of control in its column. Each control is "packaged" into a column. The column itself is part of a row (e.g. ITR or TR). Sometimtes the size of the column is bigger than the size of the control. In this case the "align" property specify the position of the control inside the column.	Optional	top middle bottom
colstyle	CSS style definition that is directly passed into this control. With the style you can individually influence the rendering of the control. You can specify any style sheet expressions. Examples are: border: 1px solid #FF0000 background-color: #808080 You can combine expressions by appending and separating them with a semicolon. Sometimes it is useful to have a look into the generated HTML code in order to know where direct style definitions are applied. Press right mouse-button in your browser and select the "View source" or "View frame's source" function.	Optional	background-color: #FF0000 color: #0000FF font-weight: bold
spanstyle	CSS style definition that is directly passed into this control. With the style you can individually influence the rendering of the control. You can specify any style sheet expressions. Examples are: border: 1px solid #FF0000 background-color: #808080 You can combine expressions by appending and separating them with a semicolon. Sometimes it is useful to have a look into the generated HTML code in order to know where direct style definitions are applied. Press right mouse-button in your browser and select the "View source" or "View frame's source" function.	Optional	background-color: #FF0000 color: #0000FF font-weight: bold
tabindex	Index that defines the tab order of the control. Controls are selected in increasing index order and in source order to resolve duplicates.	Optional	-1 0

			1 2 5 10 32767
Online Help			
title	Text that is shown as tooltip for the control. Either specify the text "hard" by using this TITLE property - or use the TITLETEXTID in order to define a language dependent literal.	Optional	
titletextid	Text ID that is passed to the multi lanaguage management - representing the tooltip text that is used for the control.	Optional	
titleprop	(already explained above)		

28 FIELD

The FIELD control is used for entering data. It provides the following features:

- Normal input/output of text.
- Password input.
- Dynamic control if input is allowed.
- Dynamic highlighting of field in case of errors.
- Flush the input directly to the server when leaving the field.
- Raise an event on pressing F4 or F7 or on click - useful for value help popup dialogs
- Adapt the output to a data type (e.g. transfer "YYYYMMDD" to a visible date field)

The following topics are covered below:

Built-in Events

findValidValuesForXXX

Properties

Basic			
valueprop	Name of the adapter parameter that provides the content of the control.	Obligatory	
width	Width of the control. There are three possibilities to define the width: (A) You do not define a width at all. In this case the width of the control will either be a default width or - in case of container controls - it will follow the width that is occupied by its content. (B) Pixel sizing: just input a number value (e.g. "100"). (C) Percentage sizing: input a percantage value (e.g. "50%"). Pay attention: percentage sizing will only bring up correct results if the parent element of the control properly defines a width this control can reference. If you specify this control to have a width of 50% then the parent element (e.g. an ITR-row) may itself define a width of "100%". If the parent element does not	Sometimes obligatory	100 120 140 160 180 200 50% 100%

	specify a width then the rendering result may not represent what you expect.		
comment	Comment without any effect on rendering and behaviour. The comment is shown in the layout editor's tree view.	Optional	
Appearance			
width	(already explained above)		
length	Width of FIELD in amount of characters. WIDTH and LENGTH should not be used together. Note that the actual size of the control depends on the font definition if using the LENGTH property.	Optional	5 10 15 20 int-value
maxlength	Maximum number of characters that a user may enter into this FIELD. This property is not depending on the LENGTH property - please do not get confused by the similar naming. MAXLENGTH has nothing to do with the optical sizing of the control but only with the number of characters you may input.	Optional	5 10 15 20 int-value
textalign	Alignment of text inside the control.	Optional	left center right
password	If set to "true", each entered character is displayed as a '*'.	Optional	true false
displayonly	If set to true, the FIELD will not be accessible for input. It is just used as an output field.	Optional	true false
uppercase	If "true" then all input is automatically transferred to upper case characters.	Optional	true false
align	Horizontal alignment of control in its column. Each control is "packaged" into a column. The column itself is part of a row (e.g. ITR or TR). Sometimes the size of the column is bigger than	Optional	left center right

	the size of the control itself. In this case the "align" property specifies the position of the control inside the column. In most cases you do not require the align control to be explicitly defined because the size of the column around the controls exactly is sized in the same way as the contained control. If you want to directly control the alignment of text: in most text based controls there is an explicit property "textalign" in which you align the control's contained text.		
valign	Vertical alignment of control in its column. Each control is "packaged" into a column. The column itself is part of a row (e.g. ITR or TR). Sometimtes the size of the column is bigger than the size of the control. In this case the "align" property specify the position of the control inside the column.	Optional	top middle bottom
colspan	Column spanning of control. If you use TR table rows then you may sometimes want to control the number of columns your control occupies. By default it is "1" - but you may want to define the control to span over more than one columns. The property only makes sense in table rows that are snychronized within one container (i.e. TR, STR table rows). It does not make sense in ITR rows, because these rows are explicitly not synched.	Optional	1 2 3 4 5 50 int-value
rowspan	Row spanning of control. If you use TR table rows then you may sometimes want to control the number of rows your control occupies. By default it is "1" - but you may want to define the control two span over more than one columns. The property only makes sense in table rows that are snychronized within one container (i.e. TR, STR table rows). It does not make sense in ITR rows, because these rows are explicitly not synched.	Optional	1 2 3 4 5 50 int-value
fieldstyle	CSS style definition that is directly passed into this control.	Optional	background-color: #FF0000 color: #0000FF

	With the style you can individually influence the rendering of the control. You can specify any style sheet expressions. Examples are: border: 1px solid #FF0000 background-color: #808080 You can combine expressions by appending and separating them with a semicolon. Sometimes it is useful to have a look into the generated HTML code in order to know where direct style definitions are applied. Press right mouse-button in your browser and select the "View source" or "View frame's source" function.		font-weight: bold
noborder	Boolean value defining if the control has a border. Default is "false".	Optional	true false
transparentbackground	Boolean value defining if the control is rendered with a transparent background. Default is "false".	Optional	true false
bgcolorprop	Name of the adapter parameter that provides the background color of the control.	Optional	
fgcolorprop	Name of the adapter parameter that passes back a color value (e.g. "#FF0000" for red color). The color value is used as text color in the control. - The background color is automatically chosen dependent from the text color: for light text colors the background color is black, for dark text colors the color is default. Use BGCOLORPROP to choose both - text and background color.	Optional	
invisiblemode	If the visibility of the control is determined dynamically by an adapter property then there are two rendering modes if the visibility is "false": (1) "invisible": the control is not visible. (2) "disabled": the control is deactivated: it is "grayed" and does not show any roll over effects any more.	Optional	invisible cleared
tabindex	Index that defines the tab order of the control. Controls are selected in increasing index order and in source order to resolve duplicates.	Optional	-1 0 1

			2 5 10 32767
Binding			
valueprop	(already explained above)		
flush	Flushing behaviour of the input control. By default an input into the control is registered within the browser client - and communicated to the server adapter object when a user e.g. presses a button. By using the FLUSH property you can change this behaviour. Setting FLUSH to "server" means that directly after changing the input a synchronization with the server adapter is triggered. As consequence you directly can react inside your adapter logic onto the change of the corresponding value. - Please be aware of that during the synchronization always all changed properties - also the ones that were changed before - are transferred to the adapter object, not only the one that triggered the synchonization. Setting FLUSH to "screen" means that the changed value is populated inside the page. You use this option if you have redundant usage of the same property inside one page and if you want to pass one changed value to all its representaion directly after changing the value.	Optional	screen server
flushmethod	$en/popupwizard/njx__attr_flushmethod$	Optional	
statusprop	Name of the adapter parameter that dynamically passes information how the control should be rendered and how it should act.	Optional	
valuetextprop	Name of the adapter parameter that provides a "human understandable" description for the value: in some cases you enter an id into a FIELD but want to display the id and a description to the user. At runtime, the values provided by the VALUEPROP- and the VALUETEXTPROP-property are combined into one text (string) that is returned into the FIELD.	Optional	

textidmode	If using property "valuetextprop" then a field knows an id and a text for a certain value. There are three types of display: either both are shown together, separated by an "-" (e.g. "id - text"). Or only text is shown or only the id is shown. If not defined at all then the system's default text id-mode will be chosen. The default mode can be defined as part of the CIS session context.	Optional	0 1 2
titleprop	Name of the adapter parameter that dynamically defines the title of the control. The title is displayed as tool tip when ther user moves the mouse onto the control.	Optional	
bgcolorprop	(already explained above)		
fgcolorprop	(already explained above)		
autocallpopupmethod	Name of the adapter parameter that controls that the field's value help event is sent to the adapter with a certain offset (milliseconds) after last key down event.	Optional	true false
maxlengthprop	Name of the adapter parameter that provides the maximum number of characters that a user may enter into this FIELD. Consider to use MAXLENGTH to define this number in a static way.	Optional	
Validation			
datatype	By default, the FIELD control is managing its content as string. By explicitly setting a datatype you can define that the control... ...will check the user input if it reflects the datatype. E.g. if the user inputs "abc" into a field with datatype "int" then a corresponding error message will popup when the user leaves the field. ...will format the data coming from the server or coming form the user input: if the field has datatype "date" and the user inputs "010304" then the input will be translated into "01.03.2004" (or other representation, dependent on date format settings). In addition valeu popups are offered for the user automatically for some datatypes: e.g. when specifying datatype "date" the automatically the field provides a calendar input popup. Please note: the datatype "float" is named a bit misleading - it represents any decimal format	Optional	date float int long time timestamp color xs:decimal xs:double xs:date xs:dateTime xs:time ------------------------

	number. The server side representation may be a float value, but also can be a double or a BigDecimal property.		N n.n P n.n string n xs:byte xs:short
validationrules	Contains information used for Data Validation. Use the Validation Rules Editor to make changes!	Optional	
validation	Regular expression against which the content of the field is checked on client side when the user changes the field. If the validation fails then an error message popup up and informs the user about the wrong input.	Optional	[a-zA-Z0-9_.-]{1,}\\@[a-zA-Z0-9_.-]{1,}\\ \\d{5} [0-9)(-/+]+
validationprop	Name of the adapter parameter that provides a regular expression for the validation of the field. Works the same way as VALIDATION but in a dynamic way.	Optional	
validationuserhint	If a client side validation fails due to wrong user input then an error popup is opened. If you define a hint inside this property then the hint is output to the user in order to tell in which way to input the value. The hint is not language dependent.	Optional	
validationuserhintprop	If using validation expressions (either property "validation" or "validationprop") then a popup comes up if the user inputs wrong values into a field. Inside this popup a certain text may be added in order to explain to the user what he/she did not correctly input. This text can be either statically defined or dynamically - by using this property.	Optional	
digits	Number that specifiies how many digits are to be displayed (ie digits before the comma). If using this feature then the DATATYPE property must be set to 'float'. See also DECIMALDIGITS.	Optional	1 2 3 int-value
digitsprop	Name of the adapter parameter that provides information how many digits are to be displayed (i. e. digits before the decimal character). If this	Optional	

	feature is used, the DATATYPE property must be set to 'float'.		
decimaldigits	Number that specifiies how many decimal digits are to be displayed. If using this feature then the DATATYPE property must be set to 'float'.	Optional	1 2 3 int-value
decimaldigitsprop	Name of the adapter parameter that provides information how many decimal digits are to be displayed (i. e. digits before the decimal character). If this feature is used, the DATATYPE property must be set to 'float'.	Optional	
Valuehelp			
popupmethod	Name of the event that is sent to the adapter when the user requests value help by pressing F4 or F7 or by clicking into the FIELD with the right mouse button. See at chapter 'Popup Dialog Management' for more details. If the POPUPMETHOD is defined, a small icon is shown inside the field to indicate to the user that there is some value help available.	Optional	openIdValueCombo openIdValueHelp openIdValueComboOrPopup
popupinputonly	Boolean property that control if a field with POPUPMETHOD defined is still usable for keyboard input. If "false" (= default) then the user can input a value either directly via keyboard or by using the popupmethod's help. If set to "true" then no keyboard input is possible - but only selection from the popup-method's help.	Optional	true false
popupprop	Name of the adapter parameter that provides the information whether a POPUPMETHOD is available or not. This feature is used in scenarios in which a FIELD offers e.g. value help or not, depending on business logic inside the adapter.	Optional	
popuponalt40	Value help in a field is triggered either by clicking with the mouse or by pressing a certain key inside the field. The "traditional" keys are "cusrsor-down", "F7" or "F4". Sometimes you do not want to mix other "cursor-down" behaviour (e.g. scrolling in lists) with the value help behaviour. In this case switch this property to "true" - and the value help will only come up anymore when "alt-cursor-down" is pressed.	Optional	true false

popupcombowidth	Pixel width of the standard "openIdValueCombo" popup dialog. Default is field width or at least 150 pixel.	Optional	1 2 3 int-value
popupicon	URL of image that is displayed inside the right corner of the field to indicate to the user that there is some value help available.. Any image type (.gif, .jpg, ...) that your browser does understand is valid. Use the following options to specify the URL: (A) Define the URL relative to your page. Your page is generated directly into your project's folder. Specifiying "images/xyz.gif" will point into a directory parallel to your page. Specifying "../HTMLBasedGUI/images/new.gif" will point to an image of a neighbour project. (B) Define a complete URL, like "http://www.softwareag.com/images/logo.gif".	Optional	gif jpg jpeg
touchpadinput	Boolean property that decides if touch pad support is offered for the FIELD control. The default is "false". If switched to "true" then you can input data into the field via a touch pad. As consequence you can use this control for making inputs through a touch terminal.	Optional	true false
onlinehelp			
helpid	Help id that is passed to the online help management in case the user presses F1 on the control.	Optional	
title	Text that is shown as tooltip for the control. Either specify the text "hard" by using this TITLE property - or use the TITLETEXTID in order to define a language dependent literal.	Optional	
titletextid	Text ID that is passed to the multi lanaguage management - representing the tooltip text that is used for the control.	Optional	
formula	Contains information used by the Formula Editor. Use the Formula Editor to make changes!	Optional	
Hot Keys			

hotkeys	Comma separated list of hot keys. A hotkey consists of a list of keys and a method name. Separate the keys by "-" and the method name again with a comma Example: ctrl-alt-65;onCtrlAltA;13;onEnter ...defines two hot keys. Method onCtrlAltA is invoked if the user presses Ctrl-Alt-A. Method "onEnter" is called if the user presses the ENTER key. Use the popup help within the Layout Painter to input hot keys.	Optional	
Miscellaneous			
testtoolid	Use this attribute to assign a fixed control identifier that can be later on used within your test tool in order to do the object identification	Optional	

29 FILEUPLOAD/FILEUPLOAD2

The file upload controls simplify the process of uploading files from the client to the server. Two types are available:

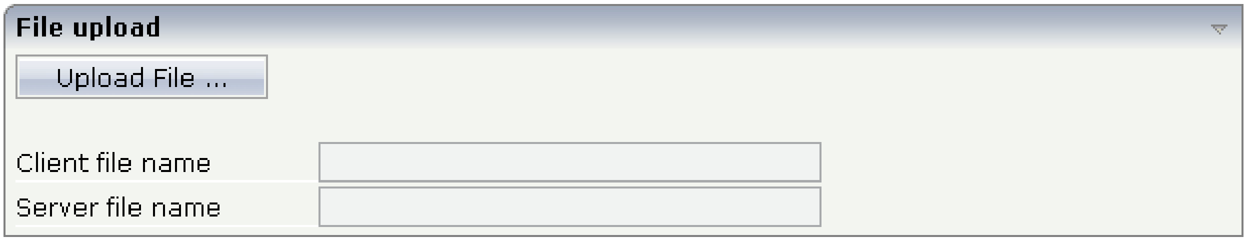
- The FILEUPLOAD control is represented by a button. When you choose the button, a dialog appears showing the file upload form (field input and a file selection button).
- With the FILEUPLOAD2 control, you embed the file upload form into your page.

Both types have the program binding, i.e. you can switch between the two types without touching your code.

The following topics are covered below:

FILEUPLOAD

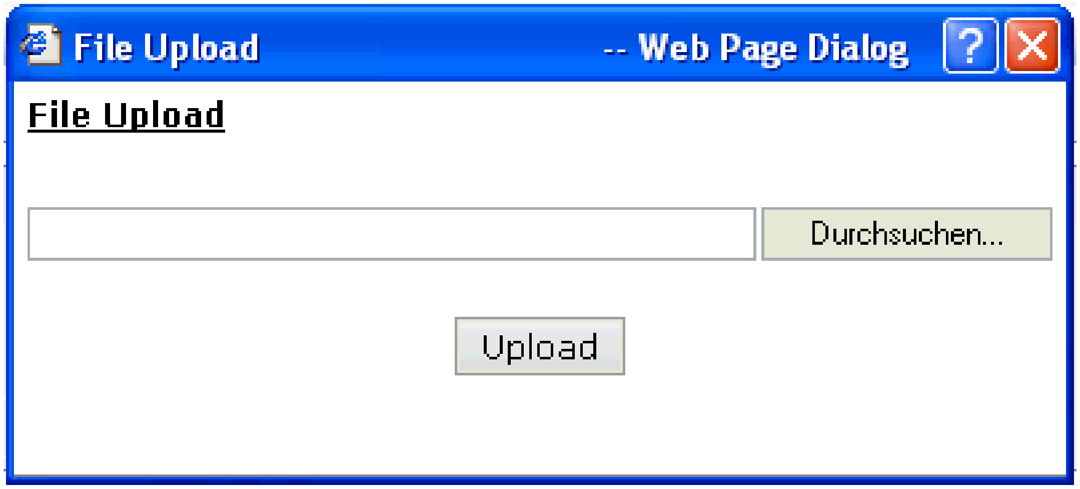
The FILEUPLOAD control simplifies the process of uploading files from the client to the server. Look at the following example:



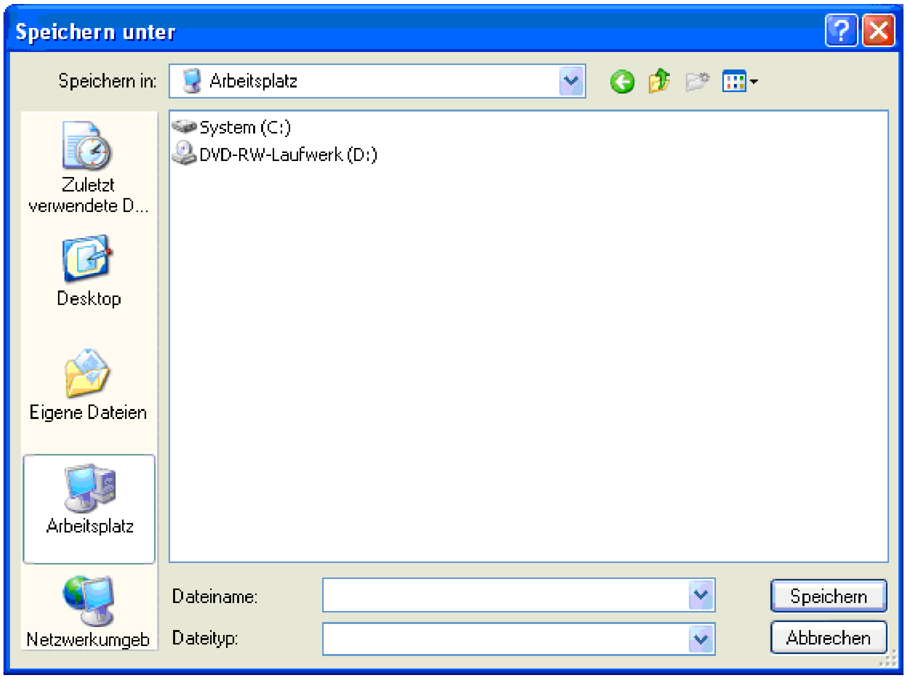
Client file name

Server file name

The control - from the look-and-feel perspective - is a button with some special reaction. When you choose the button, the following dialog appears:

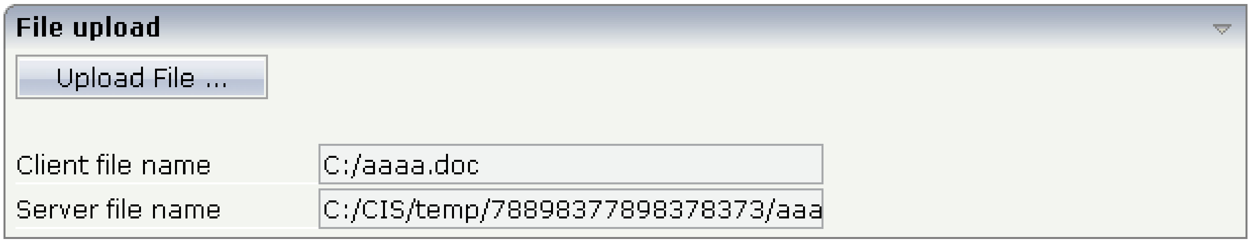


You can either enter a file name or you can invoke the file selection dialog by choosing the button to the right of the field (which appears in the language of the browser).



Speichern in: Arbeitsplatz

After choosing the **Upload** button, the first screen looks as follows:

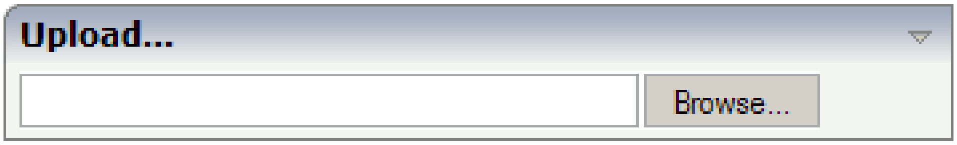


File upload

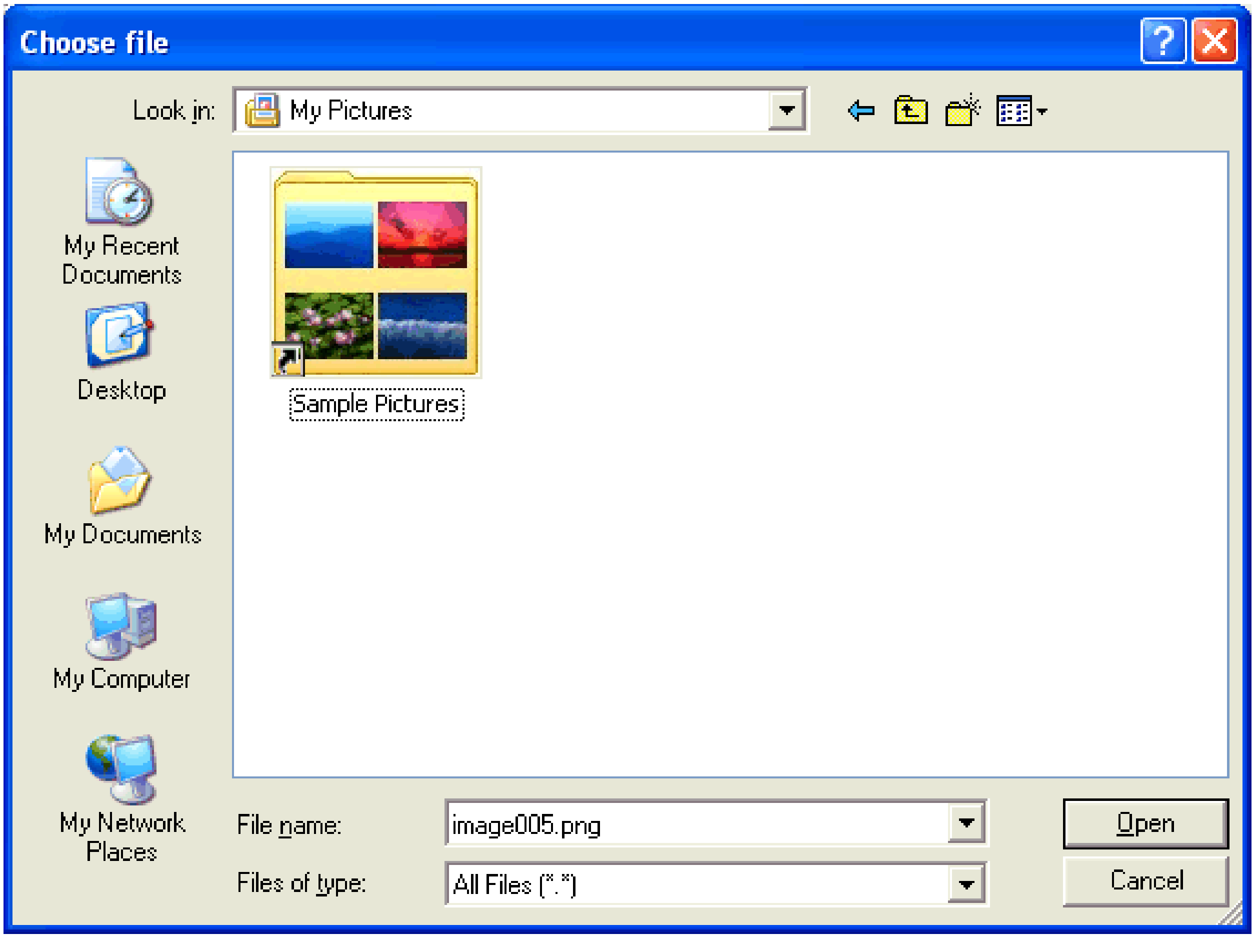
Client file name	C:/aaaa.doc
Server file name	C:/CIS/temp/78898377898378373/aaa

FILEUPLOAD2

With the FILEUPLOAD2 control, you embed the file upload form into your page.



You can either enter a file name or you can invoke the file selection dialog by choosing the button to the right of the field (which appears in the language of the browser).

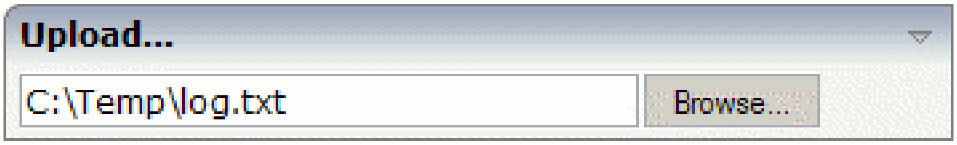


Look in: My Pictures

File name: image005.png

Files of type: All Files (*.*)

After choosing the file, the screen looks as follows:



FILEUPLOAD Properties

Basic			
name	Text that is displayed inside the control. Please do not specify the name when using the multi language management - but specify a "textid" instead.	Sometimes obligatory	
textid	Multi language dependent text that is displayed inside the control. The "textid" is translated into a corresponding string at runtime. Do not specify a "name" inside the control if specifying a "textid".	Sometimes obligatory	
cfileprop	Name of the adapter parameter in which the client file name is passed at upload time.	Obligatory	
sfileprop	Name of the adapter parameter in which at upload time the name of the target file is written, which is a copy of the client file in the server file system. Note that this file name is not the same as the client file name.	Obligatory	
method	Name of the event that is sent to the adapter when a file is uploaded. The file data is available on the server at the point of time this method is called.	Obligatory	
comment	Comment without any effect on rendering and behaviour. The comment is shown in the layout editor's tree view.	Optional	
Appearance			
image	URL of image that is displayed inside the control. Any image type (.gif, .jpg, ...) that your browser does understand is valid. Use the following options to specify the URL: (A) Define the URL relative to your page. Your page is generated directly into your project's folder. Specifiying "images/xyz.gif" will point into a directory parallel to your page. Specifying "../HTMLBasedGUI/images/new.gif" will point to an image of a neighbour project. (B) Define a complete URL, like "http://www.softwareag.com/images/logo.gif".	Optional	gif jpg jpeg
width	Width of the control.	Optional	100

	There are three possibilities to define the width: (A) You do not define a width at all. In this case the width of the control will either be a default width or - in case of container controls - it will follow the width that is occupied by its content. (B) Pixel sizing: just input a number value (e.g. "100"). (C) Percentage sizing: input a percantage value (e.g. "50%"). Pay attention: percentage sizing will only bring up correct results if the parent element of the control properly defines a width this control can reference. If you specify this control to have a width of 50% then the parent element (e.g. an ITR-row) may itself define a width of "100%". If the parent element does not specify a width then the rendering result may not represent what you expect.		120 140 160 180 200 50% 100%
height	Height of the control. There are three possibilities to define the height: (A) You do not define a height at all. As consequence the control will be rendered with its default height. If the control is a container control (containing) other controls then the height of the control will follow the height of its content. (B) Pixel sizing: just input a number value (e.g. "20"). (C) Percentage sizing: input a percantage value (e.g. "50%"). Pay attention: percentage sizing will only bring up correct results if the parent element of the control properly defines a height this control can reference. If you specify this control to have a height of 50% then the parent element (e.g. an ITR-row) may itself define a height of "100%". If the parent element does not specify a width then the rendering result may not represent what you expect.	Optional	100 150 200 250 300 250 400 50% 100%
visibleprop	Name of the adapter parameter that provides the information if this control is displayed or not. As consequence you can control the visibility of the control dynamically.	Optional	
invisiblemode	This property has three possible values: (1) "invisible": the button is not visible without occupying any space. (2) "disabled": the button is deactivated: it is "grayed" and does not show any roll over effects any more. (3)"cleared": the button is not visible but it still occupies space.	Optional	invisible cleared
buttonstyle	CSS style definition that is directly passed into this control. With the style you can individually influence the rendering of the control. You can specify any style sheet expressions. Examples are: border: 1px solid #FF0000	Optional	

	background-color: #808080 You can combine expressions by appending and separating them with a semicolon. Sometimes it is useful to have a look into the generated HTML code in order to know where direct style definitions are applied. Press right mouse-button in your browser and select the "View source" or "View frame's source" function.		
align	Horizontal alignment of control in its column. Each control is "packaged" into a column. The column itself is part of a row (e.g. ITR or TR). Sometimes the size of the column is bigger than the size of the control itself. In this case the "align" property specifies the position of the control inside the column. In most cases you do not require the align control to be explicitly defined because the size of the column around the controls exactly is sized in the same way as the contained control. If you want to directly control the alignment of text: in most text based controls there is an explicit property "textalign" in which you align the control's contained text.	Optional	left center right
valign	Vertical alignment of control in its column. Each control is "packaged" into a column. The column itself is part of a row (e.g. ITR or TR). Sometimtes the size of the column is bigger than the size of the control. In this case the "align" property specify the position of the control inside the column.	Optional	top middle bottom
colspan	Column spanning of control. If you use TR table rows then you may sometimes want to control the number of columns your control occupies. By default it is "1" - but you may want to define the control to span over more than one columns. The property only makes sense in table rows that are snychronized within one container (i.e. TR, STR table rows). It does not make sense in ITR rows, because these rows are explicitly not synched.	Optional	1 2 3 4 5 50 int-value
rowspan	Row spanning of control. If you use TR table rows then you may sometimes want to control the number of rows your control occupies. By default it is "1" - but you may want to define the control two span over more than one columns.	Optional	1 2 3 4

	The property only makes sense in table rows that are snychronized within one container (i.e. TR, STR table rows). It does not make sense in ITR rows, because these rows are explicitly not synched.		5 50 int-value
Binding			
cfileprop	(already explained above)		
sfileprop	(already explained above)		
method	(already explained above)		
visibleprop	(already explained above)		
Online Help			
title	Text that is shown as tooltip for the control. Either specify the text "hard" by using this TITLE property - or use the TITLETEXTID in order to define a language dependent literal.	Optional	
titletextid	Text ID that is passed to the multi lanaguage management - representing the tooltip text that is used for the control.	Optional	

FILEUPLOAD2 Properties

Basic			
width	Width of the control. There are three possibilities to define the width: (A) You do not define a width at all. In this case the width of the control will either be a default width or - in case of container controls - it will follow the width that is occupied by its content. (B) Pixel sizing: just input a number value (e.g. "100"). (C) Percentage sizing: input a percantage value (e.g. "50%"). Pay attention: percentage sizing will only bring up correct results if the parent element of the control properly defines a width this control can reference. If you specify this control to have a width of 50% then the parent element (e.g. an ITR-row) may itself define a width of "100%". If the parent element does not specify a width then the rendering result may not represent what you expect.	Optional	100 120 140 160 180 200 50% 100%
cfileprop	Name of the adapter parameter in which the client file name is passed at upload time.	Optional	
sfileprop	Name of the adapter parameter in which at upload time the name of the target file is written, which is a copy of the client file in the server	Optional	

	file system. Note that this file name is not the same as the client file name.		
method	Name of the event that is sent to the adapter when a file is uploaded. The file data is available on the server at the point of time this method is called.	Optional	
comment	Comment without any effect on rendering and behaviour. The comment is shown in the layout editor's tree view.	Optional	
Binding			
cfileprop	(already explained above)		
sfileprop	(already explained above)		
method	(already explained above)		
visibleprop	Name of the adapter parameter that provides the information if this control is displayed or not. As consequence you can control the visibility of the control dynamically.	Optional	
invisiblemode	If the visibility of the control is determined dynamically by an adapter property then there are two rendering modes if the visibility is "false": (1) "invisible": the control is not visible. (2) "disabled": the control is deactivated: it is "grayed" and does not show any roll over effects any more.	Optional	invisible disabled cleared
Appearance			
invisiblemode	(already explained above)		
rowspan	Row spanning of control. If you use TR table rows then you may sometimes want to control the number of rows your control occupies. By default it is "1" - but you may want to define the control two span over more than one columns. The property only makes sense in table rows that are snychronized within one container (i.e. TR, STR table rows). It does not make sense in ITR rows, because these rows are explicitly not synched.	Optional	1 2 3 4 5 50 int-value
colspan	Column spanning of control. If you use TR table rows then you may sometimes want to control the number of columns your control occupies. By default it is "1" - but you may want to define the control to span over more than one columns. The property only makes sense in table rows that are snychronized within one container (i.e. TR, STR table rows). It does not make sense in ITR rows, because these rows are explicitly not synched.	Optional	1 2 3 4 5 50

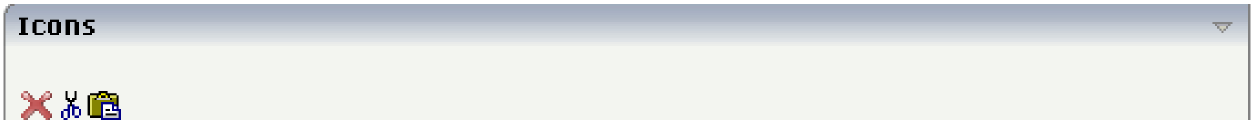
			int-value
darkbackground	Normally the background is in light colour but the CIS style sheets also have a dark(er) grey colour to be used. If DARKBACKGROUND is set to true then the darker background colour is chosen. This property typically is used to integrate light coloured controls into darker container areas.	Optional	true false

30 ICON

The ICON control is similar to the BUTTON control, but it uses an image to display its function. When chosen, it sends an event to the adapter.

The following topics are covered below:

Example



The XML layout definition is:

```
<rowarea name="Icons">
    <itr>
        <icon image="../HTMLBasedGUI/images/remove.gif" method="remove"
title="Remove">
        </icon>
       <icon image="../HTMLBasedGUI/images/cut.gif" method="cut" withdistance="true"
              title="Cut">
        </icon>
       <icon image="../HTMLBasedGUI/images/paste.gif" method="paste" title="Paste">
        </icon>
    </itr>
</rowarea>
```

Properties

Basic			
image	URL that points to the image that is shown as icon. The URL either is an absolute URL or a relative URL. If using a relative URL then be aware of that the generated page is located directly inside your project's directory. Example: "images/icon.gif" points to an icon in an images-folder that is parallel to the page itself. "../HTMLBasedGUI/images/new.gif" point to a URL that is located inside a different project.	Obligatory	gif jpg jpeg

method	Name of the event that is sent to the adapter when clicking on the control.	Obligatory	
name	Text that is displayed inside the control. Please do not specify the name when using the multi language management - but specify a "textid" instead.	Optional	
textid	Multi language dependent text that is displayed inside the control. The "textid" is translated into a corresponding string at runtime. Do not specify a "name" inside the control if specifying a "textid".	Optional	
comment	Comment without any effect on rendering and behaviour. The comment is shown in the layout editor's tree view.	Optional	
Appearance			
imagewidth	Pixel width of the image that is shown inside the icon. If not defined then the icon is rendered with its normal width.	Optional	
imageheight	Pixel height of the image that is shown inside the icon. If not defined then the icon is rendered with its normal height.	Optional	
textsize	The HTML font size of the text. Corresponding to the HTML definition "1" means "smallest" and "6" means "biggest".	Optional	1 2 3 4 5 6
imageinactive	If the visibility is dynamically controlled by using the INVISIBLEPROP then there are two ways the icon reacts if the corresponding property passes back "false". If you want the icon to switch into an inactive status then define inside this property the URL of the image that is the inactive counter part to the normal icon image. Maybe the image is a grayed version of the normal icon image. If you do not define a value for this property then the icon is made invisible.	Optional	gif jpg jpeg
align	Horizontal alignment of control in its column.	Optional	left

	Each control is "packaged" into a column. The column itself is part of a row (e.g. ITR or TR). Sometimes the size of the column is bigger than the size of the control itself. In this case the "align" property specifies the position of the control inside the column. In most cases you do not require the align control to be explicitly defined because the size of the column around the controls exactly is sized in the same way as the contained control. If you want to directly control the alignment of text: in most text based controls there is an explicit property "textalign" in which you align the control's contained text.		center right
valign	Vertical alignment of control in its column. Each control is "packaged" into a column. The column itself is part of a row (e.g. ITR or TR). Sometimtes the size of the column is bigger than the size of the control. In this case the "align" property specify the position of the control inside the column.	Optional	top middle bottom
withdistance	If set to "true" then 2 pixels of distance are kept on the left and on the right of the icon. Reason behing: if arranging several icons inside one table row (ITR, TR) then a certain distance is kept between the icons when this property is set to "true".	Optional	true false
colstyle	CSS style definition that is directly passed into this control. With the style you can individually influence the rendering of the control. You can specify any style sheet expressions. Examples are: border: 1px solid #FF0000 background-color: #808080 You can combine expressions by appending and separating them with a semicolon. Sometimes it is useful to have a look into the generated HTML code in order to know where direct style definitions are applied. Press right mouse-button in your browser and select the "View source" or "View frame's source" function.	Optional	background-color: #FF0000 color: #0000FF font-weight: bold
spanstyle	CSS style definition that is directly passed into this control.	Optional	background-color: #FF0000 color: #0000FF

	With the style you can individually influence the rendering of the control. You can specify any style sheet expressions. Examples are: border: 1px solid #FF0000 background-color: #808080 You can combine expressions by appending and separating them with a semicolon. Sometimes it is useful to have a look into the generated HTML code in order to know where direct style definitions are applied. Press right mouse-button in your browser and select the "View source" or "View frame's source" function.		font-weight: bold
invisiblemode	If the visibility of the control is determined dynamically by an adapter property then there are two rendering modes if the visibility is "false": (1) "invisible": the control is not visible. (2) "disabled": the control is deactivated: it is "grayed" and does not show any roll over effects any more.	Optional	invisible cleared
tabindex	Index that defines the tab order of the control. Controls are selected in increasing index order and in source order to resolve duplicates.	Optional	-1 0 1 2 5 10 32767
nameposition	Position of the (optional) text to the icon. Aside or below, default is aside. Set the corresponding text in the name or the text id property.	Optional	aside below
displaymenuindicator	If set to true a small indicator signals that there is a corresponding menu 'behind this icon'. Default is false.	Optional	true false
Binding			
method	(already explained above)		

visibleprop	Name of the adapter parameter that provides the information if this control is displayed or not. As consequence you can control the visibility of the control dynamically.	Optional	
titleprop	Name of the adapter parameter that dynamically defines the title of the control. The title is displayed as tool tip when ther user moves the mouse onto the control.	Optional	
Online Help			
title	Text that is shown as tooltip for the control. Either specify the text "hard" by using this TITLE property - or use the TITLETEXTID in order to define a language dependent literal.	Optional	
titletextid	Text ID that is passed to the multi lanaguage management - representing the tooltip text that is used for the control.	Optional	
titleprop	(already explained above)		
Miscellaneous			
testtoolid	Use this attribute to assign a fixed control identifier that can be later on used within your test tool in order to do the object identification	Optional	

31 ICONLIST

The ICONLIST is very similar to the BUTTONLIST, representing a list of items instead of a list of buttons. The list can either be a vertical list or a horizontal list.

The following topics are covered below:

Adapter Interface

```
DEFINE DATA PARAMETER
1 ICONLIST (1:*)
2 DRAGINFO (U) DYNAMIC
2 DROPINFO (U) DYNAMIC
2 IMAGEURL (U) DYNAMIC
2 METHOD (U) DYNAMIC
2 NAME (U) DYNAMIC
2 TEXT (U) DYNAMIC
END-DEFINE
```

Built-in Events

value-of-iconlistprop.onDrop
value-of-iconlistprop.onSelect

Properties

Basic			
iconlistprop	Name of the adapter parameter that represents the control in the application.	Obligatory	
vertical	Direction of the icon list. If not specified (or set to "true") then the icons are arranged in one column, one below the other. If specified as "false" then the icons are arrange in one row, one aside the other.	Optional	true false
cellspacing	An icons of the ICONLIST control is embedded into an internal cell. The CELLSPACING property defined the number of pixels that are kept between the icon an the border of this cell. Use the CELLSPACING in order to define a certain distance each icon keeps from the next item.	Optional	1 2 3 int-value

comment	Comment without any effect on rendering and behaviour. The comment is shown in the layout editor's tree view.	Optional	
Appearance			
imagewidth	Pixel width of the image that is shown inside the icon. If not defined then the icon is rendered with its normal width.	Optional	
imageheight	Pixel height of the image that is shown inside the icon. If not defined then the icon is rendered with its normal height.	Optional	
align	Horizontal alignment of control in its column. Each control is "packaged" into a column. The column itself is part of a row (e.g. ITR or TR). Sometimes the size of the column is bigger than the size of the control itself. In this case the "align" property specifies the position of the control inside the column. In most cases you do not require the align control to be explicitly defined because the size of the column around the controls exactly is sized in the same way as the contained control. If you want to directly control the alignment of text: in most text based controls there is an explicit property "textalign" in which you align the control's contained text.	Optional	left center right
tablestyle	Style definition (following CSS style sheet definitions) that is used for the background area of the ICONLIST control.	Optional	background-color: #FF0000 color: #0000FF font-weight: bold
cellstyle	Style definition (following CSS style sheet definitions) that is used for each cell area of the ICONLIST control in which an icon is kept.	Optional	background-color: #FF0000 color: #0000FF font-weight: bold
displaymenuindicator	If set to true a small indicator signals that there is a corresponding menu 'behind this icon'. Default is false.	Optional	true false
additionaltextposition	Position of the text that is displayed inside the control. Use method ICONLISTItem.setName to set the text.	Optional	aside below

textsize	The HTML font size of the text. Corresponding to the HTML definition "1" means "smallest" and "6" means "biggest".	Optional	1 2 3 4 5 6
withrightpadding	Flag (boolean) that indicates whether to insert a padding right hand of the last icon. This attribute does apply for horizontal ICONLIST only (see attribute VERTICAL). Default is true.	Optional	true false

32 IHTML

The IHTML control is used to embed server side generated HTML inside a page that is provided by the application. The IHTML control is very flexible on the one hand. On the other hand, you have to take care about what is defined inside the IHTML area.

Use this control if you have, for example, a server side report generation program already producing HTML as output which you want to include into your pages, etc.

The following topics are covered below:

Properties

Basic			
valueprop	Name of the adapter parameter that provides the content of the control.	Optional	
width	Width of the control. There are three possibilities to define the width: (A) You do not define a width at all. In this case the width of the control will either be a default width or - in case of container controls - it will follow the width that is occupied by its content. (B) Pixel sizing: just input a number value (e.g. "100"). (C) Percentage sizing: input a percantage value (e.g. "50%"). Pay attention: percentage sizing will only bring up correct results if the parent element of the control properly defines a width this control can reference. If you specify this control to have a width of 50% then the parent element (e.g. an ITR-row) may itself define a width of "100%". If the parent element does not specify a width then the rendering result may not represent what you expect.	Optional	100 120 140 160 180 200 50% 100%
height	Height of the control. There are three possibilities to define the height: (A) You do not define a height at all. As consequence the control will be rendered with its default height. If the control is a container control (containing) other controls then the height of the control will follow the height of its content. (B) Pixel sizing: just input a number value (e.g. "20"). (C) Percentage sizing: input a percantage value (e.g. "50%"). Pay attention: percentage sizing will only bring up correct results if the parent element of the control properly defines a height this control can reference. If you specify this control to have a height of 50% then the parent element (e.g. an ITR-row) may itself define	Optional	100 150 200 250 300 250 400 50% 100%

	a height of "100%". If the parent element does not specify a width then the rendering result may not represent what you expect.		
colspan	Column spanning of control. If you use TR table rows then you may sometimes want to control the number of columns your control occupies. By default it is "1" - but you may want to define the control to span over more than one columns. The property only makes sense in table rows that are snychronized within one container (i.e. TR, STR table rows). It does not make sense in ITR rows, because these rows are explicitly not synched.	Optional	1 2 3 4 5 50 int-value
rowspan	Row spanning of control. If you use TR table rows then you may sometimes want to control the number of rows your control occupies. By default it is "1" - but you may want to define the control two span over more than one columns. The property only makes sense in table rows that are snychronized within one container (i.e. TR, STR table rows). It does not make sense in ITR rows, because these rows are explicitly not synched.	Optional	1 2 3 4 5 50 int-value
ihtmlstyle	CSS style definition that is directly passed into this control. With the style you can individually influence the rendering of the control. You can specify any style sheet expressions. Examples are: border: 1px solid #FF0000 background-color: #808080 You can combine expressions by appending and separating them with a semicolon. Sometimes it is useful to have a look into the generated HTML code in order to know where direct style definitions are applied. Press right mouse-button in your browser and select the "View source" or "View frame's source" function.	Optional	background-color: #FF0000 color: #0000FF font-weight: bold
comment	Comment without any effect on rendering and behaviour. The comment is shown in the layout editor's tree view.	Optional	
valign	Vertical alignment of control in its column.	Optional	top middle

	Each control is "packaged" into a column. The column itself is part of a row (e.g. ITR or TR). Sometimtes the size of the column is bigger than the size of the control. In this case the "align" property specify the position of the control inside the column.		bottom

33 IMAGEOUT

The IMAGEOUT control is used to present images inside a page. The name of the image is not statically defined inside the layout but is controlled by the application through an adapter parameter.

The following topics are covered below:

Properties

Basic			
valueprop	Name of the adapter parameter that provides as value the URL of the image that is shown inside the control.	Optional	
width	Width of the control. There are three possibilities to define the width: (A) You do not define a width at all. In this case the width of the control will either be a default width or - in case of container controls - it will follow the width that is occupied by its content. (B) Pixel sizing: just input a number value (e.g. "100"). (C) Percentage sizing: input a percantage value (e.g. "50%"). Pay attention: percentage sizing will only bring up correct results if the parent element of the control properly defines a width this control can reference. If you specify this control to have a width of 50% then the parent element (e.g. an ITR-row) may itself define a width of "100%". If the parent element does not specify a width then the rendering result may not represent what you expect.	Optional	100 120 140 160 180 200 50% 100%
height	Height of the control. There are three possibilities to define the height: (A) You do not define a height at all. As consequence the control will be rendered with its default height. If the control is a container control (containing) other controls then the height of the control will follow the height of its content. (B) Pixel sizing: just input a number value (e.g. "20"). (C) Percentage sizing: input a percantage value (e.g. "50%"). Pay attention: percentage sizing will only bring up correct results if the parent element of the control properly defines a height this control can reference. If you specify this control to have a height of 50% then the parent element (e.g. an ITR-row) may itself define a height of "100%". If the parent element does not specify a width then the rendering result may not represent what you expect.	Optional	
colspan	Column spanning of control.	Optional	

	If you use TR table rows then you may sometimes want to control the number of columns your control occupies. By default it is "1" - but you may want to define the control to span over more than one columns. The property only makes sense in table rows that are snychronized within one container (i.e. TR, STR table rows). It does not make sense in ITR rows, because these rows are explicitly not synched.		
rowspan	Row spanning of control. If you use TR table rows then you may sometimes want to control the number of rows your control occupies. By default it is "1" - but you may want to define the control two span over more than one columns. The property only makes sense in table rows that are snychronized within one container (i.e. TR, STR table rows). It does not make sense in ITR rows, because these rows are explicitly not synched.	Optional	
comment	Comment without any effect on rendering and behaviour. The comment is shown in the layout editor's tree view.	Optional	

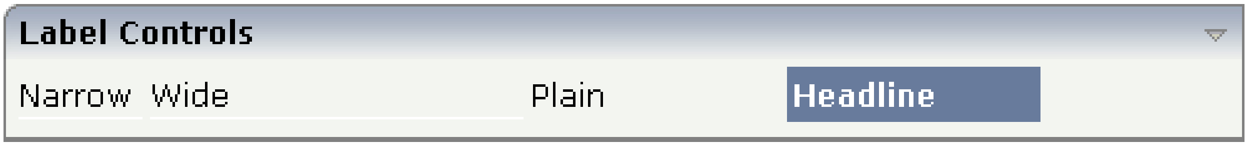
34 LABEL

The LABEL control is a static text. The tag has different properties to control the design of the label. It can be used to display plain text or as a headline of a grid.

By default, the label is rendered with a white line under the text. The default is suitable if a FIELD control follows the label.

The following topics are covered below:

Example



The XML layout definition is:

```
<rowarea name="Label Controls">
    <itr>
        <label name="Narrow" width="50">
        </label>
        <hdist>
        </hdist>
        <label name="Wide" width="150">
        </label>
        <hdist>
        </hdist>
        <label name="Plain" width="100" asplaintext="true">
        </label>
        <hdist>
        </hdist>
        <label name="Headline" width="100" asheadline="true">
        </label>
    </itr>
    <vdist>
    </vdist>
</rowarea>
```

For a better separation between the LABEL controls, horizontal distances (HDIST) were added.

Aligning the Text

Use the property `textalign` in order to align the label's text. Do not use the `align` property. `textalign` refers to the text inside the control, `align` refers to the position of the control inside the surrounding cell - if the cell is larger than the control.

Properties

Basic			
name	Text that is displayed inside the control. Please do not specify the name when using the multi language management - but specify a "textid" instead.	Sometimes obligatory	
textid	Multi language dependent text that is displayed inside the control. The "textid" is translated into a corresponding string at runtime. Do not specify a "name" inside the control if specifying a "textid".	Sometimes obligatory	
width	Width of the control. There are three possibilities to define the width: (A) You do not define a width at all. In this case the width of the control will either be a default width or - in case of container controls - it will follow the width that is occupied by its content. (B) Pixel sizing: just input a number value (e.g. "100"). (C) Percentage sizing: input a percantage value (e.g. "50%"). Pay attention: percentage sizing will only bring up correct results if the parent element of the control properly defines a width this control can reference. If you specify this control to have a width of 50% then the parent element (e.g. an ITR-row) may itself define a width of "100%". If the parent element does not specify a width then the rendering result may not represent what you expect.	Sometimes obligatory	100 120 140 160 180 200 50% 100%
comment	Comment without any effect on rendering and behaviour. The comment is shown in the layout editor's tree view.	Optional	
Appearance			
nowrap	If the textual content of the control exceeds the size of the control then the browser automatically breaks the line and arranges the text accordingly. You can avoid this behaviour by setting NOWRAP to "true". No line break will be performed by the browser.	Optional	true false
width	(already explained above)		
height	Height of the control. There are three possibilities to define the height:	Optional	100 150 200

	(A) You do not define a height at all. As consequence the control will be rendered with its default height. If the control is a container control (containing) other controls then the height of the control will follow the height of its content. (B) Pixel sizing: just input a number value (e.g. "20"). (C) Percentage sizing: input a percantage value (e.g. "50%"). Pay attention: percentage sizing will only bring up correct results if the parent element of the control properly defines a height this control can reference. If you specify this control to have a height of 50% then the parent element (e.g. an ITR-row) may itself define a height of "100%". If the parent element does not specify a width then the rendering result may not represent what you expect.		250 300 250 400 50% 100%
asheadline	If set to true, the label has a dark background and the text is written in white (if using the standard style sheet). You may use this rendering style is you use labels as headlines of control grids (ROWTABLEAREA2 control).	Optional	true false
asplaintext	If set to true, no white line is drawn under the label text (if using the standard style sheet). You may use this rendering style if the label is used to name a RADIOBUTTON control or a CHECKBOX control.	Optional	true false
textalign	Horizontal alignment of the text that is shown.	Optional	left center right
cuttext	Boolean property defining the rendering if the text of the label does not fit into the defined width. If "true" then the text is cut - the part that does not fit is hidden. If "false" then the browser opens a second line. Default is "false".	Optional	true false
labelstyle	CSS style definition that is directly passed into this control. With the style you can individually influence the rendering of the control. You can specify any style sheet expressions. Examples are: border: 1px solid #FF0000 background-color: #808080 You can combine expressions by appending and separating them with a semicolon.	Optional	background-color: #FF0000 color: #0000FF font-weight: bold

	Sometimes it is useful to have a look into the generated HTML code in order to know where direct style definitions are applied. Press right mouse-button in your browser and select the "View source" or "View frame's source" function.		
align	Horizontal alignment of control in its column. Each control is "packaged" into a column. The column itself is part of a row (e.g. ITR or TR). Sometimes the size of the column is bigger than the size of the control itself. In this case the "align" property specifies the position of the control inside the column. In most cases you do not require the align control to be explicitly defined because the size of the column around the controls exactly is sized in the same way as the contained control. If you want to directly control the alignment of text: in most text based controls there is an explicit property "textalign" in which you align the control's contained text.	Optional	left center right
valign	Vertical alignment of control in its column. Each control is "packaged" into a column. The column itself is part of a row (e.g. ITR or TR). Sometimtes the size of the column is bigger than the size of the control. In this case the "align" property specify the position of the control inside the column.	Optional	top middle bottom
colspan	Column spanning of control. If you use TR table rows then you may sometimes want to control the number of columns your control occupies. By default it is "1" - but you may want to define the control to span over more than one columns. The property only makes sense in table rows that are snychronized within one container (i.e. TR, STR table rows). It does not make sense in ITR rows, because these rows are explicitly not synched.	Optional	1 2 3 4 5 50 int-value
rowspan	Row spanning of control. If you use TR table rows then you may sometimes want to control the number of rows your control occupies. By default it is "1" - but you may want to define the control two span over more than one columns. The property only makes sense in table rows that are snychronized within one container (i.e. TR, STR table rows). It does not make sense in ITR rows, because these rows are explicitly not synched.	Optional	1 2 3 4 5 50

			int-value
invisiblemode	If the visibility of the control is determined dynamically by an adapter property then there are two rendering modes if the visibility is "false": (1) "invisible": the control is not visible. (2) "disabled": the control is deactivated: it is "grayed" and does not show any roll over effects any more.	Optional	invisible cleared
Binding			
visibleprop	Name of the adapter parameter that provides the information if this control is displayed or not. As consequence you can control the visibility of the control dynamically.	Optional	
Online Help			
title	Text that is shown as tooltip for the control. Either specify the text "hard" by using this TITLE property - or use the TITLETEXTID in order to define a language dependent literal.	Optional	
titletextid	Text ID that is passed to the multi lanaguage management - representing the tooltip text that is used for the control.	Optional	

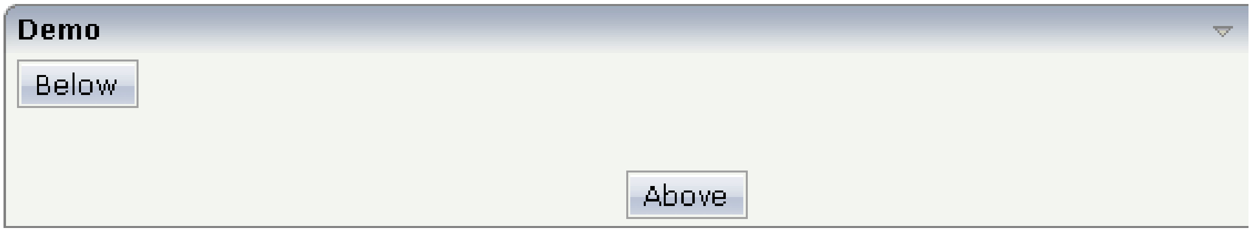
35 MENUBUTTON

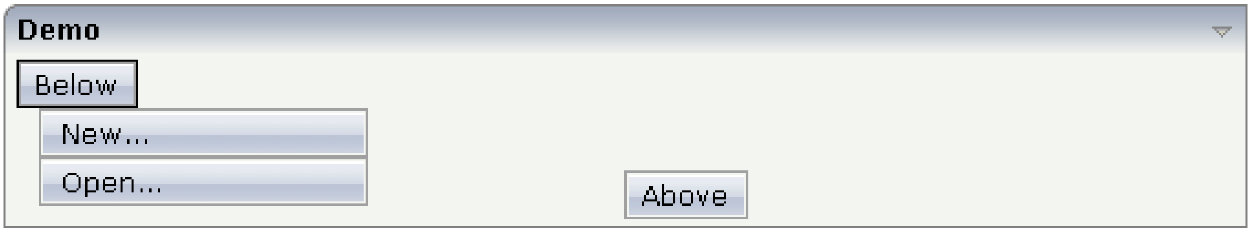
The MENUBUTTON control offers the possibility to arrange buttons in a hierarchy.

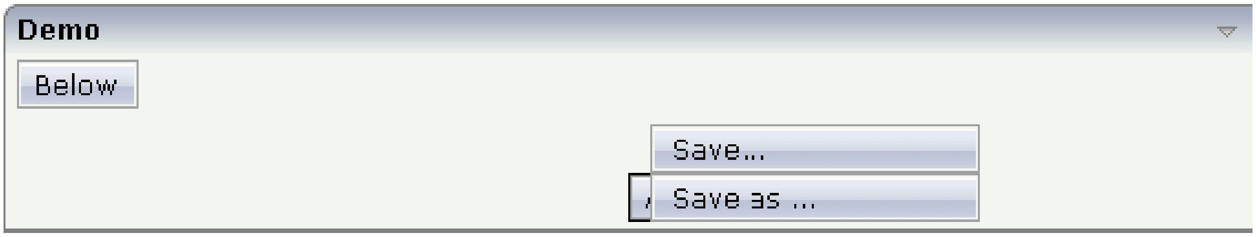
The following topics are covered below:

Example

In the following example, there are two menu buttons which act differently when they are selected:







The XML code for the example looks as follows:

```
<rowarea name="Demo">
    <itr takefullwidth="true">
        <coltable0 width="50%" takefullheight="true">
            <itr>
                <menubutton name="Below" menuposition="below">
                    <menuitem name="New..." method="newFile" pixelwidth="150">
                    </menuitem>
                    <menuitem name="Open..." method="openFile" pixelwidth="150">
                    </menuitem>
                </menubutton>
            </itr>
        </coltable0>
        <coltable0 width="50%">
            <vdist height="50">
            </vdist>
            <itr>
                <menubutton name="Above" menuposition="above">
                    <menuitem name="Save..." method="saveFile" pixelwidth="150">
                    </menuitem>
                 <menuitem name="Save as ..." method="saveAsFile" pixelwidth="150">
                    </menuitem>
                </menubutton>
            </itr>
        </coltable0>
    </itr>
</rowarea>
```

In the definition of a menu item, an event that is to be sent to an adapter is exactly defined like with a normal button.

MENUBUTTON Properties

Basic			
name	Text that is displayed inside the control. Please do not specify the name when using the multi language management - but specify a "textid" instead.	Sometimes obligatory	
textid	Multi language dependent text that is displayed inside the control. The "textid" is translated into a corresponding string at runtime. Do not specify a "name" inside the control if specifying a "textid".	Sometimes obligatory	
comment	Comment without any effect on rendering and behaviour. The comment is shown in the layout editor's tree view.	Optional	
Appearance			
menuposition	above if the menu should popup above the base menu button - below if the menu should popup below the base menu button. The default is below.	Optional	above below
width	Width of the control. There are three possibilities to define the width: (A) You do not define a width at all. In this case the width of the control will either be a default width or - in case of container controls - it will follow the width that is occupied by its content. (B) Pixel sizing: just input a number value (e.g. "100").	Optional	100 120 140 160 180

	(C) Percentage sizing: input a percantage value (e.g. "50%"). Pay attention: percentage sizing will only bring up correct results if the parent element of the control properly defines a width this control can reference. If you specify this control to have a width of 50% then the parent element (e.g. an ITR-row) may itself define a width of "100%". If the parent element does not specify a width then the rendering result may not represent what you expect.		200 50% 100%
visibleprop	Name of the adapter parameter that provides the information if this control is displayed or not. As consequence you can control the visibility of the control dynamically.	Optional	
colspan	Column spanning of control. If you use TR table rows then you may sometimes want to control the number of columns your control occupies. By default it is "1" - but you may want to define the control to span over more than one columns. The property only makes sense in table rows that are snychronized within one container (i.e. TR, STR table rows). It does not make sense in ITR rows, because these rows are explicitly not synched.	Optional	1 2 3 4 5 50 int-value
rowspan	Row spanning of control. If you use TR table rows then you may sometimes want to control the number of rows your control occupies. By default it is "1" - but you may want to define the control two span over more than one columns. The property only makes sense in table rows that are snychronized within one container (i.e. TR, STR table rows). It does not make sense in ITR rows, because these rows are explicitly not synched.	Optional	1 2 3 4 5 50 int-value
buttonstyle	CSS style definition that is directly passed into this control. With the style you can individually influence the rendering of the control. You can specify any style sheet expressions. Examples are: border: 1px solid #FF0000 background-color: #808080 You can combine expressions by appending and separating them with a semicolon. Sometimes it is useful to have a look into the generated HTML code in order to know where direct style definitions are applied. Press	Optional	

	right mouse-button in your browser and select the "View source" or "View frame's source" function.		

MENUITEM Properties

Basic			
name	Text that is displayed inside the control. Please do not specify the name when using the multi language management - but specify a "textid" instead.	Sometimes obligatory	
textid	Multi language dependent text that is displayed inside the control. The "textid" is translated into a corresponding string at runtime. Do not specify a "name" inside the control if specifying a "textid".	Sometimes obligatory	
pixelwidth	Width of the control in pixels.	Obligatory	
comment	Comment without any effect on rendering and behaviour. The comment is shown in the layout editor's tree view.	Optional	
Appearance			
pixelheight	Height of the control in pixels.	Optional	
itemstyle	CSS style definition that is directly passed into this control. With the style you can individually influence the rendering of the control. You can specify any style sheet expressions. Examples are: border: 1px solid #FF0000 background-color: #808080 You can combine expressions by appending and separating them with a semicolon. Sometimes it is useful to have a look into the generated HTML code in order to know where direct style definitions are applied. Press right mouse-button in your browser and select the "View source" or "View frame's source" function.	Optional	

36 METHODLINK

The METHODLINK is a control that renders a text that is dynamically provided by the application through an adapter parameter. The text is rendered as a hyperlink. When clicking on the hyperlink, an event is sent to the adapter. It is used in scenarios in which users are in the habit of following links instead of choosing buttons or icons.

The following topics are covered below:

Properties

Basic			
name	Text that is displayed inside the control. Please do not specify the name when using the multi language management - but specify a "textid" instead.	Optional	
textid	Multi language dependent text that is displayed inside the control. The "textid" is translated into a corresponding string at runtime. Do not specify a "name" inside the control if specifying a "textid".	Optional	
method	Name of the event that is sent to the adapter when clicking on the control.	Obligatory	
valueprop	Name of the adapter parameter that provides the text that is shown as link.	Obligatory	
width	Width of the control. There are three possibilities to define the width: (A) You do not define a width at all. In this case the width of the control will either be a default width or - in case of container controls - it will follow the width that is occupied by its content. (B) Pixel sizing: just input a number value (e.g. "100"). (C) Percentage sizing: input a percantage value (e.g. "50%"). Pay attention: percentage sizing will only bring up correct results if the parent element of the control properly defines a width this control can reference. If you specify this control to have a width of 50% then the parent element (e.g. an ITR-row) may itself define a width of "100%". If the parent element does not specify a width then the rendering result may not represent what you expect.	Sometimes obligatory	100 120 140 160 180 200 50% 100%
comment	Comment without any effect on rendering and behaviour. The comment is shown in the layout editor's tree view.	Optional	
Appearance			

width	(already explained above)		
straighttext	If the text of the control contains HTML tags then these are by default interpreted by the browser. Specifiying STRAIGHTTEXT as "true" means that the browser will directly render the characters without HTML interpretation. Example: if you want to output the source of an HTML text then STRAIGHTTEXT should be set to "true". MOZILLA: this property is not available in Mozilla!	Optional	true false
linkstyle	CSS style definition that is directly passed into this control. With the style you can individually influence the rendering of the control. You can specify any style sheet expressions. Examples are: border: 1px solid #FF0000 background-color: #808080 You can combine expressions by appending and separating them with a semicolon. Sometimes it is useful to have a look into the generated HTML code in order to know where direct style definitions are applied. Press right mouse-button in your browser and select the "View source" or "View frame's source" function.	Optional	background-color: #FF0000 color: #0000FF font-weight: bold
linkclass	CSS style class definition that is directly passed into this control. The style class can be either one which is part of the "normal" CIS style sheet files (i.e. the ones that you maintain with the style sheet editor) - or it can be one of an other style sheet file that you may reference via the ADDSTYLESHEET property of the PAGE tag.	Optional	
align	Horizontal alignment of control in its column. Each control is "packaged" into a column. The column itself is part of a row (e.g. ITR or TR). Sometimes the size of the column is bigger than the size of the control itself. In this case the "align" property specifies the position of the control inside the column. In most cases you do not require the align control to be explicitly defined because the size of the column around the controls exactly is sized in the same way as the contained control. If you want to directly control the alignment of text: in most text based controls there is an explicit property "textalign" in which you align the control's contained text.	Optional	left center right
valign	Vertical alignment of control in its column.	Optional	top

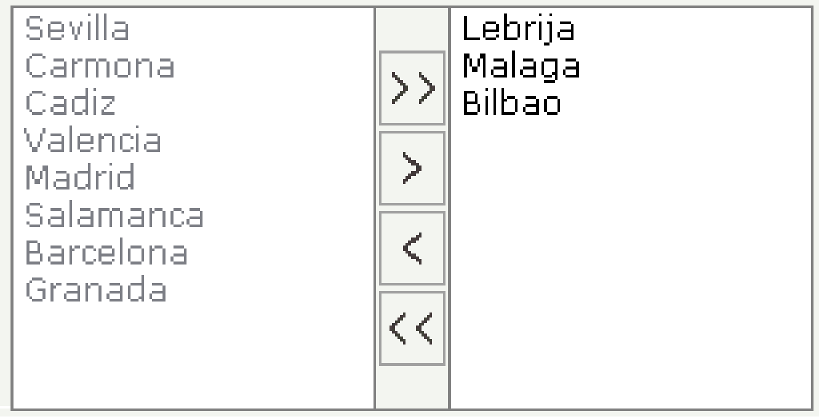
	Each control is "packaged" into a column. The column itself is part of a row (e.g. ITR or TR). Sometimtes the size of the column is bigger than the size of the control. In this case the "align" property specify the position of the control inside the column.		middle bottom
colspan	Column spanning of control. If you use TR table rows then you may sometimes want to control the number of columns your control occupies. By default it is "1" - but you may want to define the control to span over more than one columns. The property only makes sense in table rows that are snychronized within one container (i.e. TR, STR table rows). It does not make sense in ITR rows, because these rows are explicitly not synched.	Optional	1 2 3 4 5 50 int-value
rowspan	Row spanning of control. If you use TR table rows then you may sometimes want to control the number of rows your control occupies. By default it is "1" - but you may want to define the control two span over more than one columns. The property only makes sense in table rows that are snychronized within one container (i.e. TR, STR table rows). It does not make sense in ITR rows, because these rows are explicitly not synched.	Optional	1 2 3 4 5 50 int-value
Binding			
valueprop	(already explained above)		
method	(already explained above)		
titleprop	Name of the adapter parameter that dynamically defines the title of the control. The title is displayed as tool tip when ther user moves the mouse onto the control.	Optional	

37 MULTISELECT

The MULTISELECT control allows comfortable input of multiple selections of items from a defined number of items.

The following topics are covered below:

Example



Sevilla
Carmona
Cadiz
Valencia
Madrid
Salamanca
Barcelona
Granada

Lebrija
Malaga
Bilbao

The available items are rendered on the left and are brought to the right by choosing the corresponding button. There are buttons to bring all items from the left to the right, and back.

Adapter Interface

```
DEFINE DATA PARAMETER
1 TOWNS (1:*)
2 ID (U) DYNAMIC
2 SELECTED (L)
```

```
2 TEXT (U) DYNAMIC
END-DEFINE
```

Properties

Basic			
valueprop	Name of the adapter parameter representing this control in the application.	Obligatory	
width	Width of the control. There are three possibilities to define the width: (A) You do not define a width at all. In this case the width of the control will either be a default width or - in case of container controls - it will follow the width that is occupied by its content. (B) Pixel sizing: just input a number value (e.g. "100"). (C) Percentage sizing: input a percantage value (e.g. "50%"). Pay attention: percentage sizing will only bring up correct results if the parent element of the control properly defines a width this control can reference. If you specify this control to have a width of 50% then the parent element (e.g. an ITR-row) may itself define a width of "100%". If the parent element does not specify a width then the rendering result may not represent what you expect.	Obligatory	100 120 140 160 180 200 50% 100%
height	Height of the control. There are three possibilities to define the height: (A) You do not define a height at all. As consequence the control will be rendered with its default height. If the control is a container control (containing) other controls then the height of the control will follow the height of its content. (B) Pixel sizing: just input a number value (e.g. "20"). (C) Percentage sizing: input a percantage value (e.g. "50%"). Pay attention: percentage sizing will only bring up correct results if the parent element of the control properly defines a height this control can reference. If you specify this control to have a height of 50% then the parent element (e.g. an ITR-row) may itself define a height of "100%". If the parent element does not specify a width then the rendering result may not represent what you expect.	Obligatory	100 150 200 250 300 250 400 50% 100%
comment	Comment without any effect on rendering and behaviour. The comment is shown in the layout editor's tree view.	Optional	
Appearance			

displayonly	If set to true, the FIELD will not be accessible for input. It is just used as an output field.	Optional	true false
align	Horizontal alignment of control in its column. Each control is "packaged" into a column. The column itself is part of a row (e.g. ITR or TR). Sometimes the size of the column is bigger than the size of the control itself. In this case the "align" property specifies the position of the control inside the column. In most cases you do not require the align control to be explicitly defined because the size of the column around the controls exactly is sized in the same way as the contained control. If you want to directly control the alignment of text: in most text based controls there is an explicit property "textalign" in which you align the control's contained text.	Optional	left center right
valign	Vertical alignment of control in its column. Each control is "packaged" into a column. The column itself is part of a row (e.g. ITR or TR). Sometimtes the size of the column is bigger than the size of the control. In this case the "align" property specify the position of the control inside the column.	Optional	top middle bottom
colspan	Column spanning of control. If you use TR table rows then you may sometimes want to control the number of columns your control occupies. By default it is "1" - but you may want to define the control to span over more than one columns. The property only makes sense in table rows that are snychronized within one container (i.e. TR, STR table rows). It does not make sense in ITR rows, because these rows are explicitly not synched.	Optional	1 2 3 4 5 50 int-value
rowspan	Row spanning of control. If you use TR table rows then you may sometimes want to control the number of rows your control occupies. By default it is "1" - but you may want to define the control two span over more than one columns. The property only makes sense in table rows that are snychronized within one container (i.e. TR, STR table rows). It does not make sense in ITR rows, because these rows are explicitly not synched.	Optional	1 2 3 4 5 50 int-value
msstyle	CSS style definition that is directly passed into this control.	Optional	

	With the style you can individually influence the rendering of the control. You can specify any style sheet expressions. Examples are: border: 1px solid #FF0000 background-color: #808080 You can combine expressions by appending and separating them with a semicolon. Sometimes it is useful to have a look into the generated HTML code in order to know where direct style definitions are applied. Press right mouse-button in your browser and select the "View source" or "View frame's source" function.		
Binding			
valueprop	(already explained above)		
flush	Flushing behaviour of the input control. By default an input into the control is registered within the browser client - and communicated to the server adapter object when a user e.g. presses a button. By using the FLUSH property you can change this behaviour. Setting FLUSH to "server" means that directly after changing the input a synchronization with the server adapter is triggered. As consequence you directly can react inside your adapter logic onto the change of the corresponding value. - Please be aware of that during the synchronization always all changed properties - also the ones that were changed before - are transferred to the adapter object, not only the one that triggered the synchonization. Setting FLUSH to "screen" means that the changed value is populated inside the page. You use this option if you have redundant usage of the same property inside one page and if you want to pass one changed value to all its representaion directly after changing the value.	Optional	screen server
Online Help			
helpid	Help id that is passed to the online help management in case the user presses F1 on the control.	Optional	

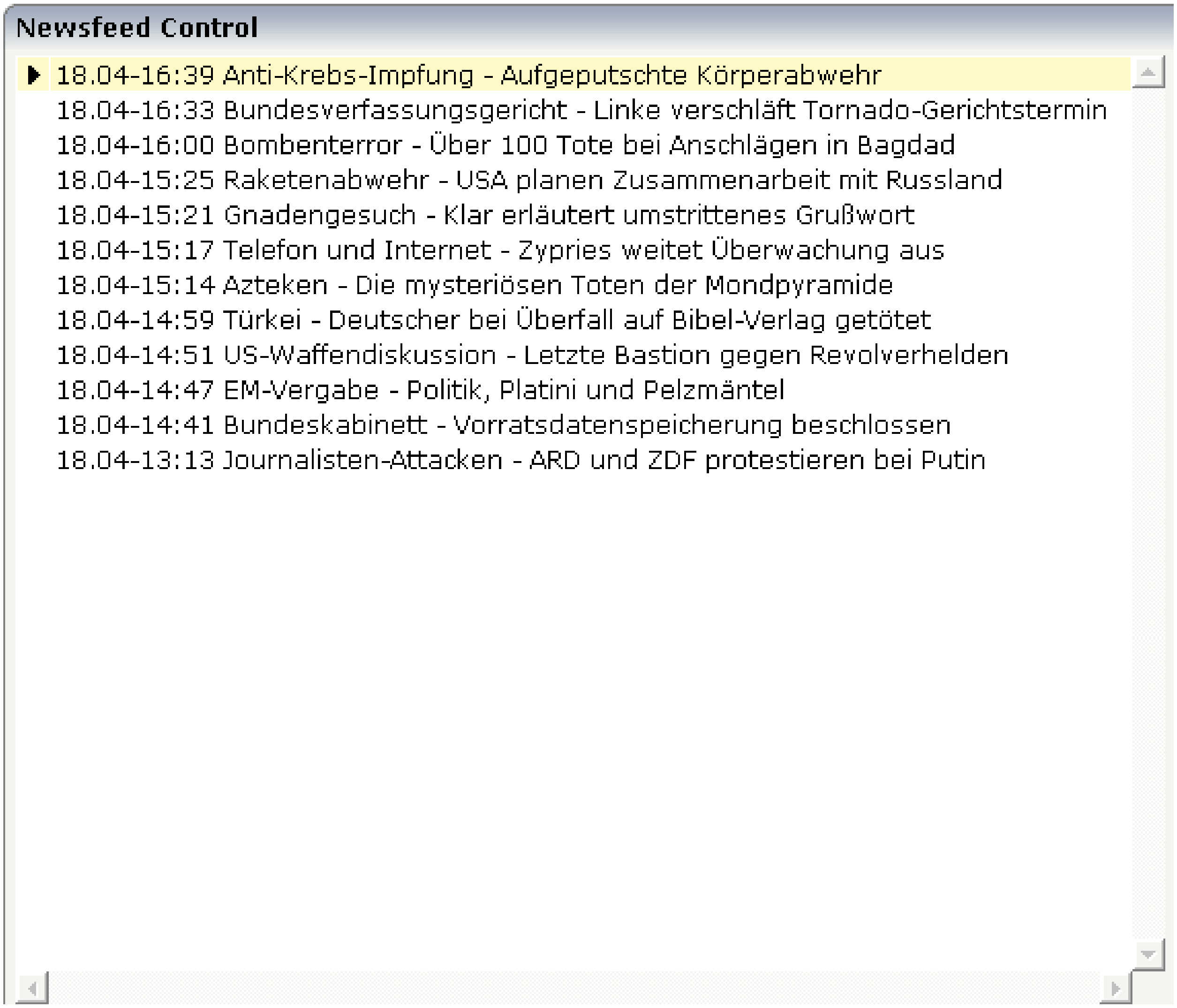
38 NEWSFEED

The NEWSFEED control is a simple-to-use 「newsreader」 within the Application Designer pages. It offers the possibility to read news feeds (RSS feeds and Atom feeds).

⚠ **Important:** In order to use the NEWSFEED control, you have to specify a valid RSS or Atom feed URL (for example ***http://www.news.com/2547-1001_3-0-5.xml***). If necessary, you also have to specify your proxy server settings (host, port, user name, password).

The following topics are covered below:

Example



Newsfeed Control

18.04-16:39 Anti-Krebs-Impfung - Aufgeputschte Körperabwehr
18.04-16:33 Bundesverfassungsgericht - Linke verschläft Tornado-Gerichtstermin
18.04-16:00 Bombenterror - Über 100 Tote bei Anschlägen in Bagdad
18.04-15:25 Raketenabwehr - USA planen Zusammenarbeit mit Russland
18.04-15:21 Gnadengesuch - Klar erläutert umstrittenes Grußwort
18.04-15:17 Telefon und Internet - Zypries weitet Überwachung aus
18.04-15:14 Azteken - Die mysteriösen Toten der Mondpyramide
18.04-14:59 Türkei - Deutscher bei Überfall auf Bibel-Verlag getötet
18.04-14:51 US-Waffendiskussion - Letzte Bastion gegen Revolverhelden
18.04-14:47 EM-Vergabe - Politik, Platini und Pelzmäntel
18.04-14:41 Bundeskabinett - Vorratsdatenspeicherung beschlossen
18.04-13:13 Journalisten-Attacken - ARD und ZDF protestieren bei Putin

The XML layout definition is:

```
<rowarea name="Newsfeed Control" width="560">
  <newsfeed infoprop="newsfeedinfoprop" width="550" height="450">
  </newsfeed>
</rowarea>
```

Built-in Events

value-of-infoprop.onOpenLink
value-of-infoprop.onOpenLinkNewTarget

Properties

Basic			
infoprop	Name of the adapter parameter that represents the control in the adapter.	Obligatory	
height	Height of the control. There are three possibilities to define the height: (A) You do not define a height at all. As consequence the control will be rendered with its default height. If the control is a container control (containing) other controls then the height of the control will follow the height of its content. (B) Pixel sizing: just input a number value (e.g. "20"). (C) Percentage sizing: input a percantage value (e.g. "50%"). Pay attention: percentage sizing will only bring up correct results if the parent element of the control properly defines a height this control can reference. If you specify this control to have a height of 50% then the parent element (e.g. an ITR-row) may itself define a height of "100%". If the parent element does not specify a width then the rendering result may not represent what you expect.	Optional	100 150 200 250 300 250 400 50% 100%
splitstyle	By default the newsfeed control appears within a vsplit control. Headers on the left and content on the right. Set this value to hsplit and the control appears within a hsplit control. Headers on top, content on the bottom.	Optional	vsplit hsplit
comment	Comment without any effect on rendering and behaviour. The comment is shown in the layout editor's tree view.	Optional	

39 RADIOBUTTON

The RADIOBUTTON control displays the radio button. Radio buttons can be grouped together so that a group of RADIOBUTTON controls manipulates one adapter parameter. Each RADIOBUTTON instance represents one value for the adapter parameter.

The following topics are covered below:

Properties

Basic			
valueprop	Name of the adapter parameter that provides the content of the control.	Obligatory	
value	Value that represents this instance of the RADIOBUTTON control. The value is set into the adapter property that is defined by the VALUEPROP property when the user clicks onto the control. - Vice versa: the control is switched to "marked" when the adapter property holds the value defined.	Optional	
comment	Comment without any effect on rendering and behaviour. The comment is shown in the layout editor's tree view.	Optional	
Appearance			
width	Width of the control. There are three possibilities to define the width: (A) You do not define a width at all. In this case the width of the control will either be a default width or - in case of container controls - it will follow the width that is occupied by its content. (B) Pixel sizing: just input a number value (e.g. "100"). (C) Percentage sizing: input a percantage value (e.g. "50%"). Pay attention: percentage sizing will only bring up correct results if the parent element of the control properly defines a width this control can reference. If you specify this control to have a width of 50% then the parent element (e.g. an ITR-row) may itself define a width of "100%". If the parent element does not specify a width then the rendering result may not represent what you expect.	Optional	100 120 140 160 180 200 50% 100%
displayonly	If set to true, the FIELD will not be accessible for input. It is just used as an output field.	Optional	true false

align	Horizontal alignment of control in its column. Each control is "packaged" into a column. The column itself is part of a row (e.g. ITR or TR). Sometimes the size of the column is bigger than the size of the control itself. In this case the "align" property specifies the position of the control inside the column. In most cases you do not require the align control to be explicitly defined because the size of the column around the controls exactly is sized in the same way as the contained control. If you want to directly control the alignment of text: in most text based controls there is an explicit property "textalign" in which you align the control's contained text.	Optional	left center right
valign	Vertical alignment of control in its column. Each control is "packaged" into a column. The column itself is part of a row (e.g. ITR or TR). Sometimtes the size of the column is bigger than the size of the control. In this case the "align" property specify the position of the control inside the column.	Optional	top middle bottom
colspan	Column spanning of control. If you use TR table rows then you may sometimes want to control the number of columns your control occupies. By default it is "1" - but you may want to define the control to span over more than one columns. The property only makes sense in table rows that are snychronized within one container (i.e. TR, STR table rows). It does not make sense in ITR rows, because these rows are explicitly not synched.	Optional	1 2 3 4 5 50 int-value
rowspan	Row spanning of control. If you use TR table rows then you may sometimes want to control the number of rows your control occupies. By default it is "1" - but you may want to define the control two span over more than one columns. The property only makes sense in table rows that are snychronized within one container (i.e. TR, STR table rows). It does not make sense in ITR rows, because these rows are explicitly not synched.	Optional	1 2 3 4 5 50 int-value
invisiblemode	If the visibility of the control is determined dynamically by an adapter property then there are two rendering modes if the visibility is "false":	Optional	invisible cleared

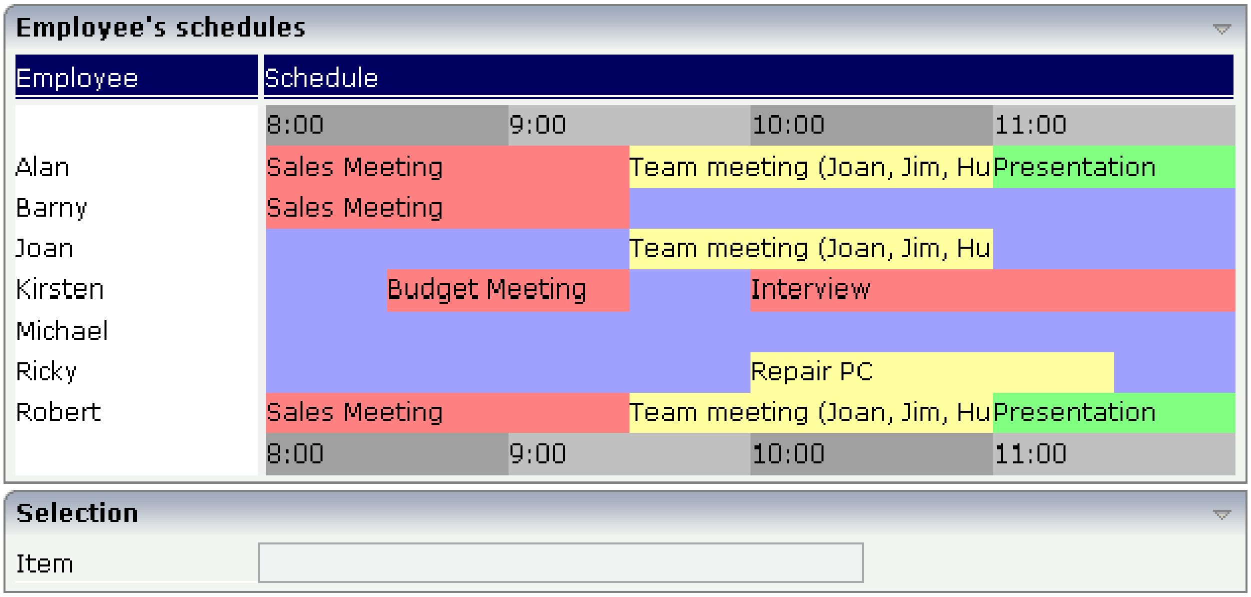
	(1) "invisible": the control is not visible. (2) "disabled": the control is deactivated: it is "grayed" and does not show any roll over effects any more.		
tabindex	Index that defines the tab order of the control. Controls are selected in increasing index order and in source order to resolve duplicates.	Optional	-1 0 1 2 5 10 32767
Label			
name	Text that is displayed inside the control. Please do not specify the name when using the multi language management - but specify a "textid" instead.	Optional	
textid	Multi language dependent text that is displayed inside the control. The "textid" is translated into a corresponding string at runtime. Do not specify a "name" inside the control if specifying a "textid".	Optional	
hdistpixelwidth	Witdh of the distance between checkbox and label in pixel.	Optional	
labelstyle	CSS style definition that is directly passed into this control. With the style you can individually influence the rendering of the control. You can specify any style sheet expressions. Examples are: border: 1px solid #FF0000 background-color: #808080 You can combine expressions by appending and separating them with a semicolon. Sometimes it is useful to have a look into the generated HTML code in order to know where direct style definitions are applied. Press right mouse-button in your browser and select the "View source" or "View frame's source" function.	Optional	background-color: #FF0000 color: #0000FF font-weight: bold
Binding			
valueprop	(already explained above)		

statusprop	Name of the adapter parameter that dynamically passes information how the control should be rendered and how it should act.	Optional	
flush	Flushing behaviour of the input control. By default an input into the control is registered within the browser client - and communicated to the server adapter object when a user e.g. presses a button. By using the FLUSH property you can change this behaviour. Setting FLUSH to "server" means that directly after changing the input a synchronization with the server adapter is triggered. As consequence you directly can react inside your adapter logic onto the change of the corresponding value. - Please be aware of that during the synchronization always all changed properties - also the ones that were changed before - are transferred to the adapter object, not only the one that triggered the synchonization. Setting FLUSH to "screen" means that the changed value is populated inside the page. You use this option if you have redundant usage of the same property inside one page and if you want to pass one changed value to all its representaion directly after changing the value.	Optional	screen server
flushmethod	$en/popupwizard/njx__attr_flushmethod$	Optional	
Online Help			
helpid	Help id that is passed to the online help management in case the user presses F1 on the control.	Optional	
title	Text that is shown as tooltip for the control. Either specify the text "hard" by using this TITLE property - or use the TITLETEXTID in order to define a language dependent literal.	Optional	
titletextid	Text ID that is passed to the multi lanaguage management - representing the tooltip text that is used for the control.	Optional	
Miscellaneous			
testtoolid	Use this attribute to assign a fixed control identifier that can be later on used within your test tool in order to do the object identification	Optional	

The RADIOBUTTON control is typically followed by a label explaining its meaning.

40 SCHEDULELINE

The SCHEDULELINE control is used to define screens like the following:



Employee's schedules

Employee	Schedule
	8:00 9:00 10:00 11:00
Alan	Sales Meeting Team meeting (Joan, Jim, Hu Presentation
Barny	Sales Meeting
Joan	Team meeting (Joan, Jim, Hu
Kirsten	Budget Meeting Interview
Michael	
Ricky	Repair PC
Robert	Sales Meeting Team meeting (Joan, Jim, Hu Presentation
	8:00 9:00 10:00 11:00

Selection

Item

You can display a certain sequence of items, each item holding a text, a color value, a size and an identifier. When clicking on an item, a certain event is sent to your adapter and the ID of the selected item is returned to perform activities in your program.

The following topics are covered below:

Properties

Basic			
valueprop	Name of the adapter parameter that represents the control in the adapter. It returns a semicolon separated list of schedule items. Each item is represented by a color, a width, a text and a selection id. The width is not a pixel width but represents a "portion" that this schedule item represents. Example: #FF0000\"1000;Text 1;1;#00FF00;500;Text 2;2 The total "logical width" is 1500. The firts item occupies 2/3 of the width, the right item occupies 1/3 of the width. The selection is required in case you want to react on user selections. If a user clicks onto one schedule item then the adapter is notified by a certain event - the id of the	Obligatory	

	schedule item is passed as reference. Please have a look into the corresponding property descriptions.		
width	Width of the control. There are three possibilities to define the width: (A) You do not define a width at all. In this case the width of the control will either be a default width or - in case of container controls - it will follow the width that is occupied by its content. (B) Pixel sizing: just input a number value (e.g. "100"). (C) Percentage sizing: input a percantage value (e.g. "50%"). Pay attention: percentage sizing will only bring up correct results if the parent element of the control properly defines a width this control can reference. If you specify this control to have a width of 50% then the parent element (e.g. an ITR-row) may itself define a width of "100%". If the parent element does not specify a width then the rendering result may not represent what you expect.	Obligatory	100 120 140 160 180 200 50% 100%
pixelheight	Height of the control in pixels.	Optional	
comment	Comment without any effect on rendering and behaviour. The comment is shown in the layout editor's tree view.	Optional	
Appearance			
width	(already explained above)		
pixelheight	(already explained above)		
pixelsizemode	A schedule line consists of sections, each one rendered with a certain width. By default the width does not represent a pixel value but represents a logical size. The width of the section depends on the logical size of one section compared with the logical size of the other sections. When switching this property to "true" then the size of the sections are interpreted as real pixel values.	Optional	true false
cellalign	Horizontal alignment of the text inside the control's schedule items.	Optional	left center right
cellvalign	Vertical alignement of the text inside the control's schedule items.	Optional	top middle bottom

cellstyle	Style that is used inside the schedule item cells. Can be any CSS style.	Optional	background-color: #FF0000 color: #0000FF font-weight: bold
cellnowrap	If switched to "true" then the text inside the schedule item cells is not broken if exceeding the size of the control - the text is cut instead. Default is "false".	Optional	true false
valign	Vertical alignment of control in its column. Each control is "packaged" into a column. The column itself is part of a row (e.g. ITR or TR). Sometimtes the size of the column is bigger than the size of the control. In this case the "align" property specify the position of the control inside the column.	Optional	top middle bottom
colspan	Column spanning of control. If you use TR table rows then you may sometimes want to control the number of columns your control occupies. By default it is "1" - but you may want to define the control to span over more than one columns. The property only makes sense in table rows that are snychronized within one container (i.e. TR, STR table rows). It does not make sense in ITR rows, because these rows are explicitly not synched.	Optional	1 2 3 4 5 50 int-value
rowspan	Row spanning of control. If you use TR table rows then you may sometimes want to control the number of rows your control occupies. By default it is "1" - but you may want to define the control two span over more than one columns. The property only makes sense in table rows that are snychronized within one container (i.e. TR, STR table rows). It does not make sense in ITR rows, because these rows are explicitly not synched.	Optional	1 2 3 4 5 50 int-value
crosslineids	Flag (true | false) that indicates that cells of different lines (within ROWTABLEAREA2) does not have same ids. If set to false the control is able to detect and skip unnecessary re-draws (performance).	Optional	true false

tablestyle	CSS style definition that is directly passed into this control. With the style you can individually influence the rendering of the control. You can specify any style sheet expressions. Examples are: border: 1px solid #FF0000 background-color: #808080 You can combine expressions by appending and separating them with a semicolon. Sometimes it is useful to have a look into the generated HTML code in order to know where direct style definitions are applied. Press right mouse-button in your browser and select the "View source" or "View frame's source" function.	Optional	background-color: #FF0000 color: #0000FF font-weight: bold
Binding			
valueprop	(already explained above)		
selectmethod	Name of the event that is sent to the adapter when the user selects one schedule item with the mouse.	Optional	
selscheduleprop	Name of an adapter parameter in which the id of the selected schedule item is passed.	Optional	
seltypeprop	Name of an adapter parameter that is used in the following way: If the user selects an item it can also be determined, if the item is selected by the left or by the right mouse button. In case the user uses the left mouse button, the value LEFT is passed into the property, which is referenced by the SELTYPEPROP property. In case the user uses the right mouse button, the value RIGHT is passed.	Optional	
preselectmode	If set to "true" then schedule items holding an id can be "preselected": the user can click on a schedule item and it is "grayed" as consequence - without directly calling the selection method. The selection method is called when double clicking onto the schedule item. Default is "false". The reaction of the control when clicking with the right mouse button remains untouched: still the selection method is called by a single right mouse button click.	Optional	true false
Vertical			
verticalschedule	Flag that indicates if the line is rendered vertically. Default is false.	Optional	true

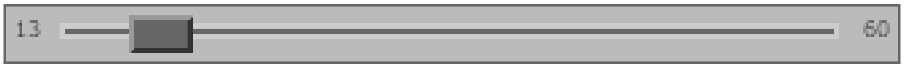
			false
tooltipprop	Name of an adapter parameter that contains the comma separated list of help texts that are displayed on mouse over (tooltip).	Optional	
imageprop	Name of an adapter parameter that returns a comma separated string of image URLs. An URL either is an absolute URL or a relative URL. If using a relative URL then be aware of that the generated page is located directly inside your project's directory. Example: "images/green.gif;;red.gif"	Optional	
imageorientation	Flag that indicates to render the image at the left or right hand of the text.	Optional	left right
dropinfoprop	Name of the adapter parameter to that the id of the dragged cell is set. Do not use this property if you do not want to support drag and drop within the SCHEDULELINE. The server side property needs to be of type "String".	Optional	
onmovemethod	Name of the event that is sent to the adapter on drop of one cell (source) over another cell (target). Use property DROPINFOPROP to get the id of the dragged cell (source). Use SELSCHEDULEPROP to get the id of the cell that got the drop (target).	Optional	
controlkeyprop	Name of an adapter parameter to that the information is set whether the user pressed the CTRL key when selecting a cell.	Optional	

41 SLIDER

The SLIDER control represents a slider. The main use of the slider is to limit the user input to specific values. It uses a number representation for its values, but the numbers can also be used to express string values.

The following topics are covered below:

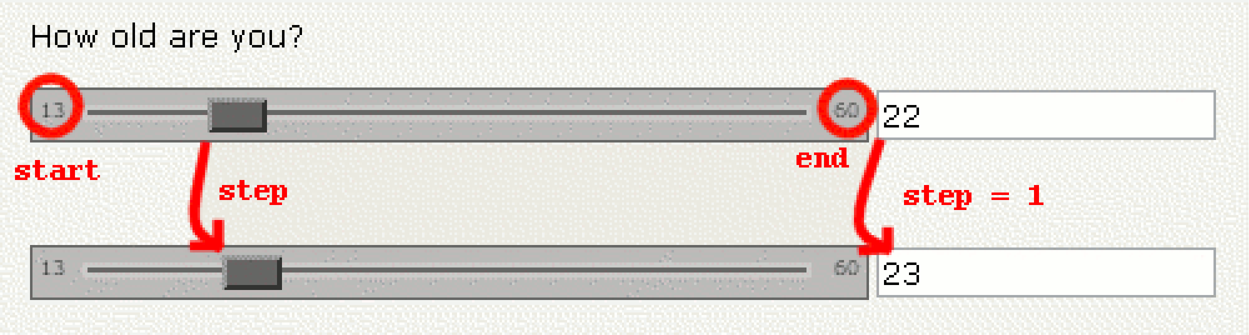
Example



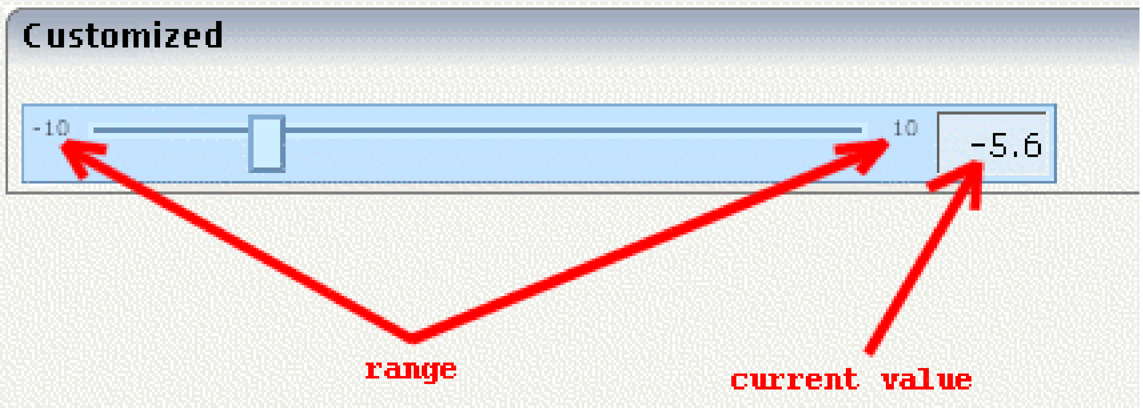
The XML layout definition is:

```
<rowarea name="Number Output">
    <itr>
        <slider valueprop="slider1" from="13" to="60" showrange="true"
                                              showcurrentvalue="false">
        </slider>
    </itr>
</rowarea>
```

The control can be customized by setting its start value, end value and a step. The start and end values form a closed interval. The step defines the distance between two valid values represented by the slider in this interval.



In the above example, the value for the step is the default value "1". The possible values represented by the slider are the integers from "13" to "60". It is possible to specify a floating-point number as a step, for example "0,25". The slider can be further customized by setting the properties `showrange` and `showcurrentvalue` which show the range (start and end value) and the current value of the slider while the user is moving it. The width and height of the slider point is adjustable. The slider point is the element which the user drags and drops. The colors, the borders of the slider, the point, the line, the range and the current value can also be customized.



Adapter Interface

```
DEFINE DATA PARAMETER
1 SLIDER
2 DISPLAYONLY (L)
2 FROM (F4)
2 SLIDERVALUE (F4)
2 STEP (F4)
2 TO (F4)
END-DEFINE
```

Properties

Basic			
valueprop	Name of the adapter parameter that provides the content of the control.	Obligatory	
Appearance			
width	Width of the slider. Can be given in pixels or percentage.	Optional	100 120 140 160 180 200 50%

			100%
displayonly	If set to true, the SLIDER will not be accessible for input. It is just used as an output.	Optional	true false
showrange	Boolean value. Whether to show the range of the slider. The range is the "from" and "to" values.	Optional	true false
showcurrentvalue	Boolean value. Whether to show the current value of the slider while it is moving.	Optional	true false
mainbgcolor	Background color of the slider container. This should be a valid CSS color value. For example a name(blue, red), a hexadecimal value(#99CCFF) or others.	Optional	#FF0000 #00FF00 #0000FF #FFFFFF #808080 #000000
mainbordercolor	Border color of the slider container. This should be a valid CSS border-color value. You can specify a different color for the top, right, bottom and left border in this sequence. For example: #BBBBBB #666666 #666666 #BBBBBB	Optional	#bbb #666 #666 #bbb #BFCFFF #00248F #00248F #BFCFFF
mainborderwidth	Border width of the slider container.	Optional	thin medium thick 1px 2px 5px 10px
pointbgcolor	Background color of the slider point. This should be a valid CSS color value. For example a name(blue, red), a hexadecimal value(#99CCFF) or others.	Optional	#FF0000 #00FF00 #0000FF

			#FFFFFF #808080 #000000
pointbordercolor	Border color of the slider point. This should be a valid CSS border-color value. You can specify a different color for the top, right, bottom and left border in this sequence. For example: #BBBBBB #666666 #666666 #BBBBBB	Optional	#bbb #666 #666 #bbb #BFCFFF #00248F #00248F #BFCFFF
pointborderwidth	Border width of the slider point.	Optional	thin medium thick 1px 2px 5px 10px
pointwidth	Width of the slider point in pixels. The value must be an integer value.	Optional	10 20 40 100 300
pointheight	Height of the slider point in pixels. The value must be an integer value.	Optional	10 20 40 100 300
linebgcolor	Background color of the slider line. This should be a valid CSS color value. For example a name(blue, red), a hexadecimal value(#99CCFF) or others.	Optional	#FF0000 #00FF00 #0000FF

			#FFFFFF #808080 #000000
linebordercolor	Border color of the slider line. This should be a valid CSS border-color value. You can specify a different color for the top, right, bottom and left border in this sequence. For example: #BBBBBB #666666 #666666 #BBBBBB	Optional	#bbb #666 #666 #bbb #BFCFFF #00248F #00248F #BFCFFF
lineborderwidth	Border width of the slider line.	Optional	thin medium thick 1px 2px 5px 10px
rangefontsize	Font size of the slider range.	Optional	xx-small x-small small medium large x-large xx-large smaller larger 150%
valuebgcolor	Background color of the slider current value which is shown if the "showcurrentvalue" property is set to true.	Optional	#FF0000 #00FF00 #0000FF

	This should be a valid CSS color value. For example a name(blue, red), a hexadecimal value(#99CCFF) or others.		#FFFFFF #808080 #000000
valuebordercolor	Background color of the slider current value which is shown if the "showcurrentvalue" property is set to true. This should be a valid CSS border-color value. You can specify a different color for the top, right, bottom and left border in this sequence. For example: #bbb #666 #666 #bbb	Optional	#bbb #666 #666 #bbb #BFCFFF #00248F #00248F #BFCFFF
valueborderwidth	Border width of the slider current value which is shown if the "showcurrentvalue" property is set to true.	Optional	thin medium thick 1px 2px 5px 10px
valuefontsize	Font size of the slider current value which is shown if the "showcurrentvalue" property is set to true.	Optional	xx-small x-small small medium large x-large xx-large smaller larger 150%

42 STRIPSEL

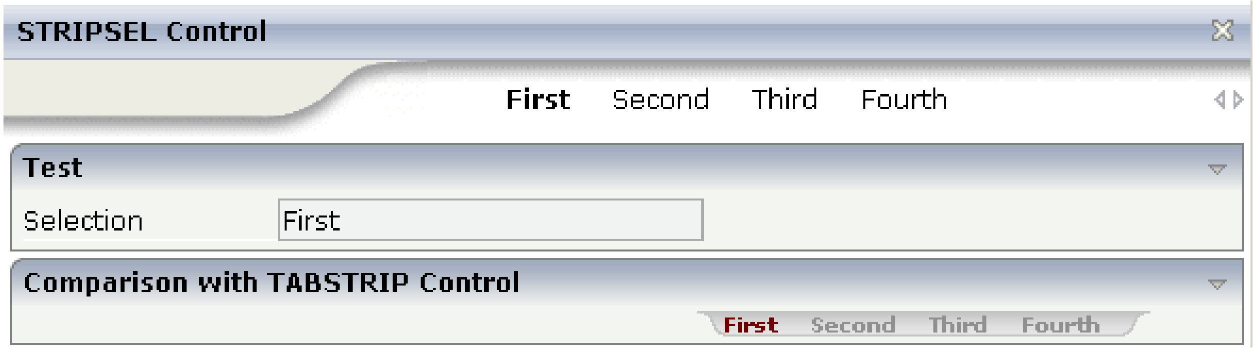
The STRIPSEL control is very similar to the TABSTRIP2 control: the user selects one option out of many.

The STRIPSEL control is typically located somewhere at the top of a page, but it can also be positioned anywhere else.

The following topics are covered below:

Example

Programming a STRIPSEL control is the same as programming the TABSTRIP2 control - just the rendering of the control differs:



Selection: First

In this example, the STRIPSEL control is the control below the titlebar. For comparison, the TABSTRIP2 control has also been added.

Properties

Basic			
tabstripprop	Name of the adapter parameter that represents the control in the adapter.	Optional	
align	Horizontal alignment of control in its column. Each control is "packaged" into a column. The column itself is part of a row (e.g. ITR or TR). Sometimes the size of the column is bigger than the size of the control itself. In this case the "align" property specifies the position of the control inside the column. In most cases you do not require the align control to be explicitly defined because the size of the column around the controls exactly is sized in the same way as the contained control.	Optional	left center right

	If you want to directly control the alignment of text: in most text based controls there is an explicit property "textalign" in which you align the control's contained text.		
scrollable	Flag that indicates if the control shows scroll icons on the right upper corner. Default is true	Optional	true false
backgroundstyle	CSS style definition that is directly passed into this control. With the style you can individually influence the rendering of the control. You can specify any style sheet expressions. Examples are: border: 1px solid #FF0000 background-color: #808080 You can combine expressions by appending and separating them with a semicolon. Sometimes it is useful to have a look into the generated HTML code in order to know where direct style definitions are applied. Press right mouse-button in your browser and select the "View source" or "View frame's source" function.	Optional	background-color: #FF0000 color: #0000FF font-weight: bold
scrolllefttitle	Help text that is displayed if the user moves the mouse of the scroll to left icon.	Optional	
scrolllefttitletextid	Multi language dependent text that is displayed inside the control. The "textid" is translated into a corresponding string at runtime. Do not specify a "name" inside the control if specifying a "textid".	Optional	
scrollrighttitle	Help text that is displayed if the user moves the mouse of the scroll to right icon.	Optional	
scrollrighttitletextid	Multi language dependent text that is displayed inside the control. The "textid" is translated into a corresponding string at runtime. Do not specify a "name" inside the control if specifying a "textid".	Optional	
comment	Comment without any effect on rendering and behaviour. The comment is shown in the layout editor's tree view.	Optional	

43 SUBPAGE

The SUBPAGE control defines an area in which an HTML page is shown. The URL of the page is not statically defined, but is dynamically controlled by the application.

Due to the browser's capability to embed installed plug-ins, you can use non-HTML objects to be called - and which the browser is able to understand. For example, if you have Microsoft Office installed (or the viewers for Microsoft Office documents) and you pass the name of a Word document as the URL, the Word document will be embedded into the page.

The following topics are covered below:

Properties

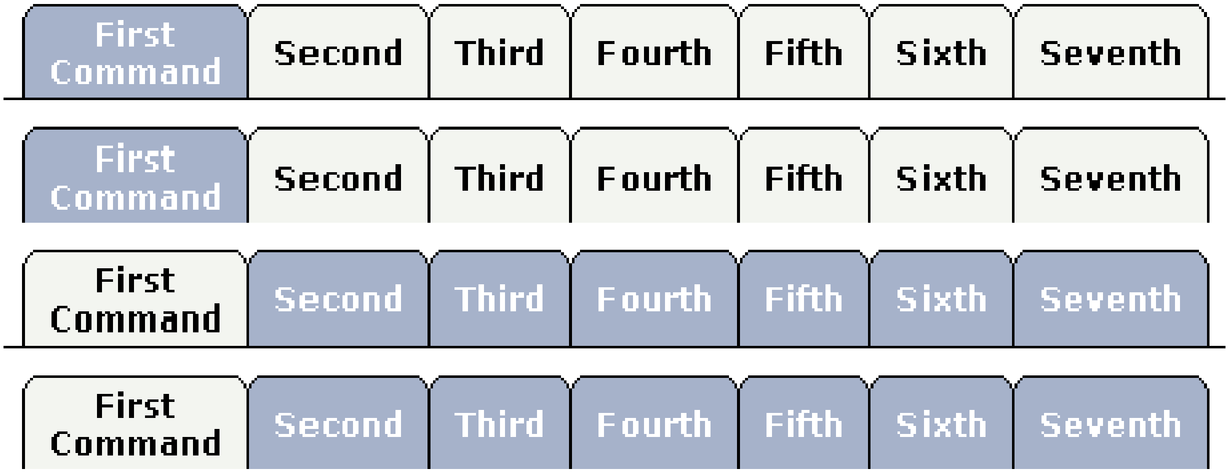
Basic			
valueprop	Name of the adapter parameter that provides the URL to be displayed inside the SUBPAGE control. Please note: the SUBPAGE control only re-renders its inner content if the URL provided by the property really changes. The SUBPAGE control does not "know" if something changed inside the contained page and that it has to redraw the page. - If you want to refresh the inner page explicitly append some random number to your URL, e.g.: http://...url...?RANDOM=45435. By changing the number the browser will reload the URL.	Obligatory	
width	Width of the control. There are three possibilities to define the width: (A) You do not define a width at all. In this case the width of the control will either be a default width or - in case of container controls - it will follow the width that is occupied by its content. (B) Pixel sizing: just input a number value (e.g. "100"). (C) Percentage sizing: input a percantage value (e.g. "50%"). Pay attention: percentage sizing will only bring up correct results if the parent element of the control properly defines a width this control can reference. If you specify this control to have a width of 50% then the parent element (e.g. an ITR-row) may itself define a width of "100%". If the parent element does not specify a width then the rendering result may not represent what you expect.	Sometimes obligatory	100 120 140 160 180 200 50% 100%
height	Height of the control. There are three possibilities to define the height: (A) You do not define a height at all. As consequence the control will be rendered with its default height. If the control is a container control	Sometimes obligatory	100 150 200 250

	(containing) other controls then the height of the control will follow the height of its content. (B) Pixel sizing: just input a number value (e.g. "20"). (C) Percentage sizing: input a percantage value (e.g. "50%"). Pay attention: percentage sizing will only bring up correct results if the parent element of the control properly defines a height this control can reference. If you specify this control to have a height of 50% then the parent element (e.g. an ITR-row) may itself define a height of "100%". If the parent element does not specify a width then the rendering result may not represent what you expect.		300 250 400 50% 100%
comment	Comment without any effect on rendering and behaviour. The comment is shown in the layout editor's tree view.	Optional	
Appearance			
width	(already explained above)		
height	(already explained above)		
scrolling	Definition of the scrollbar's appearance. You can define that the scrollbars only are shown if the content is exceeding the control's area ("auto"). Or scrollbars can be shown always ("scroll"). Or scrollbars are never shown - and the content is cut ("hidden"). Default is "auto".	Optional	auto yes no
pagestyle	CSS style definition that is directly passed into this control. With the style you can individually influence the rendering of the control. You can specify any style sheet expressions. Examples are: border: 1px solid #FF0000 background-color: #808080 You can combine expressions by appending and separating them with a semicolon. Sometimes it is useful to have a look into the generated HTML code in order to know where direct style definitions are applied. Press right mouse-button in your browser and select the "View source" or "View frame's source" function.	Optional	
colspan	Column spanning of control. If you use TR table rows then you may sometimes want to control the number of columns your control occupies. By default it is "1" - but you may want to define the control to span over more than one columns. The property only makes sense in table rows that are snychronized within one container (i.e. TR, STR table rows). It does not make sense in ITR rows, because these rows are explicitly not synched.	Optional	1 2 3 4 5

			50 int-value
rowspan	Row spanning of control. If you use TR table rows then you may sometimes want to control the number of rows your control occupies. By default it is "1" - but you may want to define the control two span over more than one columns. The property only makes sense in table rows that are snychronized within one container (i.e. TR, STR table rows). It does not make sense in ITR rows, because these rows are explicitly not synched.	Optional	1 2 3 4 5 50 int-value
Binding			
valueprop	(already explained above)		

44 TABSEL

The TABSEL control looks as shown in the following example:



The number of tabs is dynamically defined at runtime. There are various output options:

- With/without a horizontal line below the control.
- Normal or reverse coloring.

Like the TABSTRIP control, the TABSEL control does not provide internal containers that are switched when selecting tabs. It just represents one tab line.

The following topics are covered below:

Adapter Interface

```
DEFINE DATA PARAMETER
1 TABS
2 SELECTEDITEM (I4)
2 TSITEMS (1:*)
3 NAME (U) DYNAMIC
```

```
3 TITLE (U) DYNAMIC
END-DEFINE
```

Built-in Events

value-of-tabselprop.onSelect

Properties

Basic			
tabselprop	Name of the adapter parameter that represents the control in the adapter.	Obligatory	
bottomborder	If set to "true" then a bottom border is rendered below the tab selection. If set to "false" then no bottom border will be drawn.	Optional	true false
reversecolors	Reverses the color scheme of the TABSEL control.	Optional	true false
leftindent	Inserts a horizontal distance left of the first "tab" and shifts the "tabs" to the right as consequence. The value you may define represents the number of pixels that are inserted.	Optional	1 2 3 int-value
comment	Comment without any effect on rendering and behaviour. The comment is shown in the layout editor's tree view.	Optional	

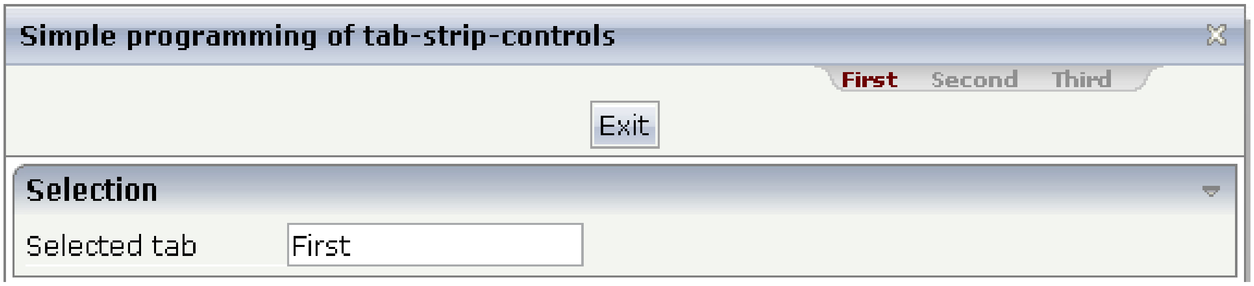
45 TABSTRIP2

The TABSTRIP2 control is used to navigate through certain aspects of your application. The way you navigate depends completely on your implementation.

The following topics are covered below:

Example

The control looks as follows:



Selected tab: First

For each aspect, there is one tab holding a name and an index. The left-most tab holds index 1, the next one 2, etc.

Interface Adapter

```
DEFINE DATA PARAMETER
1 TABS
2 SELINDEX (I4)
2 TSITEMS (1:*)
3 NAME (U) DYNAMIC
END-DEFINE
```

Built-in Events

`value-of-tabstripprop`.onSelect

Properties

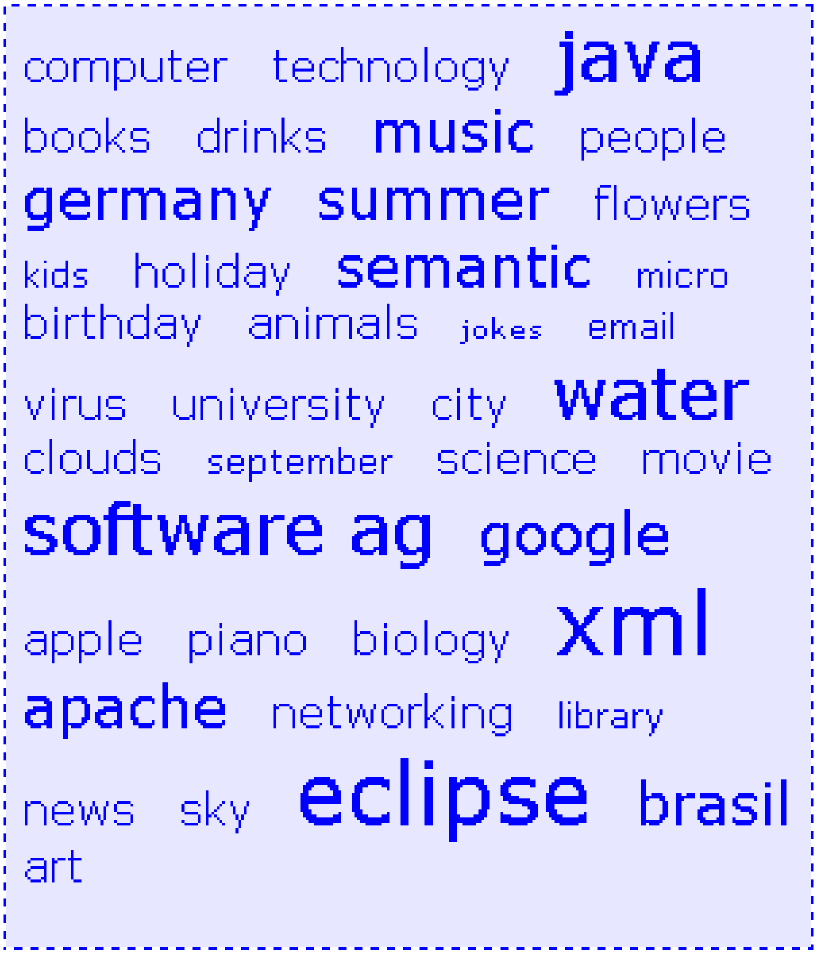
Basic			
tabstripprop	Name of the adapter parameter that represents the control in the adapter.	Optional	
align	Horizontal alignment of the control's content.	Optional	left center right
scrollable	If set to "true" then small icons will appear on the right border of the control. If the size of the "tabs" is too big and some tabs are cut as consequence then you can use these icons for scrolling left and right.	Optional	true false
backgroundstyle	CSS style definition that is directly passed into this control. With the style you can individually influence the rendering of the control. You can specify any style sheet expressions. Examples are: border: 1px solid #FF0000 background-color: #808080 You can combine expressions by appending and separating them with a semicolon. Sometimes it is useful to have a look into the generated HTML code in order to know where direct style definitions are applied. Press right mouse-button in your browser and select the "View source" or "View frame's source" function.	Optional	background-color: #FF0000 color: #0000FF font-weight: bold
comment	Comment without any effect on rendering and behaviour. The comment is shown in the layout editor's tree view.	Optional	

46 TAGCLOUD

The TAGCLOUD control represents a collection of tags. A tag is a keyword assigned to an information resource (picture, video clip or others). In a tag cloud, the tags are mainly shown by their popularity.

The following topics are covered below:

Example



computer technology java
books drinks music people
germany summer flowers
kids holiday semantic micro
birthday animals jokes email
virus university city water
clouds september science movie
software ag google
apple piano biology xml
apache networking library
news sky eclipse brasil
art

As you can see, different tags can be added to a tag cloud. They differ by their popularity. The most popular tags are those with a bigger font size.

The XML layout definition is:

```
<itr>
   <tagcloud tagcloudprop="tagCloud"
          width="300" height="350"
          borderstyle="dotted" borderwidth="1px"
          bordercolor="#0000FF" backgroundcolor="#E6E6FA"
          textcolor="#0000FF">
   </tagcloud>
</itr>
```

The tag cloud can be customized by defining a background color.

Interface Adapter

```
DEFINE DATA PARAMETER
1 TAGCLOUD
2 TCLITEM (1:*)
3 ID (U) DYNAMIC
3 POPULARITY (I4)
3 TEXT (U) DYNAMIC
END-DEFINE
```

Built-in Events

value-of-tagcloudprop.onSelect

Properties

Basic			
tagcloudprop	Name of the adapter parameter that represents the control in the adapter.	Obligatory	
width	Width of the control. There are three possibilities to define the width: (A) You do not define a width at all. In this case the width of the control will either be a default width or - in case of container controls - it will follow the width that is occupied by its content. (B) Pixel sizing: just input a number value (e.g. "100"). (C) Percentage sizing: input a percantage value (e.g. "50%"). Pay attention: percentage sizing will only bring up correct results if the parent element of the control properly defines a width this control can reference. If you specify this control to have a width of 50% then the parent element (e.g. an ITR-row) may itself define a width of "100%". If the parent element does not specify a width then the rendering result may not represent what you expect.	Optional	100 120 140 160 180 200 50% 100%
height	Height of the control. There are three possibilities to define the height:	Optional	100 150 200

	(A) You do not define a height at all. As consequence the control will be rendered with its default height. If the control is a container control (containing) other controls then the height of the control will follow the height of its content. (B) Pixel sizing: just input a number value (e.g. "20"). (C) Percentage sizing: input a percantage value (e.g. "50%"). Pay attention: percentage sizing will only bring up correct results if the parent element of the control properly defines a height this control can reference. If you specify this control to have a height of 50% then the parent element (e.g. an ITR-row) may itself define a height of "100%". If the parent element does not specify a width then the rendering result may not represent what you expect.		250 300 250 400 50% 100%
borderstyle	Choose the style the controls border.	Optional	solid double groove dotted dashed inset outset ridge hidden
borderwidth	Border size of control in pixels. Specify "0" not to render any border at all.	Optional	thin medium thick 1px 2px 5px 10px
bordercolor	Sets the border color of the control.	Optional	#FF0000 #00FF00 #0000FF #FFFFFF

			#808080 #000000
backgroundcolor	Sets the background color of the control.	Optional	#FF0000 #00FF00 #0000FF #FFFFFF #808080 #000000
textcolor	Sets the text color of the control.	Optional	#FF0000 #00FF00 #0000FF #FFFFFF #808080 #000000

47 TEXT

The TEXT control represents a multi line text edit control. It represents the value of an adapter parameter.

The following topics are covered below:

Properties

Basic			
valueprop	Name of the adapter parameter that provides the content of the control.	Obligatory	
width	Width of the control. There are three possibilities to define the width: (A) You do not define a width at all. In this case the width of the control will either be a default width or - in case of container controls - it will follow the width that is occupied by its content. (B) Pixel sizing: just input a number value (e.g. "100"). (C) Percentage sizing: input a percantage value (e.g. "50%"). Pay attention: percentage sizing will only bring up correct results if the parent element of the control properly defines a width this control can reference. If you specify this control to have a width of 50% then the parent element (e.g. an ITR-row) may itself define a width of "100%". If the parent element does not specify a width then the rendering result may not represent what you expect.	Sometimes obligatory	100 120 140 160 180 200 50% 100%
height	Height of the control. There are three possibilities to define the height: (A) You do not define a height at all. As consequence the control will be rendered with its default height. If the control is a container control (containing) other controls then the height of the control will follow the height of its content. (B) Pixel sizing: just input a number value (e.g. "20"). (C) Percentage sizing: input a percantage value (e.g. "50%"). Pay attention: percentage sizing will only bring up correct results if the parent element of the control properly defines a height this control can reference. If you specify this control to have a height of 50% then the parent element (e.g. an ITR-row) may itself define a height of "100%". If the parent element does not specify a width then the rendering result may not represent what you expect.	Optional	100 150 200 250 300 250 400 50% 100%
flush	Flushing behaviour of the input control.	Optional	screen server

	By default an input into the control is registered within the browser client - and communicated to the server adapter object when a user e.g. presses a button. By using the FLUSH property you can change this behaviour. Setting FLUSH to "server" means that directly after changing the input a synchronization with the server adapter is triggered. As consequence you directly can react inside your adapter logic onto the change of the corresponding value. - Please be aware of that during the synchronization always all changed properties - also the ones that were changed before - are transferred to the adapter object, not only the one that triggered the synchonization. Setting FLUSH to "screen" means that the changed value is populated inside the page. You use this option if you have redundant usage of the same property inside one page and if you want to pass one changed value to all its representaion directly after changing the value.		
flushmethod	$en/popupwizard/njx__attr_flushmethod$	Optional	
comment	Comment without any effect on rendering and behaviour. The comment is shown in the layout editor's tree view.	Optional	
Appearance			
width	(already explained above)		
displayonly	If set to true, the FIELD will not be accessible for input. It is just used as an output field.	Optional	true false
statusprop	Name of the adapter parameter that dynamically passes information how the control should be rendered and how it should act.	Optional	
wrap	Specifies the line wrapping inside the control. By default a line that exceeds the width of the control is broken automatically. You may define this property to not wrap at all ("off") - in this case the text control offers horizontal scroll bars to scroll the text. There are two styles of wrapping "soft" and "hard". The difference between "soft" and "hard" is the way the text is - if changed by the user - passed back to the adapter property: when specifying "soft" then line breaks which are caused by wrapping are not sent to the server, when specifying "hard" then line breaks caused by wrapping are sent as carriage return/ line feed. - Be carefule when specifying "hard" as consequence!	Optional	soft hard off
rows	Height of control specified by number of rows. Either define the height by the HEIGHT property or by the ROWS property. Do not specify both! When specifying the height by ROWS then be aware of that the height depends from the font size used inside the control (that is defined in the styles sheet definition).	Optional	

cols	Width of control specified by number of characters. Either define the width by the WIDTH property or by the COLS property. Do not specify both! When specifying the width by COLS then be aware of that the width depends from the font size used inside the control (that is defined in the styles sheet definition).	Optional	
colspan	Column spanning of control. If you use TR table rows then you may sometimes want to control the number of columns your control occupies. By default it is "1" - but you may want to define the control to span over more than one columns. The property only makes sense in table rows that are snychronized within one container (i.e. TR, STR table rows). It does not make sense in ITR rows, because these rows are explicitly not synched.	Optional	1 2 3 4 5 50 int-value
rowspan	Row spanning of control. If you use TR table rows then you may sometimes want to control the number of rows your control occupies. By default it is "1" - but you may want to define the control two span over more than one columns. The property only makes sense in table rows that are snychronized within one container (i.e. TR, STR table rows). It does not make sense in ITR rows, because these rows are explicitly not synched.	Optional	1 2 3 4 5 50 int-value
textareastyle	CSS style definition that is directly passed into this control. With the style you can individually influence the rendering of the control. You can specify any style sheet expressions. Examples are: border: 1px solid #FF0000 background-color: #808080 You can combine expressions by appending and separating them with a semicolon. Sometimes it is useful to have a look into the generated HTML code in order to know where direct style definitions are applied. Press right mouse-button in your browser and select the "View source" or "View frame's source" function.	Optional	
title	Text that is shown as tooltip for the control.	Optional	

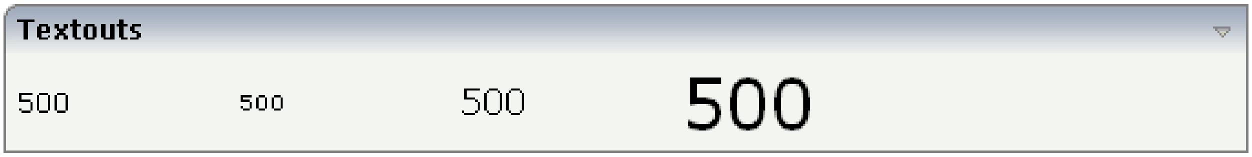
	Either specify the text "hard" by using this TITLE property - or use the TITLETEXTID in order to define a language dependent literal.		
titletextid	Text ID that is passed to the multi lanaguage management - representing the tooltip text that is used for the control.	Optional	
titleprop	Name of the adapter parameter that dynamically defines the title of the control. The title is displayed as tool tip when ther user moves the mouse onto the control.	Optional	
scroll	Definition of the scrollbar's appearance. You can define that the scrollbars only are shown if the content is exceeding the control's area ("auto"). Or scrollbars can be shown always ("scroll"). Or scrollbars are never shown - and the content is cut ("hidden"). Default is "auto".	Optional	auto scroll hidden
tabindex	Index that defines the tab order of the control. Controls are selected in increasing index order and in source order to resolve duplicates.	Optional	-1 0 1 2 5 10 32767
Online Help			
helpid	Help id that is passed to the online help management in case the user presses F1 on the control.	Optional	
title	(already explained above)		
titletextid	(already explained above)		
titleprop	(already explained above)		

48 TEXTOUT

The TEXTOUT control is used to display plain text. The text is not statically defined (as a label) but is controlled by an adapter property.

The following topics are covered below:

Example



Textouts

500 500 500 500

The XML layout definition is:

```
<rowarea name="Textouts">
    <itr>
        <textout valueprop="factor1" width="100">
        </textout>
        <textout valueprop="factor1" width="100" textsize="1">
        </textout>
        <textout valueprop="factor1" width="100" textsize="3">
        </textout>
        <textout valueprop="factor1" width="100" textsize="6">
        </textout>
    </itr>
</rowarea>
```

Properties

Basic			
width	Width of the control. There are three possibilities to define the width: (A) You do not define a width at all. In this case the width of the control will either be a default width or - in case of container controls - it will follow the width that is occupied by its content. (B) Pixel sizing: just input a number value (e.g. "100").	Sometimes obligatory	100 120 140 160 180 200

	(C) Percentage sizing: input a percantage value (e.g. "50%"). Pay attention: percentage sizing will only bring up correct results if the parent element of the control properly defines a width this control can reference. If you specify this control to have a width of 50% then the parent element (e.g. an ITR-row) may itself define a width of "100%". If the parent element does not specify a width then the rendering result may not represent what you expect.		50% 100%
valueprop	Name of the adapter parameter that provides the content of the control.	Obligatory	
comment	Comment without any effect on rendering and behaviour. The comment is shown in the layout editor's tree view.	Optional	
Appearance			
width	(already explained above)		
height	Height of the control. There are three possibilities to define the height: (A) You do not define a height at all. As consequence the control will be rendered with its default height. If the control is a container control (containing) other controls then the height of the control will follow the height of its content. (B) Pixel sizing: just input a number value (e.g. "20"). (C) Percentage sizing: input a percantage value (e.g. "50%"). Pay attention: percentage sizing will only bring up correct results if the parent element of the control properly defines a height this control can reference. If you specify this control to have a height of 50% then the parent element (e.g. an ITR-row) may itself define a height of "100%". If the parent element does not specify a width then the rendering result may not represent what you expect.	Optional	100 150 200 250 300 250 400 50% 100%
nowrap	If the textual content of the control exceeds the size of the control then the browser automatically breaks the line and arranges the text accordingly. You can avoid this behaviour by setting NOWRAP to "true". No line break will be performed by the browser.	Optional	true false
textsize	The HTML font size of the text. Corresponding to the HTML definition "1" means "smallest" and "6" means "biggest".	Optional	1 2 3 4 5

			6
textcolor	Colour of the text. Input a value like "#FF0000".	Optional	#FF0000 #00FF00 #0000FF #FFFFFF #808080 #000000
datatype	By default, the control is managing its content as string. By explicitly setting a datatype you can define that the control will format the data coming from the server: if the field has datatype "date" and the user inputs "010304" then the input will be translated into "01.03.2004" (or other representation, dependent on date format settings). Please note: the datatype "float" is named a bit misleading - it represents any decimal format number. The server side representation may be a float value, but also can be a double or a BigDecimal property.	Optional	date float int long time timestamp color ------------------------ N n.n P n.n string n xs:double xs:byte xs:short
straighttext	If the text of the control contains HTML tags then these are by default interpreted by the browser. Specifiying STRAIGHTTEXT as "true" means that the browser will directly render the characters without HTML interpretation. Example: if you want to output the source of an HTML text then STRAIGHTTEXT should be set to "true". MOZILLA: this property is not available in Mozilla!	Optional	true false
align	Horizontal alignment of control in its column.	Optional	left

	Each control is "packaged" into a column. The column itself is part of a row (e.g. ITR or TR). Sometimes the size of the column is bigger than the size of the control itself. In this case the "align" property specifies the position of the control inside the column. In most cases you do not require the align control to be explicitly defined because the size of the column around the controls exactly is sized in the same way as the contained control. If you want to directly control the alignment of text: in most text based controls there is an explicit property "textalign" in which you align the control's contained text.		center right
valign	Vertical alignment of control in its column. Each control is "packaged" into a column. The column itself is part of a row (e.g. ITR or TR). Sometimtes the size of the column is bigger than the size of the control. In this case the "align" property specify the position of the control inside the column.	Optional	top middle bottom
colspan	Column spanning of control. If you use TR table rows then you may sometimes want to control the number of columns your control occupies. By default it is "1" - but you may want to define the control to span over more than one columns. The property only makes sense in table rows that are snychronized within one container (i.e. TR, STR table rows). It does not make sense in ITR rows, because these rows are explicitly not synched.	Optional	1 2 3 4 5 50 int-value
rowspan	Row spanning of control. If you use TR table rows then you may sometimes want to control the number of rows your control occupies. By default it is "1" - but you may want to define the control two span over more than one columns. The property only makes sense in table rows that are snychronized within one container (i.e. TR, STR table rows). It does not make sense in ITR rows, because these rows are explicitly not synched.	Optional	1 2 3 4 5 50 int-value
bgcolorprop	Name of an adapter parameter that passes back a color value (e.g. "#FF0000" for red color). The color value is used as background color in the control. The color of the text color is automatically chosen dependent from the background	Optional	

	color: for light background colors the text color is black, for dark background colors the color is white. Use FGCOLORPROP to choose the text color on your own.		
fgcolorprop	Name of an adapter parameter that passes back a color value (e.g. "#FF0000" for red color). The color value is used as text color in the control. The background color is automatically chosen dependent from the text color: for dark text colors the background color is transparent (default), for light text colors the color is black. Use BGCOLORPROP to choose both - the text and background color.	Optional	
textoutstyle	CSS style definition that is directly passed into this control. With the style you can individually influence the rendering of the control. You can specify any style sheet expressions. Examples are: border: 1px solid #FF0000 background-color: #808080 You can combine expressions by appending and separating them with a semicolon. Sometimes it is useful to have a look into the generated HTML code in order to know where direct style definitions are applied. Press right mouse-button in your browser and select the "View source" or "View frame's source" function.	Optional	background-color: #FF0000 color: #0000FF font-weight: bold
textoutclass	CSS style class definition that is directly passed into this control. The style class can be either one which is part of the "normal" CIS style sheet files (i.e. the ones that you maintain with the style sheet editor) - or it can be one of an other style sheet file that you may reference via the ADDSTYLESHEET property of the PAGE tag.	Optional	
Binding			
valueprop	(already explained above)		
bgcolorprop	(already explained above)		
fgcolorprop	(already explained above)		
visibleprop	Name of the adapter parameter that provides the information if this control is displayed or not. As consequence you can control the visibility of the control dynamically.	Optional	
invisiblemode	If the visibility of the control is determined dynamically by an adapter property then there are two rendering modes if the visibility is "false": (1) "invisible": the control is not visible.	Optional	invisible cleared

	(2) "disabled": the control is deactivated: it is "grayed" and does not show any roll over effects any more.		

49 TOGGLE

The TOGGLE control is used to display and to edit a selection status. In principle, it acts similar to a CHECKBOX control, but it

- allows to define different icon images for the "true" and "false" representations;
- allows being informed when the user presses the CTRL or SHIFT key when clicking the icon. With this information, you can react on a combination of SHIFT and click in a different way than to a normal click or a combination of CTRL and click. This is especially useful inside grid processing when you want to allow the user to do mass selections.

The following topics are covered below:

Properties

Basic			
valueprop	Name of the adapter parameter that represents the value of the control.	Obligatory	
trueimage	Image URL that is shown if the corresponding property value is "true".	Obligatory	gif jpg jpeg
falseimage	Image URL that is shown if the corresponding property value is "true".	Obligatory	gif jpg jpeg
comment	Comment without any effect on rendering and behaviour. The comment is shown in the layout editor's tree view.	Optional	
Appearance			
width	Width of the control. There are three possibilities to define the width: (A) You do not define a width at all. In this case the width of the control will either be a default width or - in case of container controls - it will follow the width that is occupied by its content. (B) Pixel sizing: just input a number value (e.g. "100"). (C) Percentage sizing: input a percantage value (e.g. "50%"). Pay attention: percentage sizing will only bring up correct results if the parent element of the control properly defines a width this control can reference. If you specify this control to have a width of 50% then the parent element (e.g. an ITR-row) may itself define a width	Optional	100 120 140 160 180 200 50% 100%

	of "100%". If the parent element does not specify a width then the rendering result may not represent what you expect.		
height	Height of the control. There are three possibilities to define the height: (A) You do not define a height at all. As consequence the control will be rendered with its default height. If the control is a container control (containing) other controls then the height of the control will follow the height of its content. (B) Pixel sizing: just input a number value (e.g. "20"). (C) Percentage sizing: input a percantage value (e.g. "50%"). Pay attention: percentage sizing will only bring up correct results if the parent element of the control properly defines a height this control can reference. If you specify this control to have a height of 50% then the parent element (e.g. an ITR-row) may itself define a height of "100%". If the parent element does not specify a width then the rendering result may not represent what you expect.	Optional	
partialimage	Image URL that is shown if the corresponding property value is "null".	Optional	
colspan	Column spanning of control. If you use TR table rows then you may sometimes want to control the number of columns your control occupies. By default it is "1" - but you may want to define the control to span over more than one columns. The property only makes sense in table rows that are snychronized within one container (i.e. TR, STR table rows). It does not make sense in ITR rows, because these rows are explicitly not synched.	Optional	1 2 3 4 5 50 int-value
rowspan	Row spanning of control. If you use TR table rows then you may sometimes want to control the number of rows your control occupies. By default it is "1" - but you may want to define the control two span over more than one columns. The property only makes sense in table rows that are snychronized within one container (i.e. TR, STR table rows). It does not make sense in ITR rows, because these rows are explicitly not synched.	Optional	1 2 3 4 5 50 int-value
tabindex	Index that defines the tab order of the control. Controls are selected in increasing index order and in source order to resolve duplicates.	Optional	-1

			0 1 2 5 10 32767
backgroundclass	CSS style class definition that is directly passed into this control. The style class can be either one which is part of the "normal" CIS style sheet files (i.e. the ones that you maintain with the style sheet editor) - or it can be one of an other style sheet file that you may reference via the ADDSTYLESHEET property of the PAGE tag.	Optional	
Binding			
valueprop	(already explained above)		
statusprop	$en/popupwizard/njx_njx_attr_statusprop$	Optional	
shiftmethod	Name of the event that is sent to the adapter when the user clicks on the toggle control and presses the Shift-key the same time.	Optional	
controlmethod	Name of the event that is sent to the adapter when the user clicks on the toggle control and presses the Ctrl-key the same time.	Optional	
flush	Flushing behaviour of the input control. By default an input into the control is registered within the browser client - and communicated to the server adapter object when a user e.g. presses a button. By using the FLUSH property you can change this behaviour. Setting FLUSH to "server" means that directly after changing the input a synchronization with the server adapter is triggered. As consequence you directly can react inside your adapter logic onto the change of the corresponding value. - Please be aware of that during the synchronization always all changed properties - also the ones that were changed before - are transferred to the adapter object, not only the one that triggered the synchonization. Setting FLUSH to "screen" means that the changed value is populated inside the page. You use this option if you have redundant usage of the same property inside one page and if you want to pass one changed value to all its representaion directly after changing the value.	Optional	screen server
flushmethod	$en/popupwizard/njx__attr_flushmethod$	Optional	
Online Help			
title	Text that is shown as tooltip for the control.	Optional	

	Either specify the text "hard" by using this TITLE property - or use the TITLETEXTID in order to define a language dependent literal.		
titletextid	Text ID that is passed to the multi lanaguage management - representing the tooltip text that is used for the control.	Optional	

50 ACTIVEX

This is a 「hot topic」: embedding ActiveX controls in pages. Before telling you what the control does, let us explain why we do it:

Of course, the client integration of ActiveX controls has - from browser or SWT perspective - only disadvantages:

- ActiveX controls are not secure: you decide to run one control or not. But do not have a 「sandbox」 as you have with JavaScript or with applets. Using an ActiveX control means that this contol - once running - has native access to your computer, just as any other native program.
- ActiveX controls are bound to the Microsoft Windows platform.
- ActiveX controls need to be explicitly installed on the client side - maybe automated in some way, but still an explicit installation is necessary.

But - and this is why we support them - in some cases, they are a nice way to integrate other software which runs out of the scope of the browser.

Example: you may want to integrate your user interface with a barcode reader which is connected to your client via a serial interface. In this case, there is no way to access this barcode reader via JavaScript. You need to use an ActiveX control (or a signed applet) to connect to the serial device.

There is a simple interface between HTML/JavaScript and ActiveX, and vice versa. ActiveX controls can be embedded into an HTML page and it is possible to directly access properties of the ActiveX control from JavaScript. This interface was used for building the ACTIVEX control that you can use as an Application Designer control.

The following topics are covered below:

Properties

Basic			
classid	Class id of the ActiveX control. A string in the format "8E27C92B-1264-101C-8A2F-040224009C02" representing the UUID of the ActiveX component. The CLASSID is used inside the HTML client to reference the ActiveX control.	Optional	
progid	The unique program identifier which has been registered for this ActiveX Control like "Shell.Explorer"	Optional	
xinitparams	Init parameters that are used for creating an instance of the ActiveX control. Values are passed as semicolon separated string: property;value;property;value etc. The property is the name of the ActiveX control's property that is initialized with the corresponding value.	Optional	

setxparams	Same as GETXPARAMS but now the other direction. Adapter properties that are transferred (on change) into corresponding ActiveX properties with each repsonse. The string format is the same: activeXProperty;adapterProperty;activeXProperty;adapterProperty etc.	Optional	
getxparams	Semicolon separated list of which ActiveX control are linked with which adapter properties. The format is: activeXProperty;adapterProperty;activeXProperty;adapterProperty etc. With each request send from the browser the ActiveX properties are collected in from the ActiveX control and are transferred (if they have changed) into the corresponding adapter properties.activex_attr_progid"Program id of the ActiveX control. E.g. "MSCAL.Calendar" for the Microsoft calendar. The PROGID is used inside the SWT client to access the ActiveX control.	Optional	
width	Width of the control. There are three possibilities to define the width: (A) You do not define a width at all. In this case the width of the control will either be a default width or - in case of container controls - it will follow the width that is occupied by its content. (B) Pixel sizing: just input a number value (e.g. "100"). (C) Percentage sizing: input a percantage value (e.g. "50%"). Pay attention: percentage sizing will only bring up correct results if the parent element of the control properly defines a width this control can reference. If you specify this control to have a width of 50% then the parent element (e.g. an ITR-row) may itself define a width of "100%". If the parent element does not specify a width then the rendering result may not represent what you expect.	Optional	100 120 140 160 180 200 50% 100%
height	Height of the control. There are three possibilities to define the height: (A) You do not define a height at all. As consequence the control will be rendered with its default height. If the control is a container control (containing) other controls then the height of the control will follow the height of its content. (B) Pixel sizing: just input a number value (e.g. "20"). (C) Percentage sizing: input a percantage value (e.g. "50%"). Pay attention: percentage sizing will only bring up correct results if the parent element of the control properly defines a height this control can reference. If you specify this control to have a height of 50% then the parent element (e.g. an ITR-row) may itself define a height of "100%". If the parent element does not specify a width then the rendering result may not represent what you expect.	Optional	100 150 200 250 300 250 400 50% 100%
reloadprop	Inidicates that the ActiveX component is reloaded with every repsonse from the server that changed data of the ActiveX component.	Optional	

reloadprop	Inidicates that the ActiveX component is reloaded with every response from the server that changed data of the ActiveX component.	Optional	
comment	Comment without any effect on rendering and behaviour. The comment is shown in the layout editor's tree view.	Optional	

51 GOOGLEMAP2

The GOOGLEMAP2 control is used to provide for Google Maps support within Application Designer pages. The control internally makes use of the Google Maps API. In order to use the control on your site, you need to sign up for a Google Maps API key at ***http://www.google.com/apis/maps/***. Make sure that you agree with the Google Maps API Terms of Use (***http://www.google.com/apis/maps/terms.html***).

The following topics are covered below:

Before You Start

In order to use the GOOGLEMAP2 control, you need to sign up for a Google Maps API key. A key is valid for a single 「directory」 on your web server only, i.e. you sign up for a URL like *http://www.mysite.com/mywebapp/myproject*. With a standard installation of Application Designer on localhost, you may sign up for the URL *http://localhost:8080/mywebapp/myproject*. Typically, you develop your Application Designer web application not on the site on which you run it later in productive mode. Therefore, you may sign up for two different sites (development and production site).

Required Steps

1. Choose the project directory that keeps the layouts using the GOOGLEMAP2 control.
2. Sign up for a Google Maps API key at ***http://www.google.com/apis/maps/*** for this project directory (e.g. *http://localhost:8080/mywebapp/myproject*).
3. Create the API key page. Store the key page in the registered project directory. You are free in naming the file (the file extension must be "html"). The GOOGLEMAP2 control embeds your API key as a subpage. The subpage must have the following minimum structure:

```
<html>
  <head>
    <script src="
http://maps.google.com/maps?file=api&amp;v=2.x&amp;key=YOUR_API_KEY"></script>
    <script src="../HTMLBasedGUI/general/googlemapsscript.js"></script>
  </head>
  <body>
    <div id="map" style="position:absolute; top0; left:0;"></div>
  </body>
</html>
```

   You see that the page includes two JavaScript libraries. The first line refers to the Google Maps API. Replace the placeholder "YOUR_API_KEY" with your Google Maps API key. With the second line, the page includes the control's scripting (calls from Application Designer to the Google Maps). The page body is quite simple: it contains a single `div` tag with the ID "map". This `div` is used as an anchor to insert Google Maps controls dynamically.

Example

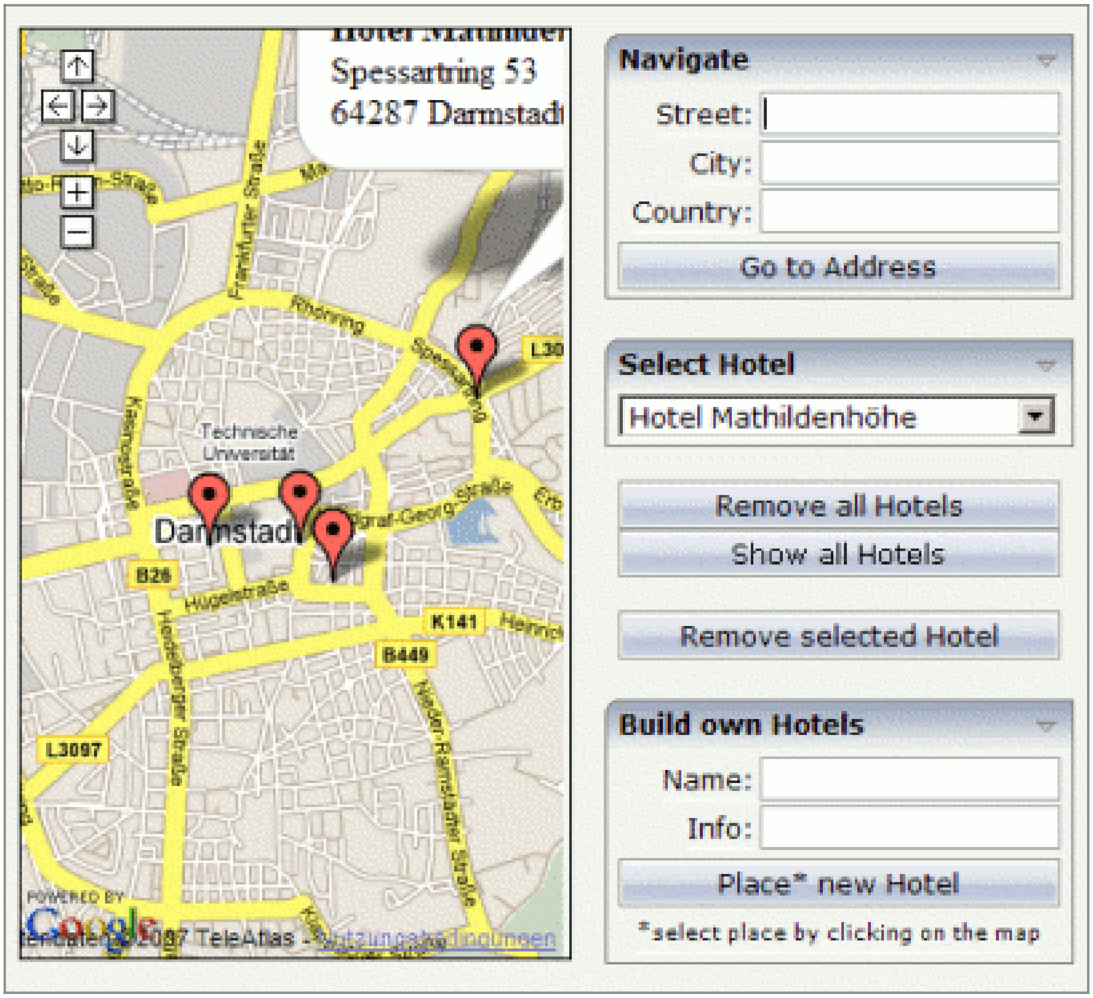
The following topics are covered below:

- General Usage

General Usage

The map options are taken from the property `infoprop`. On this object, you may set the address (or latitude and longitude), the zoom level and the map size as well as the map type.

Note: The usage of address or longitude/latitude is mutually exclusive.



Spessartring 53
64287 Darmstadt

Select Hotel: Hotel Mathildenhöhe

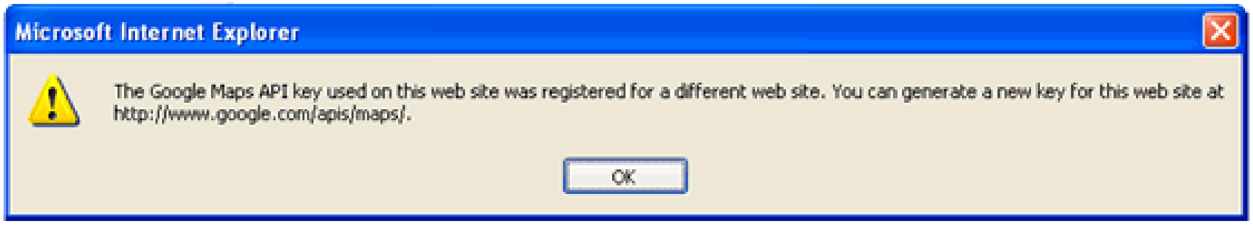
Typical Problems

The following topics are covered below:

- Google Map API Key
- Map Remains Gray

Google Map API Key

Your Google Maps API key is bound to a directory on a certain web server (i.e. you sign up for the URL *http://mycomputer.mydomain.com:8080/mywebapp/myproject*). If you use your key for another URL, Google shows an error message:



The Google Maps API key used on this web site was registered for a different web site. You can generate a new key for this web site at http://www.google.com/apis/maps/.

Reasons that cause the error:

- You have registered your computer using the computer's name (e.g. *http://mycomputer...*). But the Application Designer development workplace is started using the URL *http://localhost...*.

  Solution: start the Application Designer workplace with *http://mycomputer...*.

- The registered directory (e.g. *.../mywebapp/myproject*) does not match your installation (either a mistake in writing when signing up for the key or you have renamed the web application or project after registration).

  Solution: rename your web application or project to match the registered names. Or sign up for a new key and insert the new key into the API key page. In the latter case, delete the content of the browser's cache. Otherwise, the browser will use the former API key page (and thus the old key).

Map Remains Gray

If you use longitude and latitude for placing the marker on the map, their values may exceed the map top (or bottom) border. If you are able to find the map by scrolling down (or up), then this is the case. Check the values for longitude and latitude in this case.

Properties

Basic			
infoprop	$en/popupwizard/njx_googlemap2_attr_infoprop$	Obligatory	
apikeypagename	Name of the Maps API Key page. Example: mygooglemapsapikey.html. Keep this file within the project directory (directory within the CIS HTML pages are kept). The GOOGLEMAP-control expects this file within certain Javascript includes and content. Have look into chapter "Google Map - Before You Start" within the Developers Guide	Obligatory	
width	Width of the control. There are three possibilities to define the width: (A) You do not define a width at all. In this case the width of the control will either be a default width or - in case of container controls - it will follow the width that is occupied by its content. (B) Pixel sizing: just input a number value (e.g. "100"). (C) Percentage sizing: input a percantage value (e.g. "50%"). Pay attention: percentage sizing will only bring up correct results if the parent element of the control properly defines a width this control can reference. If you specify this control to have a width of 50% then the parent element (e.g. an ITR-row) may itself define a width of "100%". If the parent element does not specify a width then the rendering result may not represent what you expect.	Optional	100 120 140 160 180 200 50% 100%
height	Height of the control. There are three possibilities to define the height: (A) You do not define a height at all. As consequence the control will be rendered with its default height. If the control is a container control (containing) other controls then the height of the control will follow the height of its content. (B) Pixel sizing: just input a number value (e.g. "20"). (C) Percentage sizing: input a percantage value (e.g. "50%"). Pay attention: percentage sizing will only bring up correct results if the	Optional	100 150 200 250 300 250 400

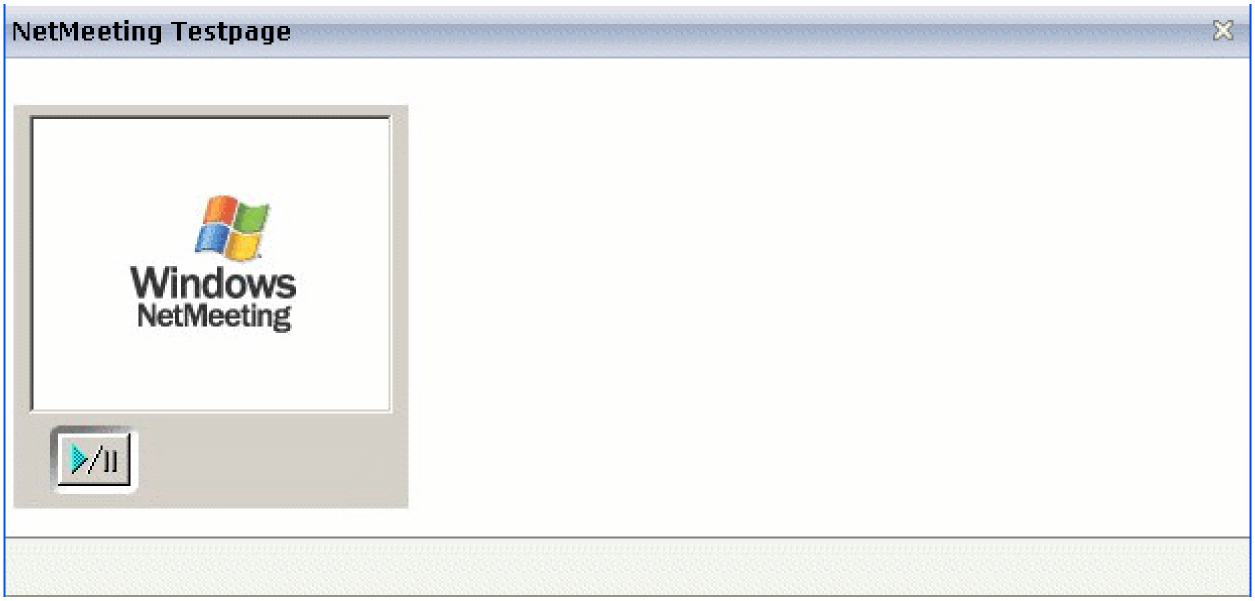
	parent element of the control properly defines a height this control can reference. If you specify this control to have a height of 50% then the parent element (e.g. an ITR-row) may itself define a height of "100%". If the parent element does not specify a width then the rendering result may not represent what you expect.		50% 100%
comment	Comment without any effect on rendering and behaviour. The comment is shown in the layout editor's tree view.	Optional	
Appearance			
pagestyle	CSS style definition that is directly passed into this control. With the style you can individually influence the rendering of the control. You can specify any style sheet expressions. Examples are: border: 1px solid #FF0000 background-color: #808080 You can combine expressions by appending and separating them with a semicolon. Sometimes it is useful to have a look into the generated HTML code in order to know where direct style definitions are applied. Press right mouse-button in your browser and select the "View source" or "View frame's source" function.	Optional	
rowspan	Row spanning of control. If you use TR table rows then you may sometimes want to control the number of rows your control occupies. By default it is "1" - but you may want to define the control two span over more than one columns. The property only makes sense in table rows that are snychronized within one container (i.e. TR, STR table rows). It does not make sense in ITR rows, because these rows are explicitly not synched.	Optional	1 2 3 4 5 50 int-value
colspan	Column spanning of control. If you use TR table rows then you may sometimes want to control the number of columns your control occupies. By default it is "1" - but you may want to define the control to span over more than one columns. The property only makes sense in table rows that are snychronized within one container (i.e. TR, STR table rows). It does not make sense in ITR rows, because these rows are explicitly not synched.	Optional	1 2 3 4 5 50 int-value

52 NETMEETING

The NETMEETING control allows you to start NetMeeting sessions within your Application Designer pages.

The following topics are covered below:

Example



The XML layout definition is:

```
<pagebody>
  <itr>
    <netmeeting calltoprop="callto" modeprop="modep" width="300">
    </netmeeting>
  </itr>
</pagebody>
```

Properties

Basic			
calltoprop	Name of the adapter parameter that provides the contact data of the 'contact' that should be called. The data has to have the following semantics. ILS Server/email adress e.g. ils.netmeeting.de/contact@testmail.com	Optional	
modeprop	Name of the adapter parameter that holds the mode of the control. Possible are: FULL, PREVIEWONLY, PREVIEWNOPAUSE, REMOTEONLY, REMOTENOPAUSE, DATAONLY	Optional	
width	Width of the control. There are three possibilities to define the width: (A) You do not define a width at all. In this case the width of the control will either be a default width or - in case of container controls - it will follow the width that is occupied by its content. (B) Pixel sizing: just input a number value (e.g. "100"). (C) Percentage sizing: input a percantage value (e.g. "50%"). Pay attention: percentage sizing will only bring up correct results if the parent element of the control properly defines a width this control can reference. If you specify this control to have a width of 50% then the parent element (e.g. an ITR-row) may itself define a width of "100%". If the parent element does not specify a width then the rendering result may not represent what you expect.	Optional	100 120 140 160 180 200 50% 100%

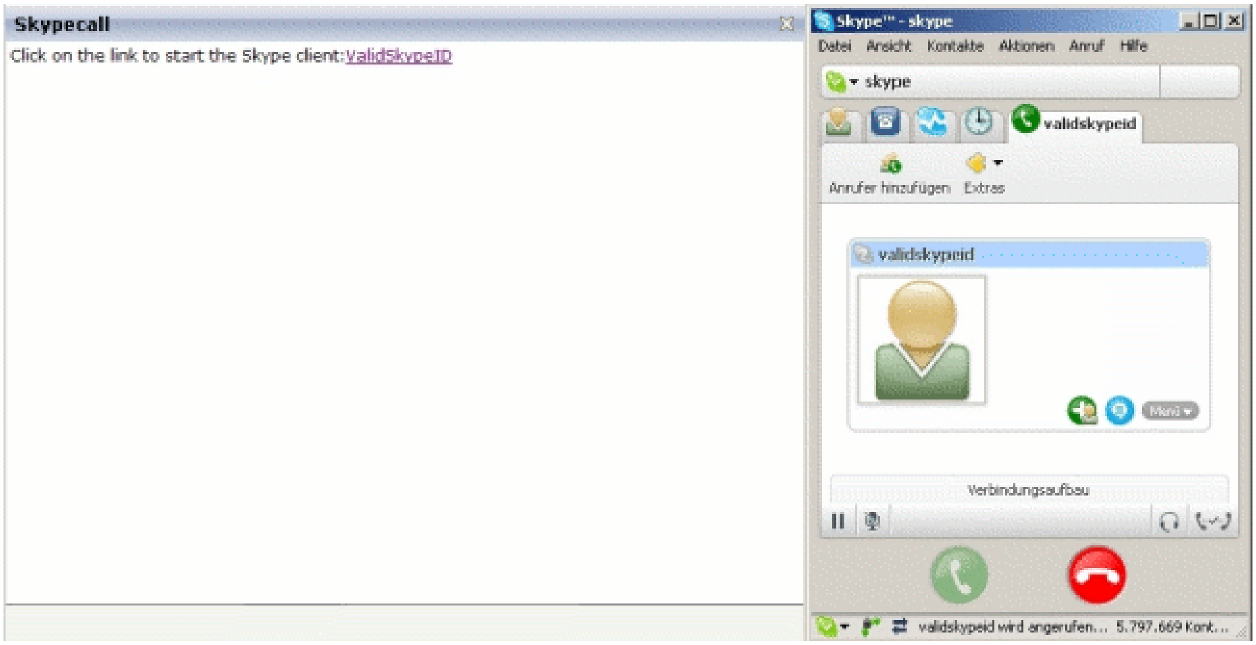
53 SKYPECALL

The SKYPECALL control allows you to start the Skype client with given contact data from your Application Designer pages.

⚠ **Important:** In order to use the SKYPECALL control you need to have a valid Skype account and the Skype client must be installed. For further information, see *http://www.skype.com/*.

The following topics are covered below:

Example



Skypecall

Click on the link to start the Skype client:ValidSkypeID

validskypeid wird angerufen... 5.797.669 Kont...

The XML layout definition is:

```
<pagebody>
  <itr>
   <label name="Click on the link to start the Skype client: "
        asplaintext="true"></label>
   <skypecall valueprop="skypecall"></skypecall>
  </itr>
</pagebody>
```

Properties

Basic			
valueprop	Name of the adapter parameter that contains the phone number or the Skype ID of the person that should be called. It is also possible to set some parameters. For further information, see the Skype API. Note: The Skype client must be installed if you want to use this control.	Obligatory	

54 Working with Grids

This chapter shows you how to deal with grids. Working with grids is as simple as working with singular properties because the grid management adapts seamlessly into the normal processing of the Application Designer environment.

The information provided in this part is organized under the following headings:

55 Basics

It is quite simple: 「normal」 controls refer to an adapter and are bound to adapter parameters. Grid controls refer to an adapter as well - but are bound to a group array. Each array element provides group elements to access its content.

Two types of grid controls are available:

- The TEXTGRID2 control is a control that displays grid data - but does not allow any change to the data. You can select grid rows and colorize them in different ways. Change the order of columns dynamically and sort columns by clicking into the title row of the grid.

  There is a TEXTGRIDSSS2 control that is a certain variant of the TEXTGRID2 control.
- The ROWTABLEAREA2 is a container that internally allows you to use any normal control to be embedded inside a grid. Therefore, you can place normal FIELD controls, CHECKBOX controls etc. inside the ROWTABLEAREA2 container.

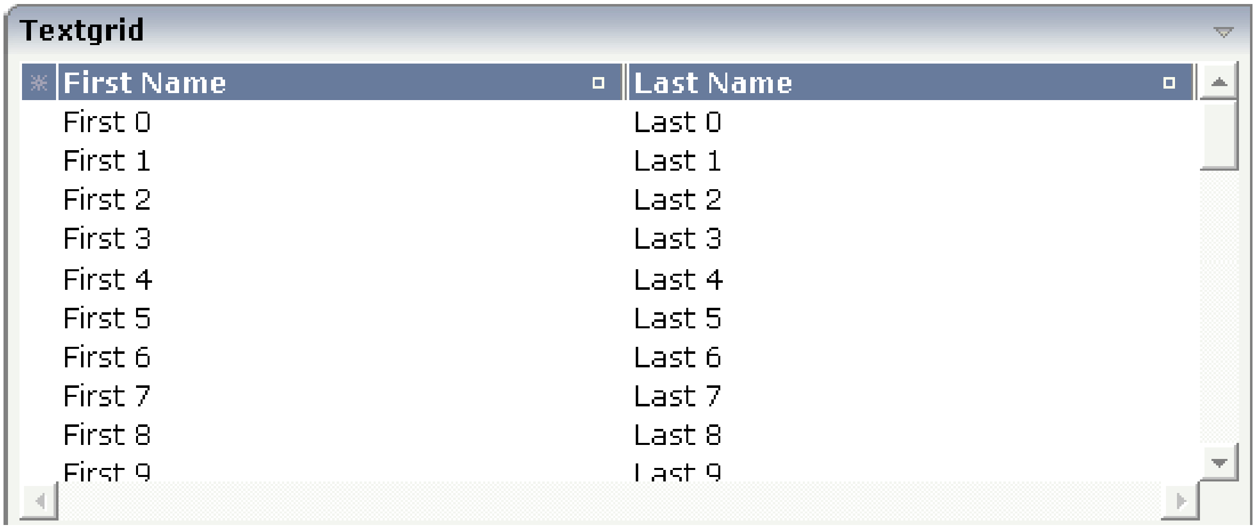
Use the TEXTGRID2 controls for displaying and selecting data. Use ROWTABLEAREA2 for entering data inside a grid.

56 TEXTGRID2

This chapter covers the following topics:

A Simple Example

The following example shows a TEXTGRID2 control:



Textgrid

First Name	Last Name
First 0	Last 0
First 1	Last 1
First 2	Last 2
First 3	Last 3
First 4	Last 4
First 5	Last 5
First 6	Last 6
First 7	Last 7
First 8	Last 8
First 9	Last 9

There are two columns which hold data. There is one column at the very left which displays a selection icon - in addition to a yellow background for a selected line. Even and odd lines are displayed in slightly different colors. At the very right of each title column, there is a symbol which indicates the sorting status; if you double-click on this symbol, the column is sorted first in ascending direction and, when clicking again, in descending direction. Change the sequence of columns by dragging the title of a column and dropping it on another column's title. Depending from where you drop, the column is either moved left or right.

The asterisk in the upper left corner of the grid is used to select/deselect all lines in the grid. The behavior depends on the setting of the `singleselect` property which determines whether multiple lines can be selected in the grid (default) or whether only one line can be selected:

- **Multiple Line Selection Mode**
  When you choose the asterisk for the first time, all lines are selected. When you choose the asterisk a second time, all lines are deselected.
- **Single Line Selection Mode**
  When you choose the asterisk (no matter how often), an existing selected line is deselected.

The XML layout definition is:

```
<rowarea name="Textgrid">
    <itr takefullwidth="true" fixlayout="true">
        <textgrid2 griddataprop="lines" width="100%" height="200"
selectprop="selected"
                   hscroll="true">
            <column name="First Name" property="firstName" width="50%">
            </column>
            <column name="Last Name" property="lastName" width="50%">
            </column>
        </textgrid2>
    </itr>
    <vdist height="5">
    </vdist>
</rowarea>
```

The TEXTGRID2 definition is bound to a grid data property `lines`.

Inside the TEXTGRID2 control definition there are two columns. These columns are bound to the properties `firstName` and `lastName`.

Adapter Interface

In the parameter data area of the adapter, the grid data is represented by the following data structure:

```
DEFINE DATA PARAMETER
1 LINES (1:*)
2 FIRSTNAME (U) DYNAMIC
2 LASTNAME (U) DYNAMIC
```

```
2 SELECTED (L)
END-DEFINE
```

Selecting Rows in a TEXTGRID2

Maybe you wonder why there is a `selected` field in the adapter parameter data area of the previous example.

This field is required for indicating which lines are currently selected and which are not. Each line which is displayed in the TEXTGRID2 control is represented in the adapter by an array occurrence of the array `LINES`. Therefore, the selection status of the grid (which lines are selected and which lines are not) is mirrored by the corresponding `selected` field of each array occurrence.

TEXTGRID2 Properties

Basic			
griddataprop	Name of the adapter parameter that represents the grid in the adapter.	Obligatory	
width	Width of the control. There are three possibilities to define the width: (A) You do not define a width at all. In this case the width of the control will either be a default width or - in case of container controls - it will follow the width that is occupied by its content. (B) Pixel sizing: just input a number value (e.g. "100"). (C) Percentage sizing: input a percantage value (e.g. "50%"). Pay attention: percentage sizing will only bring up correct results if the parent element of the control properly defines a width this control can reference. If you specify this control to have a width of 50% then the parent element (e.g. an ITR-row) may itself define a width of "100%". If the parent element does not specify a width then the rendering result may not represent what you expect.	Obligatory	100 120 140 160 180 200 50% 100%
height	Height of the control. There are three possibilities to define the height: (A) You do not define a height at all. As consequence the control will be rendered with its default height. If the control is a container control (containing) other controls	Obligatory	100 150 200 250

	then the height of the control will follow the height of its content. (B) Pixel sizing: just input a number value (e.g. "20"). (C) Percentage sizing: input a percantage value (e.g. "50%"). Pay attention: percentage sizing will only bring up correct results if the parent element of the control properly defines a height this control can reference. If you specify this control to have a height of 50% then the parent element (e.g. an ITR-row) may itself define a height of "100%". If the parent element does not specify a width then the rendering result may not represent what you expect.		300 250 400 50% 100%
comment	Comment without any effect on rendering and behaviour. The comment is shown in the layout editor's tree view.	Optional	
Selection			
selectprop	Name of the adapter parameter that is used to mark if an individual row of the text grid is selected. If the user selects a text grid row, the value "true" is passed into the corresponding array element of the adapter parameter.	Optional	
singleselect	If set to "true" then only one row can be selected inside the text grid. - If set to "false" then multiple lines can be selected by using Ctrl- and Shift-key during mouse selection. Default is "false".	Optional	true false
singleselectprop	Name of an adapter parameter that dynamically defines whether SINGLESELECT is true or false.	Optional	
onclickmethod	Name of the event that is sent to the adapter when the user selects a row. In the event handler you can find the selected rows by iterating through the rows and finding out which one's selected element is set to "true".	Optional	
ondblclickmethod	Name of the event that is sent to the adapter when the user selects a row by a double click. In the event handler you can find the selected rows by iterating through the rows and finding out which one's selected element is set to "true".	Optional	
withselectioncolumn	When defining a SELECTPROP property then automatically a selection column is added as first left column of the grid. Inside the column an icon inidicates if a row is currently selected.	Optional	true false

	Set this property to "false" in order to avoid the selection column.		
withselectioncolumnicon	Flag that indicates whether the selection column shows a "select all" icon on top. Default is true.	Optional	true false
fgselect	if switched to true then an additional "graying" of selected lines will be activated. Switch this property to "true" if you have coloured textgrid cells: the selection colour will not override the colour of each cell, as consequence you require an additional effect in order to make the user see which row is selected.	Optional	true false
focusedprop	Name of an adapter parameter that is used to mark if an individual row of the text grid should receive the focus. If the user selects a text grid row, the value "true" is passed into the corresponding array element of the adapter parameter.	Optional	
Right Mouse Button			
oncontextmenumethod	Name of the event that is sent to the adapter when the user clicks with the right mouse button onto an empty area of the grid.	Optional	
singleselectcontextmenu	With SHIFT and CTRL key the user can select multiple lines (use property SINGLESELECT to suppress this feature). Use this property to ensure that the context menu is requested only for a single line. Default is "false".	Optional	true false
enabledefaultcontextmenu	Use this property to enable the default context menu of the browser within the textgrid. Please note: do not enable the browser's context menu if your application itself provides for a context menu. Default is "false".	Optional	true false
Appearance			
width	(already explained above)		
height	(already explained above)		
minapparentrows	Number of rows that are displayed independent of the size of the server side collection.	Optional	1 2 3 int-value
hscroll	Indicates if to show a horizontal scrollbar ("true") or not ("false").	Sometimes obligatory	true

	If no scrollbar is shown then the control occupies the horizontal space that is required by its content.		false
withtitlerow	If defined as "false" then no top title row is shown. "True" is default.	Optional	true false
colspan	Column spanning of control. If you use TR table rows then you may sometimes want to control the number of columns your control occupies. By default it is "1" - but you may want to define the control to span over more than one columns. The property only makes sense in table rows that are snychronized within one container (i.e. TR, STR table rows). It does not make sense in ITR rows, because these rows are explicitly not synched.	Optional	1 2 3 4 5 50 int-value
rowspan	Row spanning of control. If you use TR table rows then you may sometimes want to control the number of rows your control occupies. By default it is "1" - but you may want to define the control two span over more than one columns. The property only makes sense in table rows that are snychronized within one container (i.e. TR, STR table rows). It does not make sense in ITR rows, because these rows are explicitly not synched.	Optional	1 2 3 4 5 50 int-value
personalizable	If defined to "false" then no re-arranging of columns is offered to the user. Default is "true". This means: if using COLUMN controls inside the grid definition then the user can re-arrange the sequence of columns by dragging and dropping them within the top title row.	Optional	true false
stylevariant	Some controls offer the possibility to define style variants. By this style variant you can address different styles inside your style sheet definition file (.css). If not defined "normal" styles are chosen, if defined (e.g. "VAR1") then other style definitions (xxxVAR1xxx) are chosen. Purpose: you can set up style variants in the style sheet defintion and use them multiple times by addressing them via the "stylevariant" property. CIS currently offerst two variants "VAR1" and "VAR2" but does not predefine any semantics behind - this is up to you!	Optional	VAR1 VAR2

backgroundstyle	CSS style definition that is directly passed into this control. With the style you can individually influence the rendering of the control. You can specify any style sheet expressions. Examples are: border: 1px solid #FF0000 background-color: #808080 You can combine expressions by appending and separating them with a semicolon. Sometimes it is useful to have a look into the generated HTML code in order to know where direct style definitions are applied. Press right mouse-button in your browser and select the "View source" or "View frame's source" function.	Optional	
vscroll	Definition of the vertical scrollbar's appearance. You can define that scrollbars only are shown if the content is exceeding the control's area ("auto"). Or scrollbars can be shown always ("scroll"). Or scrollbars are never shown - and the content is cut ("hidden"). Default is "auto".	Optional	auto scroll hidden
withrollover	The textgrid controls provide for a so called "roll over" effect. The row that is currently below the mouse pointer is highlighted in a certain way. Use this property to disable the roll over effect (Default is TRUE).	Optional	true false
fixedcolumnsizes	When switching the FIXEDCOLUMNSIZES property to value "true" then internally the grid is arranged in a way that the area always determines its size out of the width specification of the COLUMN controls. The browser does not look into the column contents in order to try to optimise the size of the area - but always follows the width that you define.	Optional	true false
requiredheight	Minimum height of the control in pixels. Use this property to ensure a minimum height if the overall control's height is a percentage of the available space - i.e. if value of property HEIGHT is a percentage (e.g. 100%). Please note:You must not use FIXLAYOUT at the surrounding row container (ITR and ROWAREA). Otherwise: if the available space is less than the required height the end of the control is just cut off.	Optional	1 2 3 int-value
disablecolumnresizing	Flag that indicates if the user can change the width of the grid columns. Default is false.	Optional	true false

disablecolumnmoving	Flag that indicates if the user can change the order of grid columns. Default is false.	Optional	true false
tabindex	Index that defines the tab order of the control. Controls are selected in increasing index order and in source order to resolve duplicates.	Optional	-1 0 1 2 5 10 32767
Drag And Drop			
draginfoprop	Name of the row item property that passes back the line's "drag info". When using this attribute the grid lines can be dragged onto "drop targets" (e.g. DROPICON control). The dragged line is identified by its "drag info". Use any string/information applicable.	Optional	
Deprecated			
directselectevent	Use ONCLICKMETHOD and ONDBLCLICKMETHOD instead.	Optional	ondblclick onclick
directselectmethod	$en/popupwizard/njx_textgrid_attr_directselectmethod$	Optional	

COLUMN Properties

The COLUMN tag is the typical tag that is placed inside a TEXTGRID2 definition. The COLUMN definition defines a column with its binding to a property of the collection elements.

Basic			
name	Text that is displayed inside the control. Please do not specify the name when using the multi language management - but specify a "textid" instead.	Sometimes obligatory	
textid	Multi language dependent text that is displayed inside the control. The "textid" is translated into a corresponding string at runtime. Do not specify a "name" inside the control if specifying a "textid".	Sometimes obligatory	

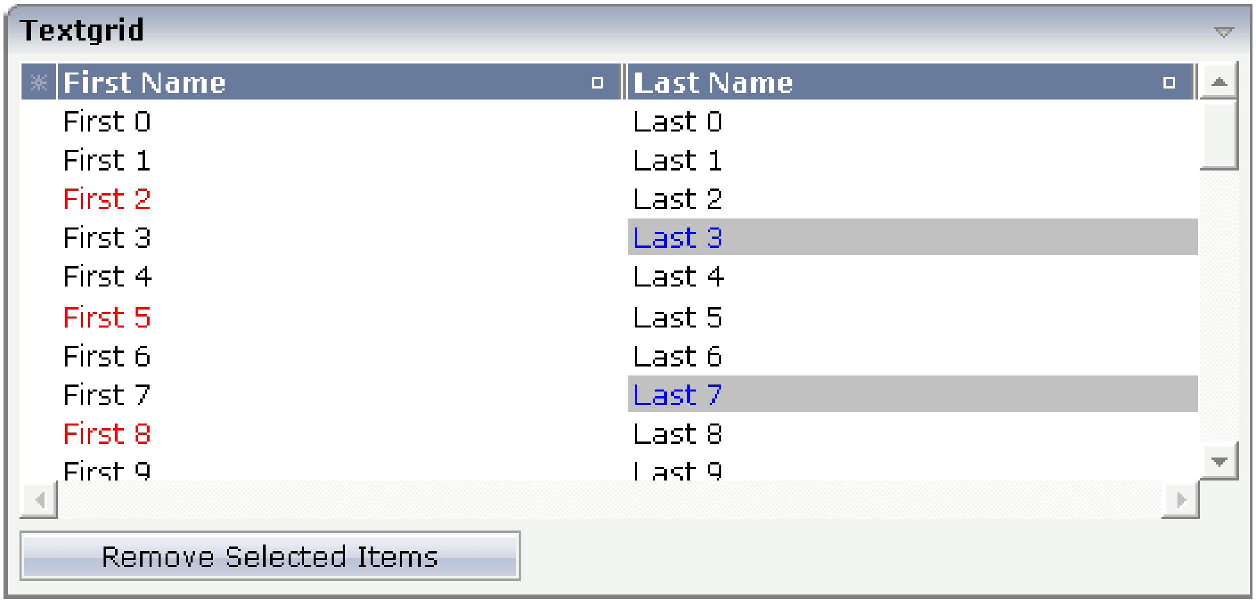
property	Property of the row item object that represents the column's content. The content typically is straight text but can also be "complex HTML".	Obligatory	
width	Width of the control. There are three possibilities to define the width: (A) You do not define a width at all. In this case the width of the control will either be a default width or - in case of container controls - it will follow the width that is occupied by its content. (B) Pixel sizing: just input a number value (e.g. "100"). (C) Percentage sizing: input a percantage value (e.g. "50%"). Pay attention: percentage sizing will only bring up correct results if the parent element of the control properly defines a width this control can reference. If you specify this control to have a width of 50% then the parent element (e.g. an ITR-row) may itself define a width of "100%". If the parent element does not specify a width then the rendering result may not represent what you expect.	Obligatory	100 120 140 160 180 200 50% 100%
comment	Comment without any effect on rendering and behaviour. The comment is shown in the layout editor's tree view.	Optional	
Appearance			
datatype	By default, the control is managing its content as string. By explicitly setting a datatype you can define that the control will format the data coming from the server: if the field has datatype "date" and the user inputs "010304" then the input will be translated into "01.03.2004" (or other representation, dependent on date format settings). Please note: the datatype "float" is named a bit misleading - it represents any decimal format number. The server side representation may be a float value, but also can be a double or a BigDecimal property.	Optional	date float int long time timestamp color ----------------------- N n.n P n.n string n xs:double xs:byte

			xs:short
align	Horizontal alignment of the control's content.	Optional	left center right
straighttext	If the text of the control contains HTML tags then these are by default interpreted by the browser. Specifiying STRAIGHTTEXT as "true" means that the browser will directly render the characters without HTML interpretation. Example: if you want to output the source of an HTML text then STRAIGHTTEXT should be set to "true". MOZILLA: this property is not available in Mozilla!	Optional	true false
convertspaces	If switched to "true" then all spaces inside the text that is rendered into the column are converted to non breakable spaces (andnbsp\"). Use this option if you have "meaningful" spaces inside the values you return from the server adapter object, e.g. if outputting some ASCII protocol inside a column.	Optional	true false
cuttextline	If switched to "false" then the content of the column is broken if it excceeds the column's width definition. Default is "true" i.e. if the content is too big for the column cell then it is cut.	Optional	true false
withsorticon	Flag that indicates if a small sort indicator is shown within the right corner of the control. Default is TRUE.	Optional	true false
headerimage	URL of image that is displayed inside the control. Any image type (.gif, .jpg, ...) that your browser does understand is valid. Use the following options to specify the URL: (A) Define the URL relative to your page. Your page is generated directly into your project's folder. Specifiying "images/xyz.gif" will point into a directory parallel to your page. Specifying "../HTMLBasedGUI/images/new.gif" will point to an image of a neighbour project. (B) Define a complete URL, like "http://www.softwareag.com/images/logo.gif".	Optional	
Binding			
property	(already explained above)		

textstyleprop	Name of the adapter parameter that provides a style-string that is used for rendering the column's content. As consequence you can indiviudally assign a CSS-style to each cell of your text grid.	Optional	
textclassprop	Name of the adapter parameter that provides a style class to be used for rendering the content. You can set up a limited number of style classes inside your style sheet definition - and dynamically reference them per grid cell.	Optional	
imageprop	Name of the adapter parameter that provides an image URL. The image is rendered at the very left of the column's area - in front of the text (PROPERTY property definition).	Optional	
linkmethod	Name of the event that is sent to the adapter if user clicks the column's text.	Optional	
celltitleprop	Name of the adapter parameter that provides the tooltip of this cell.	Optional	
Online help			
title	Text that is shown as tooltip for the control. Either specify the text "hard" by using this TITLE property - or use the TITLETEXTID in order to define a language dependent literal.	Optional	
titletextid	Text ID that is passed to the multi lanaguage management - representing the tooltip text that is used for the control.	Optional	
sorttitle	Text that is shown as tooltip for the sort indicator. Either input text by using this SORTTITLE property - or use the SORTTITLETEXTID in order to define a language dependent literal.	Optional	
sorttitletextid	Text ID that is passed to the multi lanaguage management - representing the tooltip text for the sort indicator.	Optional	
celltitleprop	(already explained above)		

Dynamic Setting of Text Styles in TEXTGRID2

The example from the previous sections will now be enhanced in order to demonstrate how to control the style of cells inside a TEXTGRID2 control dynamically:



First Name	Last Name
First 0	Last 0
First 1	Last 1
First 2	Last 2
First 3	Last 3
First 4	Last 4
First 5	Last 5
First 6	Last 6
First 7	Last 7
First 8	Last 8
First 9	Last 9

Some of the cells in the TEXTGRID2 control are rendered with a different style than the normal one. Each COLUMN definition has the property `textstyleprop`:

```
<rowarea name="Textgrid">
    <itr takefullwidth="true" fixlayout="true">
        <textgrid2 griddataprop="lines" width="100%" height="200"
selectprop="selected"
                   hscroll="true">
            <column name="First Name" property="firstName" width="50%"
                    textstyleprop="firstNameStyle">
            </column>
            <column name="Last Name" property="lastname" width="50%"
                    textstyleprop="lastNameStyle">
            </column>
        </textgrid2>
    </itr>
    <vdist height="5">
    </vdist>
    <itr>
        <button name="Remove Selected Items" method="onRemoveSelectedItems">
        </button>
    </itr>
</rowarea>
```

57 TEXTGRIDSSS2 - TEXTGRID2 with Server-Side Scrolling

The TEXTGRIDSSS2 control is a variant of the **TEXTGRID2** control which is explained in the previous section. "SSS" is the abbreviation for "server-side scrolling". What this means is described in this chapter.

This chapter covers the following topics:

Performance Considerations

The TEXTGRID2 control fetches all items belonging to the grid and renders them according to its layout definition. If there are more items available than the grid can display, a vertical scroll bar is displayed and you can scroll through the list.

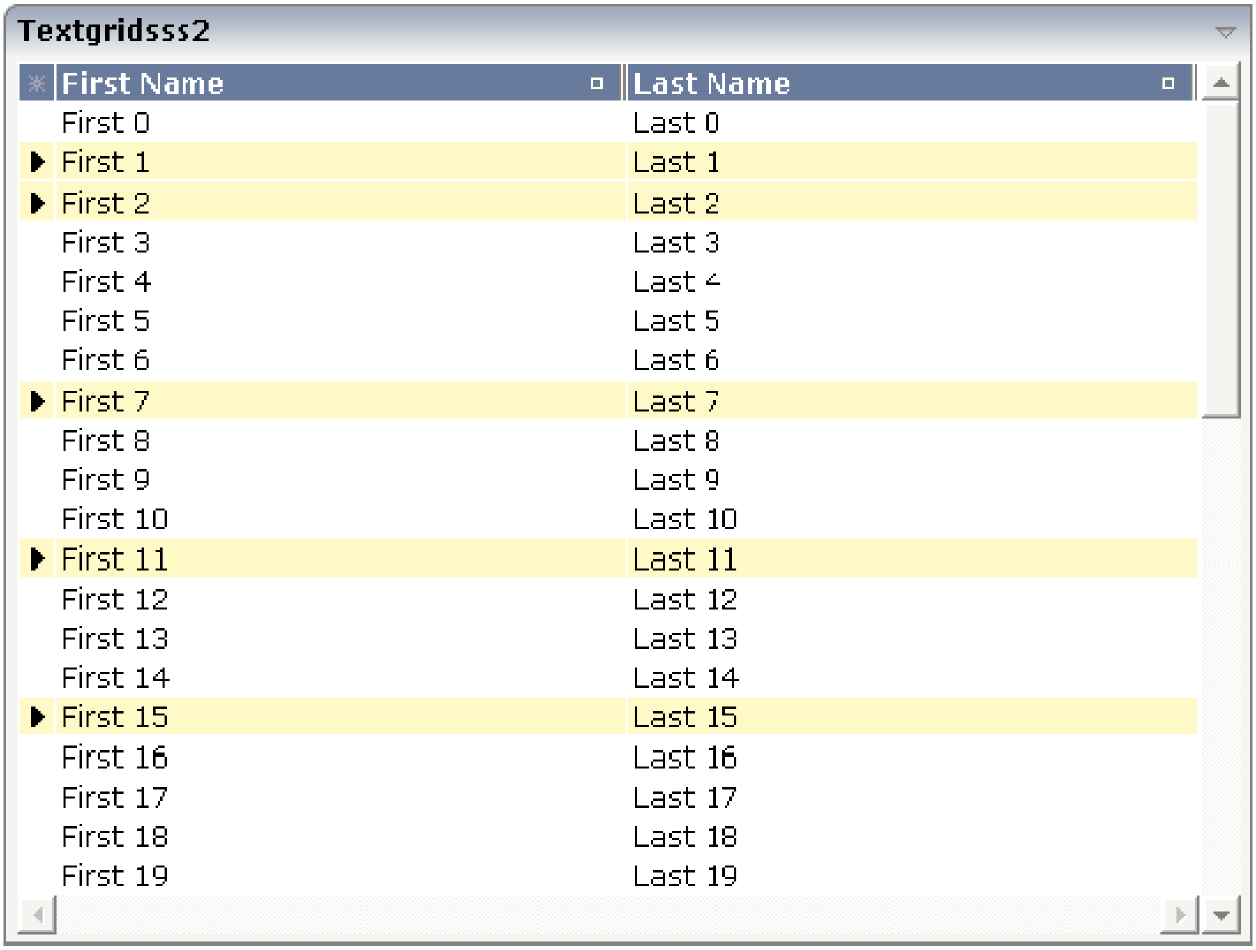
From scrolling perspective, this is very effective - the browser is very fast when scrolling is needed. But there are two disadvantages, especially for long lists:

- All the data that are to be displayed inside the grid must be available on the client side. Therefore, the data must be transferred from the server to the client at least one time. Imagine you have a grid of 10,000 lines: even if Application Designer transfers only 「net data」 and even if this happens in 「delta transfer mode」, it must be transferred.
- In addition, the grid must be built completely in order to allow fast scrolling. This means - taking the above example - that 10,000 lines have to be rendered before the grid can be displayed. Table rendering is time-consuming and needs a lot of the client's CPU performance.

Consequence: text grids of the TEXTGRID2 control are easy to use, but they have their limitations in terms of scalability. You should use it only if a limited amount of information is to be displayed.

Example

The TEXTGRIDSSS2 is very similar to the TEXTGRID2 control. However, some special behavior has been built in. The main differences are 「in the background」. The TEXTGRIDSSS2 control only receives the data of the visible items. In this example, only the data of the first 20 items are returned and rendered. When scrolling down, the next 20 items are fetched and rendered. This means: the control requests always the data which are currently displayed.



Textgridsss2

	First Name	Last Name
	First 0	Last 0
▶	First 1	Last 1
▶	First 2	Last 2
	First 3	Last 3
	First 4	Last 4
	First 5	Last 5
	First 6	Last 6
▶	First 7	Last 7
	First 8	Last 8
	First 9	Last 9
	First 10	Last 10
▶	First 11	Last 11
	First 12	Last 12
	First 13	Last 13
	First 14	Last 14
▶	First 15	Last 15
	First 16	Last 16
	First 17	Last 17
	First 18	Last 18
	First 19	Last 19

Consequence: every scrolling step requires an interaction with the server. However, only a small amount of data - which is visible - is requested, not the data of all available items. The performance of the grid does not change with the number of items which are available. There is no time difference in rendering a text grid containing 100 or 10,000 items.

The layout definition is:

```
<rowarea name="Textgridsss2">
    <itr>
        <textgridsss2 griddataprop="lines" rowcount="20" width="100%"
                      selectprop="selected" singleselect="false" hscroll="true"
                      directselectmethod="onDirectSelection"
                      directselectevent="ondblClick">
            <column name="First Name" property="firstname" width="50%">
            </column>
            <column name="Last Name" property="lastname" width="50%">
            </column>
        </textgridsss2>
    </itr>
</rowarea>
```

Adapter Interface

In the parameter data area of the adapter, the grid data is represented by the following data structure:

```
DEFINE DATA PARAMETER
1 LINES (1:*)
2 FIRSTNAME (U) DYNAMIC
2 LASTNAME (U) DYNAMIC
2 SELECTED (L)
1 LINESINFO
2 ROWCOUNT (I4)
2 SIZE (I4)
2 SORTPROPS (1:*)
3 ASCENDING (L)
3 PROPNAME (U) DYNAMIC
2 TOPINDEX (I4)
END-DEFINE
```

The parameters are nearly the same as for the TEXTGRID2 control. In addition, there is a LINESINFO structure. This structure is used to control the server-side scrolling and the server-side sorting.

Using Server-Side Scrolling

In the adapter parameters that represent the TEXTGRIDSSS2 control in the application, there are three parameters that control the server-side scrolling:

- TOPINDEX
- ROWCOUNT
- SIZE

In TOPINDEX and ROWCOUNT, the application receives the information how many items it should deliver to the page with the next scroll event and with which item the delivered amount should start.

In SIZE, the application returns the total number of items available. The client uses this information to set up the scroll bar correctly.

Using Server-Side Sorting

In the adapter parameters that represent the TEXTGRIDSSS2 control in the application, there is a substructure that controls the server-side sorting: `SORTPROPS`. With the information in this structure, the client tells the application by which sort criteria and in which order the client expects the items to be sorted.

TEXTGRIDSSS2 Properties

Basic			
griddataprop	Name of the adapter parameter that represents the grid in the adapter.	Obligatory	
rowcount	Number of rows that is renderes inside the control. There are two ways of using this property - dependent on whether you in addition define the HEIGHT property: If you do NOT define the HEIGHT property then the control is rendered with exactly the number of rows that is defined as ROWCOUNT value. If a HEIGHT value is defined an addition (e.g. as percentage value "100%") then the number of rows depends on the actual height of the control. The ROWCOUNT value in this case indicates the maximum number of rows that is picked from the server. You should define this value in a way that it is not too low - otherwise your grid will not be fully filled. On the other hand it should not be defined too high ("100") because this causes more communication traffic and more rendering effort inside the browser.	Obligatory	
width	Width of the control. There are three possibilities to define the width: (A) You do not define a width at all. In this case the width of the control will either be a default width or - in case of container controls - it will follow the width that is occupied by its content. (B) Pixel sizing: just input a number value (e.g. "100"). (C) Percentage sizing: input a percantage value (e.g. "50%"). Pay attention: percentage sizing will only bring	Obligatory	100 120 140 160 180 200 50%

	up correct results if the parent element of the control properly defines a width this control can reference. If you specify this control to have a width of 50% then the parent element (e.g. an ITR-row) may itself define a width of "100%". If the parent element does not specify a width then the rendering result may not represent what you expect.		100%
height	Height of the control. There are three possibilities to define the height: (A) You do not define a height at all. As consequence the control will be rendered with its default height. If the control is a container control (containing) other controls then the height of the control will follow the height of its content. (B) Pixel sizing: just input a number value (e.g. "20"). (C) Percentage sizing: input a percantage value (e.g. "50%"). Pay attention: percentage sizing will only bring up correct results if the parent element of the control properly defines a height this control can reference. If you specify this control to have a height of 50% then the parent element (e.g. an ITR-row) may itself define a height of "100%". If the parent element does not specify a width then the rendering result may not represent what you expect.	Optional	100 150 200 250 300 250 400 50% 100%
comment	Comment without any effect on rendering and behaviour. The comment is shown in the layout editor's tree view.	Optional	
Selection			
selectprop	Name of the adapter parameter that is used to mark if an individual row of the text grid is selected. If the user selects a text grid row, the value "true" is passed into the corresponding array element of the adapter parameter.	Optional	
singleselect	If set to "true" then only one row can be selected inside the text grid. - If set to "false" then multiple lines can be selected by using Ctrl- and Shift-key during mouse selection. Default is "false".	Optional	true false
singleselectprop	Name of an adapter parameter that dynamically defines whether SINGLESELECT is true or false.	Optional	
onclickmethod	Name of the event that is sent to the adapter when the user selects a row.	Optional	

	In the event handler you can find the selected rows by iterating through the rows and finding out which one's selected element is set to "true".		
ondblclickmethod	Name of the event that is sent to the adapter when the user selects a row by a double click. In the event handler you can find the selected rows by iterating through the rows and finding out which one's selected element is set to "true".	Optional	
withselectioncolumn	When defining a SELECTPROP property then automatically a selection column is added as first left column of the grid. Inside the column an icon inidicates if a row is currently selected. Set this property to "false" in order to avoid the selection column.	Optional	true false
withselectioncolumnicon	Flag that indicates whether the selection column shows a "select all" icon on top. Default is true.	Optional	true false
fgselect	if switched to true then an additional "graying" of selected lines will be activated. Switch this property to "true" if you have coloured textgrid cells: the selection colour will not override the colour of each cell, as consequence you require an additional effect in order to make the user see which row is selected.	Optional	true false
focusedprop	Name of an adapter parameter that is used to mark if an individual row of the text grid should receive the focus. If the user selects a text grid row, the value "true" is passed into the corresponding array element of the adapter parameter.	Optional	
Right Mouse Button			
oncontextmenumethod	Name of the event that is sent to the adapter when the user clicks with the right mouse button onto an empty area of the grid.	Optional	
singleselectcontextmenu	With SHIFT and CTRL key the user can select multiple lines (use property SINGLESELECT to suppress this feature). Use this property to ensure that the context menu is requested only for a single line. Default is "false".	Optional	true false
enabledefaultcontextmenu	Use this property to enable the default context menu of the browser within the textgrid. Please note: do not enable the browser's context menu if your application itself provides for a context menu. Default is "false".	Optional	true false

Appearance			
width	(already explained above)		
height	(already explained above)		
hscroll	Indicates if to show a horizontal scrollbar ("true") or not ("false"). If no scrollbar is shown then the control occupies the horizontal space that is required by its content.	Optional	true false
vscroll	Definition of the vertical scrollbar's appearance. You can define that scrollbars only are shown if the content is exceeding the control's area ("auto"). Or scrollbars can be shown always ("scroll"). Or scrollbars are never shown - and the content is cut ("hidden"). Default is "auto".	Optional	auto scroll hidden
touchpadinput	Boolean property that decides if touch pad support is offered for the TEXTGRID control. The default is "false". If switched to "true" then you can scroll the grid via a touch pad. As consequence you can use this control for making inputs through a touch terminal.	Optional	true false
withtitlerow	If defined as "false" then no top title row is shown. "True" is default.	Optional	true false
colspan	Column spanning of control. If you use TR table rows then you may sometimes want to control the number of columns your control occupies. By default it is "1" - but you may want to define the control to span over more than one columns. The property only makes sense in table rows that are snychronized within one container (i.e. TR, STR table rows). It does not make sense in ITR rows, because these rows are explicitly not synched.	Optional	1 2 3 4 5 50 int-value
rowspan	Row spanning of control. If you use TR table rows then you may sometimes want to control the number of rows your control occupies. By default it is "1" - but you may want to define the control two span over more than one columns. The property only makes sense in table rows that are snychronized within one container (i.e. TR, STR table rows). It does not make sense in ITR rows, because these rows are explicitly not synched.	Optional	1 2 3 4 5 50

			int-value
personalizable	If defined to "false" then no re-arranging of columns is offered to the user. Default is "true". This means: if using COLUMN controls inside the grid definition then the user can re-arrange the sequence of columns by dragging and dropping them within the top title row.	Optional	true false
stylevariant	Some controls offer the possibility to define style variants. By this style variant you can address different styles inside your style sheet definition file (.css). If not defined "normal" styles are chosen, if defined (e.g. "VAR1") then other style definitions (xxxVAR1xxx) are chosen. Purpose: you can set up style variants in the style sheet defintion and use them multiple times by addressing them via the "stylevariant" property. CIS currently offerst two variants "VAR1" and "VAR2" but does not predefine any semantics behind - this is up to you!	Optional	VAR1 VAR2
backgroundstyle	CSS style definition that is directly passed into this control. With the style you can individually influence the rendering of the control. You can specify any style sheet expressions. Examples are: border: 1px solid #FF0000 background-color: #808080 You can combine expressions by appending and separating them with a semicolon. Sometimes it is useful to have a look into the generated HTML code in order to know where direct style definitions are applied. Press right mouse-button in your browser and select the "View source" or "View frame's source" function.	Optional	
withblockscrolling	If switched to "true" then the grid will show small scroll icons by which the user can scroll the grid's content. Scrolling typically is done by using the grid's scrollbar - the scroll icons that are switched on by this property are an additional possibility to scroll.	Optional	true false
withrollover	The textgrid controls provide for a so called "roll over" effect. The row that is currently below the mouse pointer is highlighted in a certain way. Use this property to disable the roll over effect (Default is TRUE).	Optional	true false
fixedcolumnsizes	When switching the FIXEDCOLUMNSIZES property to value "true" then internally the grid is arranged in a way	Optional	true

	that the area always determines its size out of the width specification of the COLUMN controls. The browser does not look into the column contents in order to try to optimise the size of the area - but always follows the width that you define.		false
requiredheight	Minimum height of the control in pixels. Use this property to ensure a minimum height if the overall control's height is a percentage of the available space - i.e. if value of property HEIGHT is a percentage (e.g. 100%). Please note:You must not use FIXLAYOUT at the surrounding row container (ITR and ROWAREA). Otherwise: if the available space is less than the required height the end of the control is just cut off.	Optional	1 2 3 int-value
minapparentrows	Minimum number of apparent rows. Insert a valid number to make sure that (e.g. 10) rows are shown for sure.	Optional	1 2 3 int-value
disablecolumnresizing	Flag that indicates if the user can change the width of the grid columns. Default is false.	Optional	true false
disablecolumnmoving	Flag that indicates if the user can change the order of grid columns. Default is false.	Optional	true false
tabindex	Index that defines the tab order of the control. Controls are selected in increasing index order and in source order to resolve duplicates.	Optional	-1 0 1 2 5 10 32767
showemptylines	If set to false, no empty line will be rendered. By default empty lines are shown.	Optional	true false
Drag And Drop			
draginfoprop	Name of the row item property that passes back the line's "drag info". When using this attribute the grid lines can	Optional	

	be dragged onto "drop targets" (e.g. DROPICON control). The dragged line is identified by its "drag info". Use any string/information applicable.		
Deprecated			
directselectmethod	$en/popupwizard/njx_textgrid_attr_directselectmethod$	Optional	
directselectevent	Use ONCLICKMETHOD and ONDBLCLICKMETHOD instead.	Optional	ondblclick onclick

Inside the TEXTGRIDSSS2 definitions, COLUMN tags are also used to define its content. There is no difference in COLUMN tag usage between TEXTGRIDSSS2 and TEXTGRID2 definition.

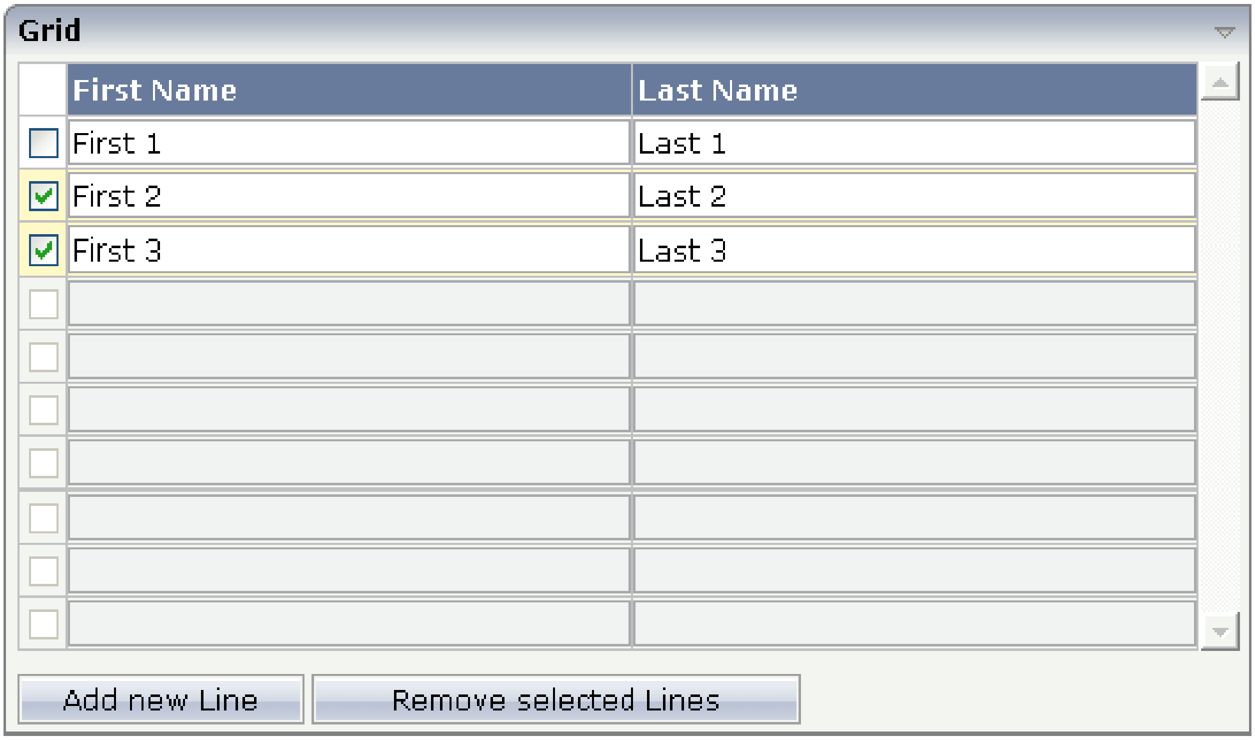
58 ROWTABLEAREA2 - The Flexible Control Grid

The ROWTABLEAREA2 is a container control that allows other controls to be arranged inside its grid management.

This chapter covers the following topics:

Example

There is a grid that contains a header row and 10 lines. Each line contains one check box and two fields. Some of the lines are highlighted.



	First Name	Last Name
☐	First 1	Last 1
☑	First 2	Last 2
☑	First 3	Last 3
☐		
☐		
☐		
☐		
☐		
☐		
☐		

The XML layout definition is:

```
<rowarea name="Grid">
    <rowtablearea2 griddataprop="lines" rowcount="10" width="100%" withborder="true">
         <tr>
             <hdist>
             </hdist>
             <label name="First Name" asheadline="true">
             </label>
             <label name="Last Name" asheadline="true">
             </label>
         </tr>
```

```
        <repeat>
            <str valueprop="selected">
                <checkbox valueprop="selected" flush="screen" width="30">
                </checkbox>
                <field valueprop="firstname" width="50%">
                </field>
                <field valueprop="lastname" width="50%">
                </field>
            </str>
        </repeat>
    </rowtablearea2>
    <vdist height="10">
    </vdist>
    <itr>
        <button name="Add new Line" method="onAddLine">
        </button>
        <hdist>
        </hdist>
        <button name="Remove selected Lines" method="onRemoveLines">
        </button>
    </itr>
</rowarea>
```

Note the following:

- There is a ROWTABLEAREA2 definition with the property `griddataprop="lines"`. There is a `rowcount` definition of "10". This is the same as for the text grid processing: the grid container is bound to a server-side collection. Similar to the TEXTGRIDSSS2 definition, there is a row count that defines the number of lines.
- Inside the ROWTABLEAREA2 definition, there is first the definition of a normal table row (TR) in which a distance and two labels are defined. The labels are rendered with `asheadline="true"`.
- Inside the REPEAT definition, there is a special table row definition "STR" (selectable table row) that itself contains one CHECKBOX and two FIELD definitions. CHECKBOX and FIELDs are bound to properties themselves.
- After the ROWTABLEAREA2 definition, there is a vertical distance and a row that contains two buttons with which a user can manipulate the grid.

The content of the REPEAT block is repeated as many times as defined inside the `rowcount` definition of ROWTABLEAREA2. The content holds a table row (STR) - therefore the result is a grid.

Adapter Interface

In the parameter data area of the adapter, the grid data is represented by the following data structure:

```
DEFINE DATA PARAMETER
1 LINES (1:*)
2 FIRSTNAME (U) DYNAMIC
2 LASTNAME (U) DYNAMIC
2 SELECTED (L)
END-DEFINE
```

Making Grids Look like Grids

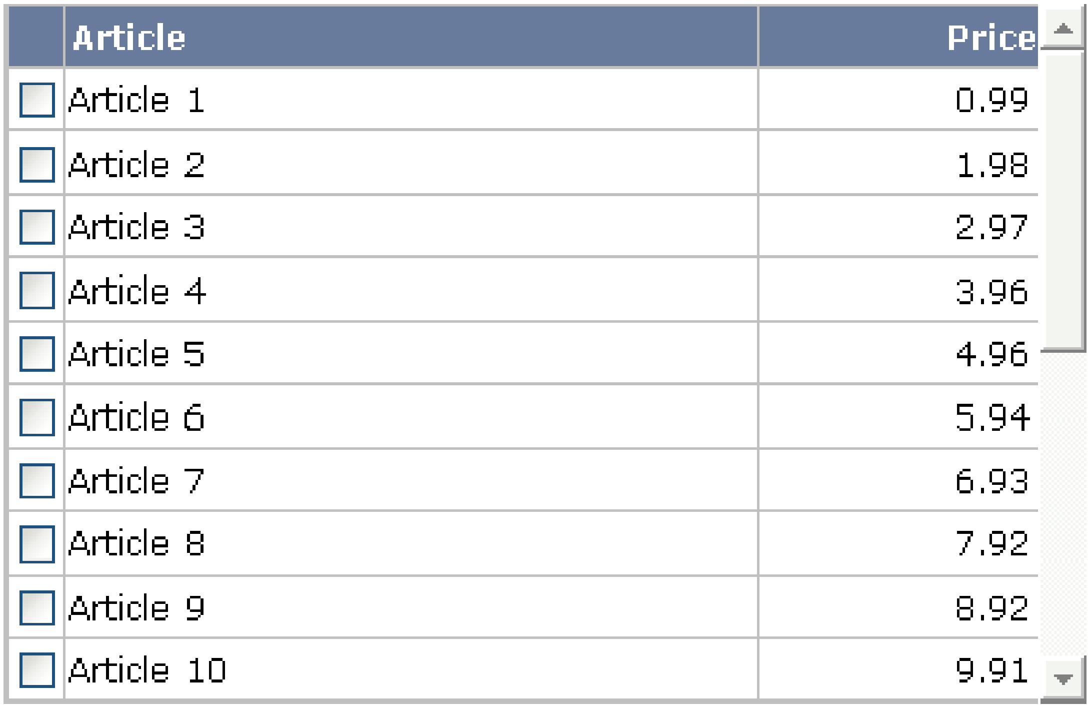
Fields typically contain a high number of FIELD controls. Typically, a FIELD control has a certain rendering that renders a field with a border and with a certain background color.

Be aware that inside the FIELD definition, there are two important properties:

- `noborder` - if set to "true", no border will be drawn
- `transparentbackground` - if set to "true", the field will always take over the background of the controls in which it is positioned (e.g. STR row).

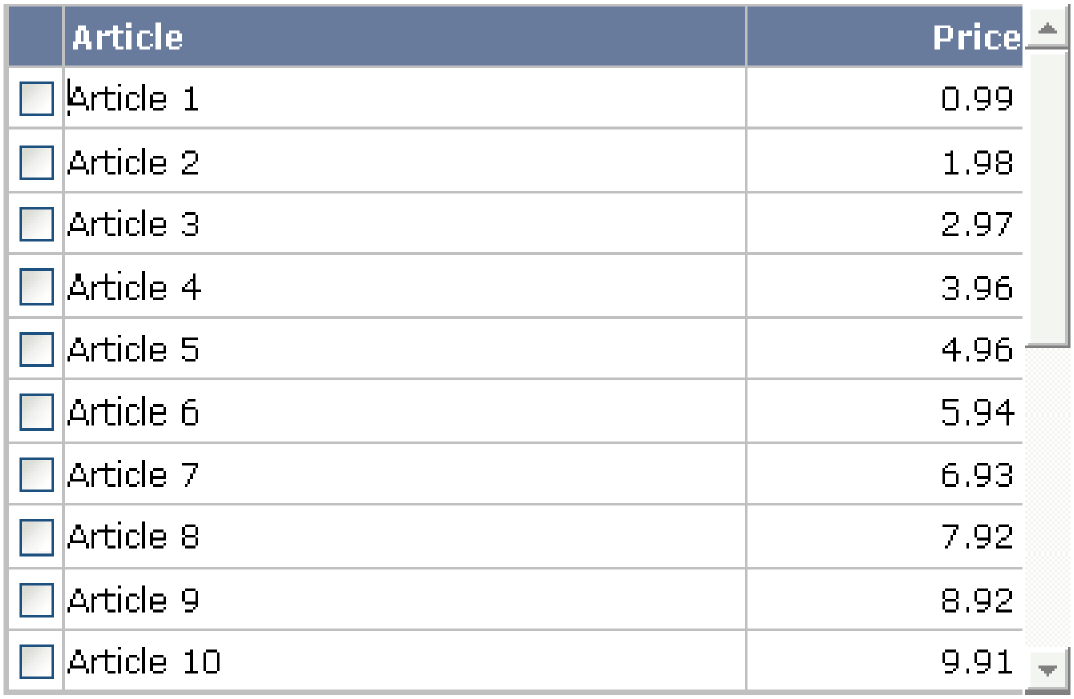
Have a look at the difference between the following screens. One screen uses the properties, the other screen does not use them.

This is a grid:



	Article	Price
□	Article 1	0.99
□	Article 2	1.98
□	Article 3	2.97
□	Article 4	3.96
□	Article 5	4.96
□	Article 6	5.94
□	Article 7	6.93
□	Article 8	7.92
□	Article 9	8.92
□	Article 10	9.91

This is collection of fields:



	Article	Price
□	Article 1	0.99
□	Article 2	1.98
□	Article 3	2.97
□	Article 4	3.96
□	Article 5	4.96
□	Article 6	5.94
□	Article 7	6.93
□	Article 8	7.92
□	Article 9	8.92
□	Article 10	9.91

ROWTABLEAREA2 Properties

Basic			
griddataprop	Name of the adapter parameter that represents the control in the adapter.	Obligatory	
rowcount	Number of rows that is renderes inside the control. There are two ways of using this property - dependent on whether you in addition define the HEIGHT property: If you do NOT define the HEIGHT property then the control is rendered with exactly the number of rows that is defined as ROWCOUNT value. If a HEIGHT value is defined an addition (e.g. as percentage value "100%") then the number of rows depends on the actual height of the control. The ROWCOUNT value in this case indicates the maximum number of rows that is picked from the server. You should define this value in a way that it is not too low - otherwise your grid will not be fully filled. On the other hand it should not be defined too high ("100") because this causes more communication traffic and more rendering effort inside the browser.	Optional	
height	Height of the control. There are three possibilities to define the height: (A) You do not define a height at all. As consequence the control will be rendered with its default height. If the control is a container control (containing) other controls then the height of the control will follow the height of its content. (B) Pixel sizing: just input a number value (e.g. "20"). (C) Percentage sizing: input a percantage value (e.g. "50%"). Pay attention: percentage sizing will only bring up correct results if the parent element of the control properly defines a height this control can reference. If you specify this control to have a height of 50% then the parent element (e.g. an ITR-row) may itself define a height of "100%". If the parent element does not specify a width then	Optional	100 150 200 250 300 250 400 50% 100%

	the rendering result may not represent what you expect.		
width	Width of the control. There are three possibilities to define the width: (A) You do not define a width at all. In this case the width of the control will either be a default width or - in case of container controls - it will follow the width that is occupied by its content. (B) Pixel sizing: just input a number value (e.g. "100"). (C) Percentage sizing: input a percantage value (e.g. "50%"). Pay attention: percentage sizing will only bring up correct results if the parent element of the control properly defines a width this control can reference. If you specify this control to have a width of 50% then the parent element (e.g. an ITR-row) may itself define a width of "100%". If the parent element does not specify a width then the rendering result may not represent what you expect.	Sometimes obligatory	100 120 140 160 180 200 50% 100%
firstrowcolwidths	If set to "true" then the grid is sized according to its first row. This first row typically is a header-TR-row in which GRIDCOLHEADER controls are used as column headers for the subsequent rows. Default is "false", i.e. the grid is sized according to its "whole content". Please note: when using the GRIDCOLHEADER control within the header-TR-row this property must be set to "true" - otherwise column resizing (by drag and drop) does not work correctly.	Sometimes obligatory	true false
comment	Comment without any effect on rendering and behaviour. The comment is shown in the layout editor's tree view.	Optional	
Appearance			
withborder	If set to "false" then no thin border is drawn around the controls that are contained in the grid. Default is "true".	Optional	true false
hscroll	Indicates if to show a horizontal scrollbar ("true") or not ("false").	Optional	true false

	If no scrollbar is shown then the control occupies the horizontal space that is required by its content.		
vscroll	Definition of the vertical scrollbar's appearance. You can define that scrollbars only are shown if the content is exceeding the control's area ("auto"). Or scrollbars can be shown always ("scroll"). Or scrollbars are never shown - and the content is cut ("hidden"). Default is "auto".	Optional	auto scroll hidden
firstrowcolwidths	(already explained above)		
clipboardaccess	If switched to true then the content of the grid can be selected and exported into the client's clipboard.	Optional	true false
withblockscrolling	If switched to "true" then the grid will show small scroll icons by which the user can scroll the grid's content. Scrolling typically is done by using the grid's scrollbar - the scroll icons that are switched on by this property are an additional possibility to scroll.	Optional	true false
touchpadinput	If set to "true" then touch screen icons for scrolling are displayed in addition. Default is "false".	Optional	true false
requiredheight	Minimum height of the control in pixels. Use this property to ensure a minimum height if the overall control's height is a percentage of the available space - i.e. if value of property HEIGHT is a percentage (e.g. 100%). Please note:You must not use FIXLAYOUT at the surrounding row container (ITR and ROWAREA). Otherwise: if the available space is less than the required height the end of the control is just cut off.	Optional	1 2 3 int-value
tablestyle	CSS style definition that is directly passed into this control. With the style you can individually influence the rendering of the control. You can specify any style sheet expressions. Examples are: border: 1px solid #FF0000 background-color: #808080	Optional	background-color: #FF0000 color: #0000FF font-weight: bold

	You can combine expressions by appending and separating them with a semicolon. Sometimes it is useful to have a look into the generated HTML code in order to know where direct style definitions are applied. Press right mouse-button in your browser and select the "View source" or "View frame's source" function.		
darkbackground	Normally the background is in light colour but the CIS style sheets also have a dark(er) grey colour to be used. If DARKBACKGROUND is set to true then the darker background colour is chosen. This property typically is used to integrate light coloured controls into darker container areas.	Optional	true false
Binding			
oncontextmenumethod	Name of the event that is sent to the adapter when the user presses the right mouse button in the grid, but not on an existing row, but in an empty area of the grid.	Optional	
fwdtabkeymethod	Name of the event that is sent to the adapter when the user presses the TAB key within the very last cell of the grid (last cell within the last line). Use property FWDTABKEYFILTER to associate this call with a grid column.	Optional	
fwdtabkeyfilter	By default the FWDTABKEYMETHOD is called if the user presses the TAB key within the veryfirst cell of the grid. Input the name of a cell's VALUEPROP to associate the method call with any other column.	Optional	
bwdtabkeymethod	Name of the event that is sent to the adapter when the user presses SHIFT and TAB keys within the first cell of a grid line. Use property BWDTABKEYFILTER to associate this call with a cell of choice.	Optional	
bwdtabkeyfilter	By default the BWDTABKEYMETHOD is called if the user presses the SHIFT and TAB keys within the very first cell of the grid. Input the name of a cell's VALUEPROP to associate the method call with any other column.	Optional	
Hot Keys			
hotkeys	Comma separated list of hot keys. A hotkey consists of a list of keys and a method name. Separate the keys by "-" and the method name again with a comma	Optional	

	Example: ctrl-alt-65;onCtrlAltA;13;onEnter ...defines two hot keys. Method onCtrlAltA is invoked if the user presses Ctrl-Alt-A. Method "onEnter" is called if the user presses the ENTER key. Use the popup help within the Layout Painter to input hot keys.		

STR Properties

STR (selectable table row) is a normal table row (TR) that highlights its background depending on an adapter property.

Basic			
valueprop	Name of the adapter parameter that defines if the row is selected or not.	Obligatory	
withalterbackground	Flag that indicates if the grid line shows alternating background color (like rows within a textgrids). Default is false. Please note: controls inside the row must have transparent background. In case of the FIELD control simply set property TRANSPARENTBACKGROUND to true.	Optional	true false
showifempty	Flag that indicates if an unused row is visible. Example: if set to false a grid with rowcount ten and a server side collection size of seven will hide the three remaining rows. Default is false.	Optional	true false
comment	Comment without any effect on rendering and behaviour. The comment is shown in the layout editor's tree view.	Optional	
Binding			
valueprop	(already explained above)		
onclickmethod	Name of the event that is sent to the adapter when the user clicks a line.	Optional	
ondblclickmethod	Name of the event that is sent to the adapter when the user double clicks a line.	Optional	
proprefprop	Name of the adapter parameter that is filled when the user clicks a FIELD control. The VALUEPROP of the clicked field control will passed.	Optional	

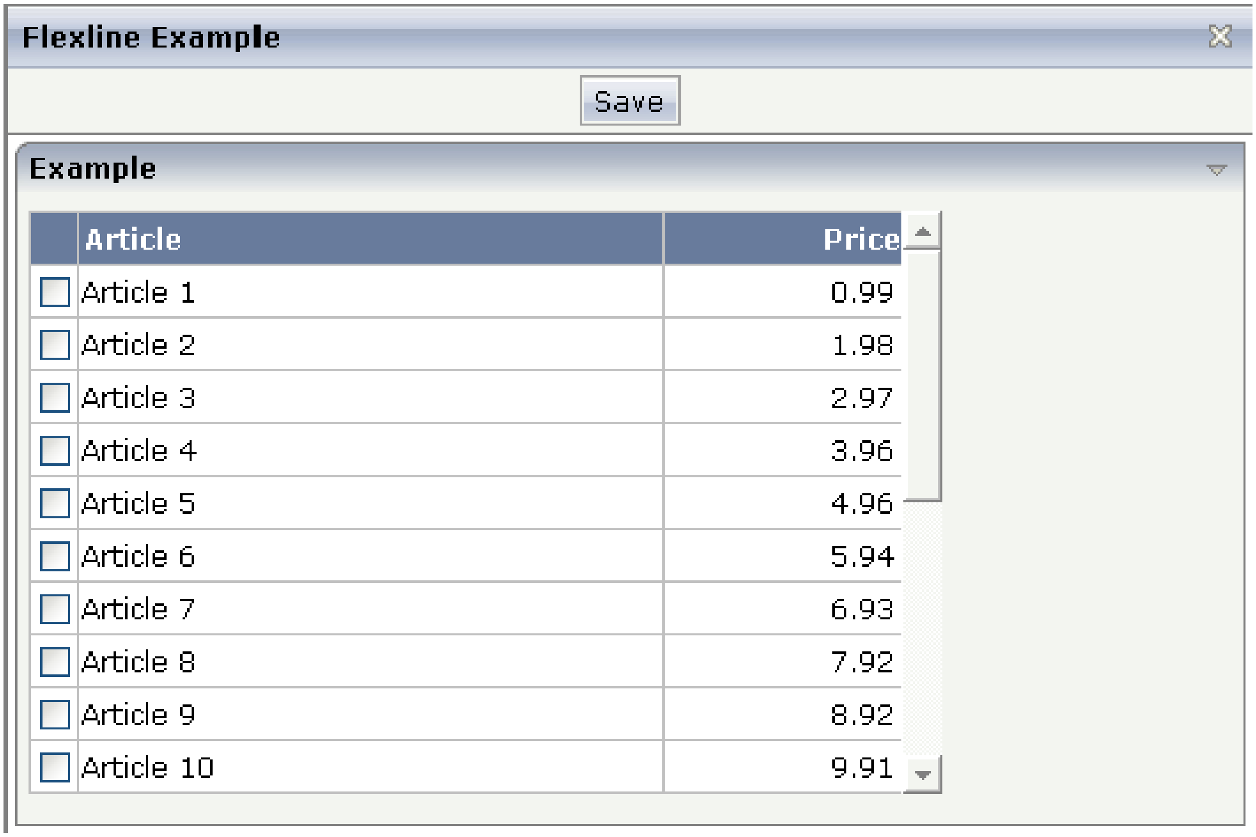
59 FLEXLINE - Flexible Columns in Control Grids

In a **previous** example, the grid was completely defined as part of the layout definition: the sequence of columns was internally defined by defining the controls that are part of an STR row.

This chapter covers the following topics:

Example

Have a look at the following example:



	Article	Price
□	Article 1	0.99
□	Article 2	1.98
□	Article 3	2.97
□	Article 4	3.96
□	Article 5	4.96
□	Article 6	5.94
□	Article 7	6.93
□	Article 8	7.92
□	Article 9	8.92
□	Article 10	9.91

The grid looks like a normal ROWTABLEAREA2 grid, but it is built in a more dynamic way.

The XML layout definition is:

```
    <pagebody>
        <rowarea name="Example">
            <vdist height="5">
            </vdist>
            <rowtablearea2 griddataprop="lines" rowcount="10" width="395"
withborder="true">
                <tr>
                    <label name=" " asheadline="true">
                    </label>
                    <flexline infoprop="headline">
                    </flexline>
                </tr>
                <repeat>
                    <str valueprop="selected">
                        <checkbox valueprop="selected" flush="screen" width="30">
                        </checkbox>
                        <flexline infoprop="/rowline">
                        </flexline>
                        <hdist width="100%">
                        </hdist>
                    </str>
                </repeat>
            </rowtablearea2>
            <vdist height="10">
            </vdist>
        </rowarea>
        <vdist height="5">
        </vdist>
    </pagebody>
```

You see that there are two FLEXLINE control definitions inside the ROWTABLEAREA2 definition:

- One definition represents the headline of the grid.
- The other definition is part of each row's content.

Each definition points to a property that passes the configuration at runtime. Within the second definition, you may see something which is new for you: the VALUEPROP references to a property `/rowline`. The "/" character at the beginning indicates that this property is dynamically controlled by the application through an adapter parameter.

Adapter Interface

In the parameter data area of the adapter, the grid data is represented by the following data structure:

```
DEFINE DATA PARAMETER
1 HEADLINE (1:*)
2 ATTRIBUTES (U) DYNAMIC
2 CONTROL (U) DYNAMIC
1 LINES (1:*)
2 SELECTED (L)
1 ROWLINE (1:*)
2 ATTRIBUTES (U) DYNAMIC
2 CONTROL (U) DYNAMIC
END-DEFINE
```

FLEXLINE Properties

Basic			
infoprop	Name of the adapter parameter that represents the control in the adapter.	Obligatory	
withborder	Flag that indicates if a border is drawn between the controls that are rendered inside the FLEXLINE control. Default is "false", i.e. no border is drawn.	Optional	true false
comment	Comment without any effect on rendering and behaviour. The comment is shown in the layout editor's tree view.	Optional	

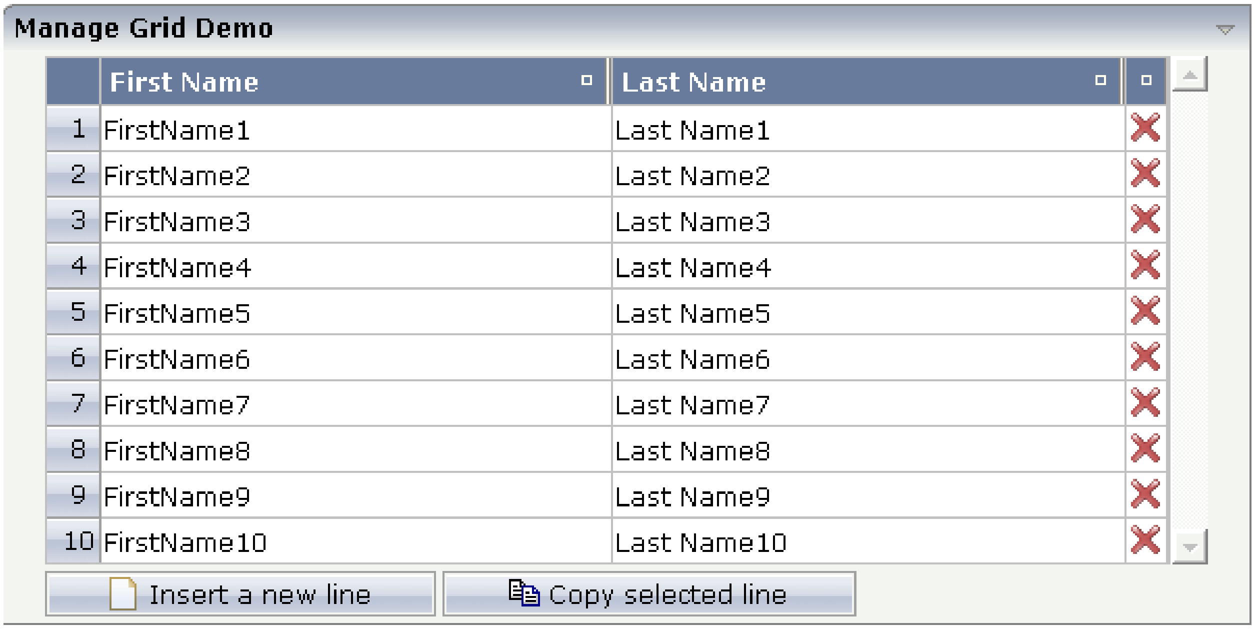
60 MGDGRID - Managing the Grid

The MGDGRID control is an extension of the ROWTABLEAREA2 control. It allows to insert, copy and delete rows of the grid.

This chapter covers the following topics:

See also ***STR Properties*** which are described with the ROWTABLEAREA2 control.

Example



	First Name	Last Name	
1	FirstName1	Last Name1	✗
2	FirstName2	Last Name2	✗
3	FirstName3	Last Name3	✗
4	FirstName4	Last Name4	✗
5	FirstName5	Last Name5	✗
6	FirstName6	Last Name6	✗
7	FirstName7	Last Name7	✗
8	FirstName8	Last Name8	✗
9	FirstName9	Last Name9	✗
10	FirstName10	Last Name10	✗

There is a grid that contains a header row and 10 lines. Each line contains two fields and a 「delete row」 control.

Each of the function controls (insert, copy, delete) can be added at the top of the MGDGRID, below the MGDGRID or within the lines of the MGDGRID.

Look at the corresponding layout definition:

```
<rowarea name="Manage Grid Demo">
  <mgdgrid griddataprop="mglines" rowcount="10" width="100%" firstrowcolwidths="true">
    <tr>
      <label name="  " width="25" asheadline="true">
      </label>
      <gridcolheader name="First Name" width="50%">
      </gridcolheader>
      <gridcolheader name="Last Name" width="50%" >
      </gridcolheader>
      <gridcolheader width="20">
      </gridcolheader>
      <hdist></hdist>
    </tr>
    <repeat>
      <str valueprop="selected" showifempty="true">
        <selector valueprop="selected" singleselect="true">
        </selector>
        <field valueprop="fname" width="100%">
        </field>
        <field valueprop="lname" width="100%">
        </field>
        <rowdelete>
        </rowdelete>
      </str>
    </repeat>
    <mgdfunctions>
      <rowinsert title="Insert a new line">
      </rowinsert>
      <rowcopy title="Copy selected line">
      </rowcopy>
    </mgdfunctions>
  </mgdgrid>
</rowarea>
```

The MGDGRID control is an extension to the ROWTABLEAREA2 control. See the description of the **ROWTABLEAREA2** control for further information.

Adapter Interface

In the parameter data area of the adapter, the grid data is represented by the following data structure:

```
DEFINE DATA PARAMETER
1 MGLINES (1:*)
2 FNAME (U) DYNAMIC
2 LNAME (U) DYNAMIC
2 SELECTED (L)
END-DEFINE
```

MGDGRID Properties

Basic			
griddataprop	Name of the adapter parameter that represents the control in the adapter.	Obligatory	
rowcount	Number of rows that is renderes inside the control. There are two ways of using this property - dependent on whether you in addition define the HEIGHT property:	Optional	

	If you do NOT define the HEIGHT property then the control is rendered with exactly the number of rows that is defined as ROWCOUNT value. If a HEIGHT value is defined an addition (e.g. as percentage value "100%") then the number of rows depends on the actual height of the control. The ROWCOUNT value in this case indicates the maximum number of rows that is picked from the server. You should define this value in a way that it is not too low - otherwise your grid will not be fully filled. On the other hand it should not be defined too high ("100") because this causes more communication traffic and more rendering effort inside the browser.		
height	Height of the control. There are three possibilities to define the height: (A) You do not define a height at all. As consequence the control will be rendered with its default height. If the control is a container control (containing) other controls then the height of the control will follow the height of its content. (B) Pixel sizing: just input a number value (e.g. "20"). (C) Percentage sizing: input a percantage value (e.g. "50%"). Pay attention: percentage sizing will only bring up correct results if the parent element of the control properly defines a height this control can reference. If you specify this control to have a height of 50% then the parent element (e.g. an ITR-row) may itself define a height of "100%". If the parent element does not specify a width then the rendering result may not represent what you expect.	Optional	100 150 200 250 300 250 400 50% 100%
width	Width of the control. There are three possibilities to define the width: (A) You do not define a width at all. In this case the width of the control will either be a default width or - in case of container controls - it will follow the width that is occupied by its content. (B) Pixel sizing: just input a number value (e.g. "100").	Sometimes obligatory	100 120 140 160 180 200 50%

	(C) Percentage sizing: input a percantage value (e.g. "50%"). Pay attention: percentage sizing will only bring up correct results if the parent element of the control properly defines a width this control can reference. If you specify this control to have a width of 50% then the parent element (e.g. an ITR-row) may itself define a width of "100%". If the parent element does not specify a width then the rendering result may not represent what you expect.		100%
firstrowcolwidths	If set to "true" then the grid is sized according to its first row. This first row typically is a header-TR-row in which GRIDCOLHEADER controls are used as column headers for the subsequent rows. Default is "false", i.e. the grid is sized according to its "whole content". Please note: when using the GRIDCOLHEADER control within the header-TR-row this property must be set to "true" - otherwise column resizing (by drag and drop) does not work correctly.	Sometimes obligatory	true false
comment	Comment without any effect on rendering and behaviour. The comment is shown in the layout editor's tree view.	Optional	
Appearance			
withborder	If set to "false" then no thin border is drawn around the controls that are contained in the grid. Default is "true".	Optional	true false
hscroll	Indicates if to show a horizontal scrollbar ("true") or not ("false"). If no scrollbar is shown then the control occupies the horizontal space that is required by its content.	Optional	true false
vscroll	Definition of the vertical scrollbar's appearance. You can define that scrollbars only are shown if the content is exceeding the control's area ("auto"). Or scrollbars can be shown always ("scroll"). Or scrollbars are never shown - and the content is cut ("hidden"). Default is "auto".	Optional	auto scroll hidden
firstrowcolwidths	(already explained above)		
clipboardaccess	If switched to true then the content of the grid can be selected and exported into the client's clipboard.	Optional	true

			false
withblockscrolling	If switched to "true" then the grid will show small scroll icons by which the user can scroll the grid's content. Scrolling typically is done by using the grid's scrollbar - the scroll icons that are switched on by this property are an additional possibility to scroll.	Optional	true false
touchpadinput	If set to "true" then touch screen icons for scrolling are displayed in addition. Default is "false".	Optional	true false
requiredheight	Minimum height of the control in pixels. Use this property to ensure a minimum height if the overall control's height is a percentage of the available space - i.e. if value of property HEIGHT is a percentage (e.g. 100%). Please note:You must not use FIXLAYOUT at the surrounding row container (ITR and ROWAREA). Otherwise: if the available space is less than the required height the end of the control is just cut off.	Optional	1 2 3 int-value
tablestyle	CSS style definition that is directly passed into this control. With the style you can individually influence the rendering of the control. You can specify any style sheet expressions. Examples are: border: 1px solid #FF0000 background-color: #808080 You can combine expressions by appending and separating them with a semicolon. Sometimes it is useful to have a look into the generated HTML code in order to know where direct style definitions are applied. Press right mouse-button in your browser and select the "View source" or "View frame's source" function.	Optional	background-color: #FF0000 color: #0000FF font-weight: bold
Binding			
oncontextmenumethod	Name of the event that is sent to the adapter when the user presses the right mouse button in the grid, but not on an existing row, but in an empty area of the grid.	Optional	
fwdtabkeymethod	Name of the event that is sent to the adapter when the user presses the TAB key within the very last	Optional	

	cell of the grid (last cell within the last line). Use property FWDTABKEYFILTER to associate this call with a grid column.		
fwdtabkeyfilter	By default the FWDTABKEYMETHOD is called if the user presses the TAB key within the veryfirst cell of the grid. Input the name of a cell's VALUEPROP to associate the method call with any other column.	Optional	
bwdtabkeymethod	Name of the event that is sent to the adapter when the user presses SHIFT and TAB keys within the first cell of a grid line. Use property BWDTABKEYFILTER to associate this call with a cell of choice.	Optional	
bwdtabkeyfilter	By default the BWDTABKEYMETHOD is called if the user presses the SHIFT and TAB keys within the very first cell of the grid. Input the name of a cell's VALUEPROP to associate the method call with any other column.	Optional	
Hot Keys			
hotkeys	Comma separated list of hot keys. A hotkey consists of a list of keys and a method name. Separate the keys by "-" and the method name again with a comma Example: ctrl-alt-65;onCtrlAltA;13;onEnter ...defines two hot keys. Method onCtrlAltA is invoked if the user presses Ctrl-Alt-A. Method "onEnter" is called if the user presses the ENTER key. Use the popup help within the Layout Painter to input hot keys.	Optional	

ROWINSERT Properties

Basic			
image	URL that points to the image that is shown as icon. The URL either is an absolute URL or a relative URL. If using a relative URL then be aware of that the generated page is located directly inside your project's directory.	Obligatory	

	Example: "images/icon.gif" points to an icon in an images-folder that is parallel to the page itself. "../HTMLBasedGUI/images/new.gif" point to a URL that is located inside a different project.		
comment	Comment without any effect on rendering and behaviour. The comment is shown in the layout editor's tree view.	Optional	
Binding			
visibleprop	Name of the adapter parameter that provides the information if this control is displayed or not. As consequence you can control the visibility of the control dynamically.	Optional	
Online Help			
title	Text that is shown as tooltip for the control. Either specify the text "hard" by using this TITLE property - or use the TITLETEXTID in order to define a language dependent literal.	Optional	
titletextid	Text ID that is passed to the multi lanaguage management - representing the tooltip text that is used for the control.	Optional	

ROWCOPY Properties

Basic			
image	URL that points to the image that is shown as icon. The URL either is an absolute URL or a relative URL. If using a relative URL then be aware of that the generated page is located directly inside your project's directory. Example: "images/icon.gif" points to an icon in an images-folder that is parallel to the page itself. "../HTMLBasedGUI/images/new.gif" point to a URL that is located inside a different project.	Obligatory	
comment	Comment without any effect on rendering and behaviour. The comment is shown in the layout editor's tree view.	Optional	
Binding			
visibleprop	Name of the adapter parameter that provides the information if this control is displayed or not. As consequence you can control the visibility of the control dynamically.	Optional	
Online Help			
title	Text that is shown as tooltip for the control. Either specify the text "hard" by using this TITLE property - or use the TITLETEXTID in order to define a language dependent literal.	Optional	
titletextid	Text ID that is passed to the multi lanaguage management - representing the tooltip text that is used for the control.	Optional	

ROWDELETE Properties

Basic			
image	URL that points to the image that is shown as icon. The URL either is an absolute URL or a relative URL. If using a relative URL then be aware of that the generated page is located directly inside your project's directory. Example: "images/icon.gif" points to an icon in an images-folder that is parallel to the page itself. "../HTMLBasedGUI/images/new.gif" point to a URL that is located inside a different project.	Obligatory	
comment	Comment without any effect on rendering and behaviour. The comment is shown in the layout editor's tree view.	Optional	
Binding			
visibleprop	Name of the adapter parameter that provides the information if this control is displayed or not. As consequence you can control the visibility of the control dynamically.	Optional	
Online Help			
title	Text that is shown as tooltip for the control. Either specify the text "hard" by using this TITLE property - or use the TITLETEXTID in order to define a language dependent literal.	Optional	
titletextid	Text ID that is passed to the multi lanaguage management - representing the tooltip text that is used for the control.	Optional	

61 Working with Trees

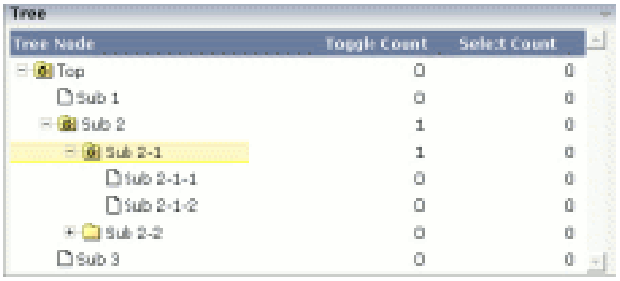
This part shows you how to work with trees and tree nodes. The information is organized under the following headings:

62 TREENODE3 in Control Grid (ROWTABLEAREA2)

This chapter covers the following topics:

Example

The following image shows an example for a tree management:



Tree

Tree Node	Toggle Count	Select Count
Top	0	0
Sub 1	0	0
Sub 2	1	0
Sub 2-1	1	0
Sub 2-1-1	0	0
Sub 2-1-2	0	0
Sub 2-2	0	0
Sub 3	0	0

The grid contains three columns: the first column shows the tree node, the other two columns display some text information.

The XML layout definition is:

```
<rowarea name="Tree">
    <rowtablearea2 griddataprop="treeGridInfo" rowcount="8" width="500"
withborder="false">
        <tr>
            <label name="Tree Node" width="200" asheadline="true">
            </label>
            <label name="Toggle Count" width="100" asheadline="true"
                   labelstyle="text-align:right">
            </label>
            <label name="Select Count" width="100" asheadline="true"
                   labelstyle="text-align:right">
            </label>
        </tr>
        <repeat>
            <tr>
                <treenode3 width="200" withplusminus="true"
                           imageopened="images/fileopened.gif"
                           imageclosed="images/fileclosed.gif"
                           imageendnode="images/fileendnode.gif">
                </treenode3>
                <textout valueprop="toggleCount" width="100" align="right">
                </textout>
                <textout valueprop="selectCount" width="100" align="right">
                </textout>
            </tr>
        </repeat>
    </rowtablearea2>
</rowarea>
```

You see that the TREENODE3 control is placed inside the control grid just as a normal control. There are certain properties available which influence the rendering: in the example, the name of the tree node images is statically overwritten. The flag `withplusminus` is set to true - consequently, small "+"/"-" icons are placed in front of the node.

Adapter Interface

In the parameter data area of the adapter, the tree data is represented by the following data structure:

```
DEFINE DATA PARAMETER
1 TREEGRIDINFO (1:*)
2 DRAGINFO (U) DYNAMIC
2 DROPINFO (U) DYNAMIC
2 LEVEL (I4)
2 OPENED (I4)
2 SELECTCOUNT (U) DYNAMIC
2 TEXT (U) DYNAMIC
2 TOGGLECOUNT (U) DYNAMIC
END-DEFINE
```

Built-in Events

`value-of-griddataprop`.reactOnSelect
`value-of-griddataprop`.reactOnToggle

Properties

Basic			
width	Width of the control. There are three possibilities to define the width: (A) You do not define a width at all. In this case the width of the control will either be a default width or - in case of container controls - it will follow the width that is occupied by its content.	Optional	1 2 3 int-value

	(B) Pixel sizing: just input a number value (e.g. "100"). (C) Percentage sizing: input a percantage value (e.g. "50%"). Pay attention: percentage sizing will only bring up correct results if the parent element of the control properly defines a width this control can reference. If you specify this control to have a width of 50% then the parent element (e.g. an ITR-row) may itself define a width of "100%". If the parent element does not specify a width then the rendering result may not represent what you expect.		
comment	Comment without any effect on rendering and behaviour. The comment is shown in the layout editor's tree view.	Optional	
Appearance			
withplusminus	If set to "true" then +/- Icons will be rendered in front of the tree items.	Optional	true false
withlines	If set to "true" then the tree elements are connected with one another by gray lines. Please pay attention: if switching this property to "true" then you have to create the instance of your server side TREECollection object with a special constructor: Example: TREECollection m_tree = new TREECollection(true)	Optional	true false
withtooltip	If set to "true" then the text of an item is also available as tool tip. Use this option in case you expect that the horizontal space of the item will not be sufficient to display the whole text of the item.	Optional	true false
withtextinput	If set to "true" then the tree node can also be edited. Editing is started when the user double clicks the node. The text that is input is passed into the property "text" which is implemented in the default NODEInfo implementation.	Optional	true false
imageopened	Image of a tree node that has subnodes and that is currently showing its nodes. The image either is defined statically by this property or also may be defined dynamically - see the corresponding properties defined with this control.	Optional	
imageclosed	Image of a tree node that has subnodes and that is currently not showing its nodes. The image either is defined statically by this property or also may be defined dynamically - see the corresponding properties defined with this control.	Optional	
imageendnode	Image of a tree node that is an end node (leaf node). The image either is defined statically by this property or also may be defined dynamically - see the corresponding properties defined with this control.	Optional	

singleselect	If set to "true" then only one item can be selected. If set to "false" then multiple icons can be selected.	Optional	true false
directselectevent	Event that represents a tree node selection. A tree node selection is done when the user clicks/doubleclicks on the tree node text. In this case the select() method is called in the corresponding node object on server side.	Optional	ondblclick onclick
pixelshift	Number of pixels that each hierarchy level is indented. If not defined then a standard is used.	Optional	1 2 3 int-value
pixelshiftendnode	Number of pixels that end nodes are indented. If not defined then a standard is used.	Optional	1 2 3 int-value
colspan	Column spanning of control. If you use TR table rows then you may sometimes want to control the number of columns your control occupies. By default it is "1" - but you may want to define the control to span over more than one columns. The property only makes sense in table rows that are snychronized within one container (i.e. TR, STR table rows). It does not make sense in ITR rows, because these rows are explicitly not synched.	Optional	1 2 3 4 5 50 int-value
rowspan	Row spanning of control. If you use TR table rows then you may sometimes want to control the number of rows your control occupies. By default it is "1" - but you may want to define the control two span over more than one columns. The property only makes sense in table rows that are snychronized within one container (i.e. TR, STR table rows). It does not make sense in ITR rows, because these rows are explicitly not synched.	Optional	1 2 3 4 5 50 int-value
pixelheight	Height of the control in pixels.	Optional	1

			2 3 int-value
tabindex	Index that defines the tab order of the control. Controls are selected in increasing index order and in source order to resolve duplicates.	Optional	-1 0 1 2 5 10 32767
Binding			
imageprop	Name of property of the item objects that provides for a image for the tree node. Each node may provide for its own image, e.g. dependent on the type of node. If the adapter property passes back "null" then the image is taken from the static definitions that you may parallely do by using the properties IMAGEOPENED, IMAGECLOSED and IMAGEENDNODE.	Optional	
imageprop	Name of an adapter parameter that provides for a image for the tree node. Each node may provide for its own image, e.g. dependent on the type of node. If the adapter property passes back an empty string, then the image is taken from the static definitions that you may parallely do by using the properties IMAGEOPENED, IMAGECLOSED and IMAGEENDNODE.	Optional	
focusedprop	Name of property of the item objects - representing the individual rows of the collection - that indicates if the row receives the keyboard focus. Must be of type "boolean"/ "Boolean". If more than one lines are returning "true" the first of them is receiving the focus.	Optional	
focusedprop	Name of an adapter parameter that indicates if the row receives the keyboard focus.	Optional	

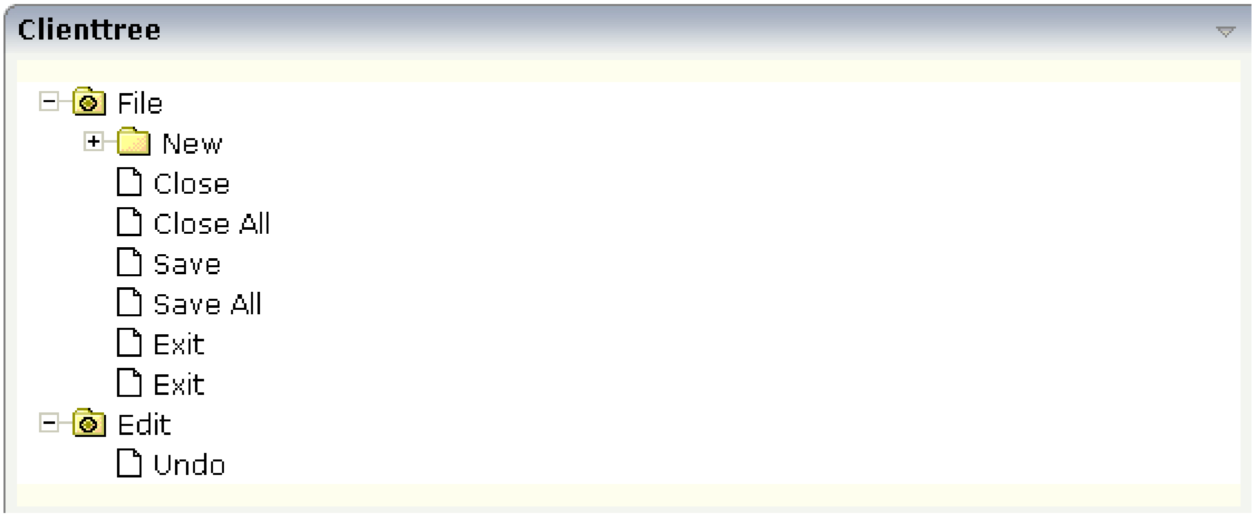
	If more than one lines are returning "true", the first of them is receiving the focus.		
flush	Flush behaviour when using the possibility of having editable tree nodes. If double clicking on the tree node then you can edit its content. The FLUSH property defines how the browser behaves when leaving the tree node's input field: If not defined ("") then nothing happens - the changed tree node text is communicated to the server side adapter object with the next roundtrip. If defined as "server" then immediately when leaving the field a roundtrip to the server is initiated - in case you want your adapter logic to directly react on the item change. If defined as "screen" then the changed tree node text is populated inside the page inside the front end.	Optional	screen server
tooltipprop	Name of property of the item objects that provides for a text that is shown if the user moves the mouse over the tree item (tooltip).	Optional	
tooltipprop	Name of an adapter parameter that provides for a text that is shown if the user moves the mouse over the tree item (tooltip).	Optional	
validdraginfosprop	Name of a property that contains a 'comma separated list' of valid drag informations.	Optional	
validdraginfosprop	Name of an adapter parameter that contains a comma separated list of valid drag informations.	Optional	
Drag and Drop			
enabledrag	If set to true then drag and drop is enabled within the tree.	Optional	true false

63 CLIENTTREE

This chapter covers the following topics:

Example

The following example shows a simple client tree:



The XML layout definition is:

```
<rowarea name="Clienttree">
    <clienttree treecollectionprop="tree" height="200" withplusminus="true"
                treestyle="background-color:#FEFEEE">
    </clienttree>
</rowarea>
```

In this example, the client tree is directly put as row into the ROWAREA container. The property `treecollectionprop` contains a reference to the property `tree` which contains the net data of the tree. With the property `treestyle`, an explicit background color is set.

Adapter Interface

In the parameter data area of the adapter, the tree data is represented by the following data structure:

```
DEFINE DATA PARAMETER
1 TREE (1:*)
2 LEVEL (I4)
2 OPENED (I4)
2 SELECTED (L)
2 TEXT (U) DYNAMIC
END-DEFINE
```

Built-in Events

value-of-treecollectionprop.reactOnContextMenuRequest
value-of-treecollectionprop.reactOnSelect
value-of-treecollectionprop.reactOnToggle

Properties

Basic			
treecollectionprop	Name of the adapter parameter that represents the control in the adapter.	Optional	
height	Height of the control. There are three possibilities to define the height: (A) You do not define a height at all. As consequence the control will be rendered with its default height. If the control is a container control (containing) other controls then the height of the control will follow the height of its content. (B) Pixel sizing: just input a number value (e.g. "20"). (C) Percentage sizing: input a percantage value (e.g. "50%"). Pay attention: percentage sizing will only bring up correct results if the parent element of the control properly defines a height this control can reference. If you specify this control to have a height of 50% then the parent element (e.g. an	Optional	100 150 200 250 300 250 400 50% 100%

	ITR-row) may itself define a height of "100%". If the parent element does not specify a width then the rendering result may not represent what you expect.		
comment	Comment without any effect on rendering and behaviour. The comment is shown in the layout editor's tree view.	Optional	
Appearance			
withplusminus	If set to "true" then +/- Icons will be rendered in front of the tree items.	Optional	true false
withtooltip	If set to "true" then the text of an item is also available as tool tip. Use this option in case you expect that the horizontal space of the item will not be sufficient to display the whole text of the item.	Optional	true false
selectionvisible	If set to "true" then the clicked item will also marked with a certain background color. The background color is defined by the style sheet settings.	Optional	true false
singleselect	If set to "true" then only one item can be selected. If set to "false" then multiple icons can be selected.	Optional	true false
imageopened	Image of a tree node that has subnodes and that is currently showing its nodes. The image either is defined statically by this property or also may be defined dynamically - see the corresponding properties defined with this control.	Optional	
imageclosed	Image of a tree node that has subnodes and that is currently not showing its nodes. The image either is defined statically by this property or also may be defined dynamically - see the corresponding properties defined with this control.	Optional	
imageendnode	Image of a tree node that is an end node (leaf node). The image either is defined statically by this property or also may be defined dynamically - see the corresponding properties defined with this control.	Optional	
treestyle	Style (following cascading style sheet definitions) that is directly passed to the background area of the client tree. You can manipulate e.g. the colour of the tree's background. The style can also be set dynamically by specifying the property TREESTYLEPROP.	Optional	
hscroll	Definition of the horizontal scrollbar's appearance. You can define that the scrollbars only are shown if the content is exceeding the control's area ("auto"). Or scrollbars can be shown always ("scroll"). Or scrollbars are never shown - and the content is cut ("hidden").	Optional	auto scroll hidden

	Default is "auto".		
pixelshift	Number of pixels that each hierarchy level is indented. If not defined then a standard is used.	Optional	1 2 3 int-value
pixelshiftendnode	Number of pixels that end nodes are indented. If not defined then a standard is used.	Optional	1 2 3 int-value
tabindex	Index that defines the tab order of the control. Controls are selected in increasing index order and in source order to resolve duplicates.	Optional	-1 0 1 2 5 10 32767
withleftpadding	Flag that indicates if the control has a 10 pixel padding on left side. Default is true.	Optional	true false
Binding			
treecollectionprop	(already explained above)		
dynamicloading	If set to "true" then you indicate to the tree control that not all tree information may be loaded when initializing the tree (i.e. the tree collection on server side). As consequence the tree control will pass the "toggle-event" to the server - in case the subnodes of a certain nodes are not yet loaded. In the case the toggle event is passed to the server, the method onToggle() is called inside the tree item.	Optional	true false
imageopenedprop	Name of the adapter parameter that provides the image URL which is shown for opened tree nodes or end tree nodes. The value may be different from tree node to tree node. Each tree node may have an own image.	Optional	

imageclosedprop	Name of the adapter parameter that provides for the image URL which is shown for closed tree nodes. The value may be different from tree node to tree node. Each tree node may have an own image.	Optional	
treestyleprop	name of the adapter parameter that dynamically provides for a style value that is passed to the control's area (background of the client tree). You can as consequence e.g. define the background-colour of the tree dependent on your server side logic.	Optional	
treeclassprop	Name of the adapter parameter that passes back the name of a style sheet class that is taken to render the client tree's background area. - Similar to the property TREESTYLEPROP, but now a style class is passed, not the style itself.	Optional	
tooltipprop	Name of the adapter parameter that provides for a text that is shown if the user moves the mouse over the tree item (tooltip).	Optional	
oncontextmenumethod	Name of the event that is sent to the adapter when the user presses the right mouse button in an empty area of the client tree.	Optional	
directselectevent	Event that represents a tree node selection. A tree node selection is done when the user clicks/doubleclicks on the tree node text. In this case the select() method is called in the corresponding node object on server side.	Optional	ondblclick onclick
focusedprop	Name of the adapter parameter that indicates if the row receives the keyboard focus. If more than one lines are returning "true", the first of them is receiving the focus.	Optional	
Drag and Drop			
enabledrag	If set to true then drag and drop is enabled within the tree.	Optional	true false

64 Working with Menus

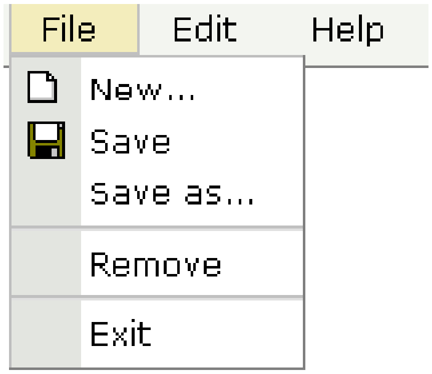
Menus are used to arrange a number of functions in a structured way.

The information provided in this part is organized under the following headings:

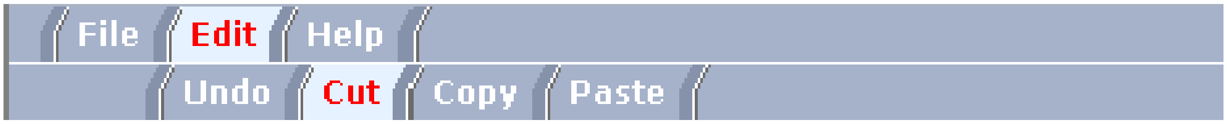
65 Types of Menus

The following menu controls are available:

- **MENU**
  This is the typical drop-down menu:



- **DLMENU**
  This is a double-line menu representing a two-level hierarchy. It can be found quite often in web applications.



When clicking an item in the first line, the corresponding subitems are shown in the second line.

All menu controls are dynamically configured by the application. This means:

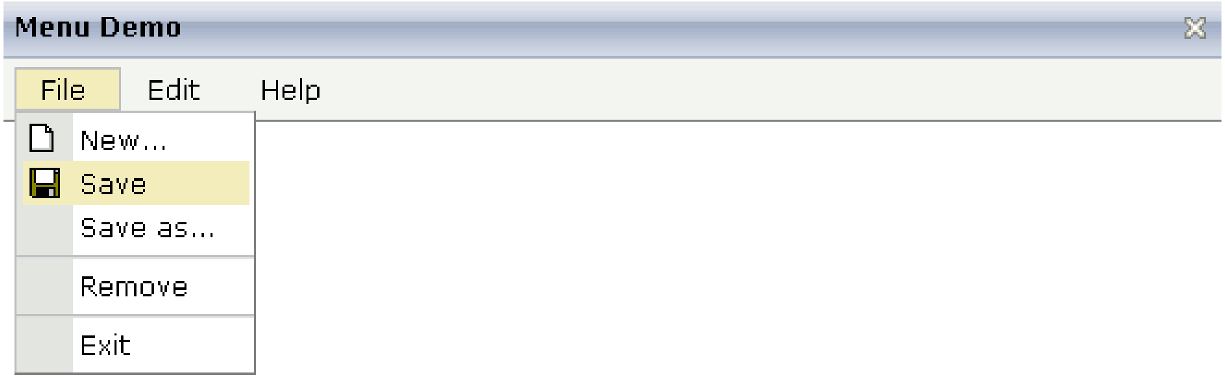
- The structure of the menu and its menu nodes is not statically defined but is dynamically controlled by the application through adapter parameters. For example, you can build a personalized menu taking the user's rights into consideration.
- Menu information can be dynamically updated during runtime.

66 MENU

This chapter covers the following topics:

Example

The example looks as follows:



When clicking on a menu item for which a function has been defined, then the name of the function is displayed in the status bar.

The XML layout definition is:

```
<page model="Menue_01_Adapter">
    <titlebar name="Menu Demo">
    </titlebar>
    <header align="left" withdistance="false">
        <menu menucollectionprop="menuData" width="100">
        </menu>
    </header>
    <pagebody>
    </pagebody>
    <statusbar withdistance="false">
    </statusbar>
</page>
```

In this example, the menu is embedded in the header. By the property `menucollectionprop`, it is bound to the adapter property `menuData`.

Adapter Interface

```
DEFINE DATA PARAMETER
1 MENUDATA (1:*)
2 IMAGEURL (U) DYNAMIC
2 LEVEL (I4)
2 METHOD (U) DYNAMIC
2 OPENED (I4)
2 TEXT (U) DYNAMIC
1 SELMENUITEM (U) DYNAMIC
END-DEFINE
```

Built-in Events

items.reactOnSelect

Properties

Basic			
menucollectionprop	Name of the adapter parameter that represents the control in the adapter.	Obligatory	
comment	Comment without any effect on rendering and behaviour. The comment is shown in the layout editor's tree view.	Optional	
Appearance			
width	Width of the control. There are three possibilities to define the width: (A) You do not define a width at all. In this case the width of the control will either be a default width or - in case of container controls - it will follow the width that is occupied by its content. (B) Pixel sizing: just input a number value (e.g. "100"). (C) Percentage sizing: input a percantage value (e.g. "50%"). Pay attention: percentage sizing will only bring up correct results if the parent element of the control properly defines a width this control can reference. If you specify this control to have a width of 50% then the parent element (e.g. an ITR-row) may itself define a width of "100%". If the parent element does not specify a width then the rendering result may not represent what you expect.	Optional	100 120 140 160 180 200 50% 100%
height	Height of the control. There are three possibilities to define the height: (A) You do not define a height at all. As consequence the control will be rendered with its default height. If the control is a container control (containing) other controls then the height of the control will follow the height of its content. (B) Pixel sizing: just input a number value (e.g. "20"). (C) Percentage sizing: input a percantage value (e.g. "50%"). Pay attention: percentage sizing will only bring up correct results if the parent element of the control properly defines a height this control can reference. If you specify this control to have a height of 50% then the parent element (e.g. an ITR-row) may itself define a height of "100%". If the parent element does not specify a width then the rendering result may not represent what you expect.	Optional	
toggleimage	URL of the image that is shown on the right end of a menu item, if this item contains subitems. If not explicitly defined then a default icon is used.	Optional	

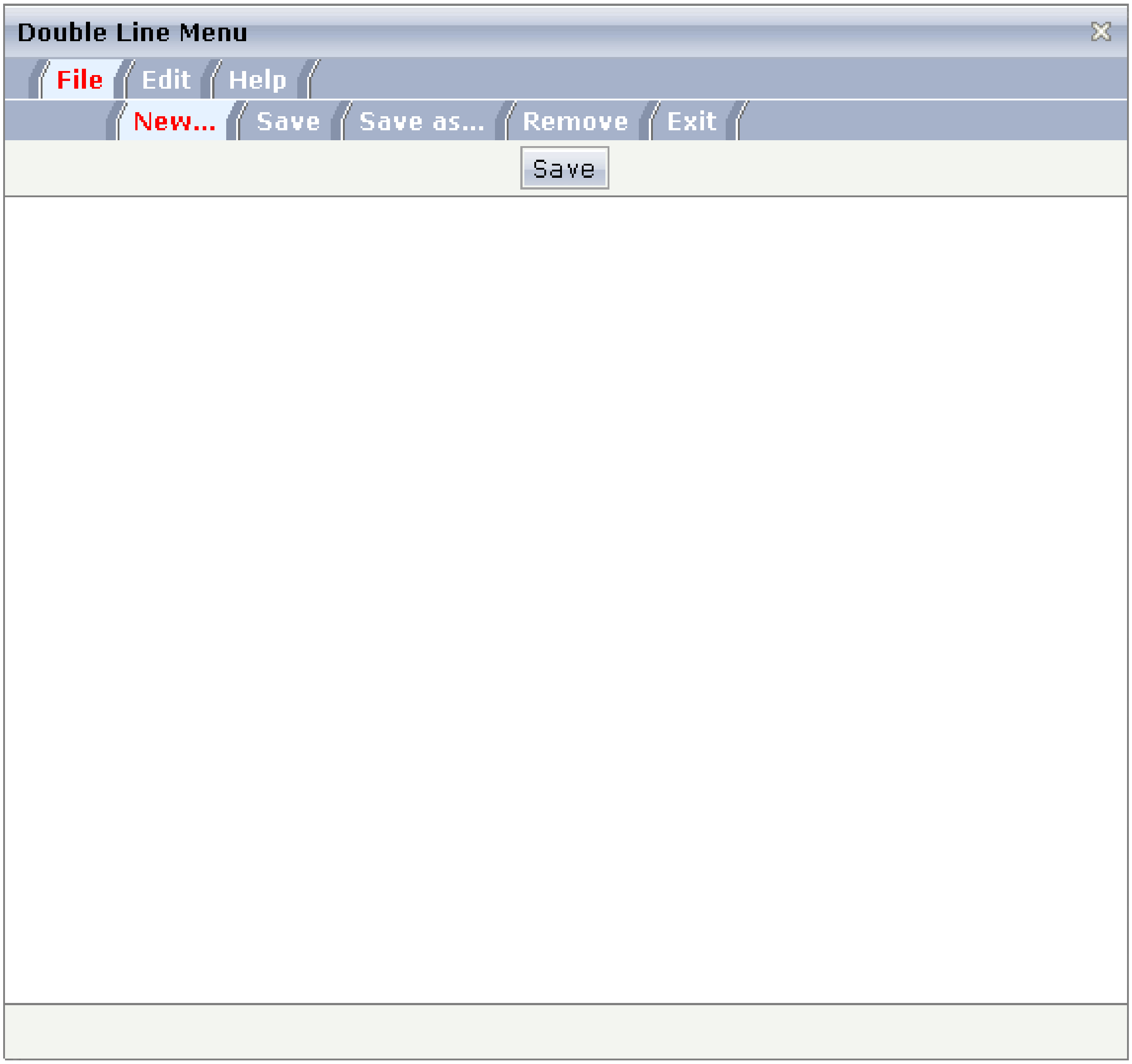
toggleimageprop	Name of the adapter parameter that provides a URL that defines the toggle image. The toggle icon is shown on the right end of a menu item that has subitems.	Optional	
menustyle	CSS style definition that is directly passed into this control. With the style you can individually influence the rendering of the control. You can specify any style sheet expressions. Examples are: border: 1px solid #FF0000 background-color: #808080 You can combine expressions by appending and separating them with a semicolon. Sometimes it is useful to have a look into the generated HTML code in order to know where direct style definitions are applied. Press right mouse-button in your browser and select the "View source" or "View frame's source" function.	Optional	
menustyleprop	Name of the adapter parameter that dynamically provides explicit style information for the control.	Optional	

67 DLMENU

This chapter covers the following topics:

Example

The example looks as follows:



A double-line menu is displayed. When selecting a menu item, then its text is written to the status bar.

The XML layout definition is:

```
<page model="menue_02_dl_Adapter">
    <titlebar name="Double Line Menu">
    </titlebar>
    <dlmenu menuprop="menuData">
    </dlmenu>
    <header withdistance="false">
        <button name="Save">
        </button>
    </header>
    <pagebody>
    </pagebody>
    <statusbar withdistance="false">
    </statusbar>
</page>
```

The DLMENU control is positioned directly following the title bar. In its property `menuprop`, it holds a binding to the property `menuData`.

Adapter Interface

```
DEFINE DATA PARAMETER
1 ITEMS (1:*)
2 LEVEL (I4)
2 METHOD (U) DYNAMIC
2 TEXT (U) DYNAMIC
END-DEFINE
```

Built-in Events

items.onSelectSubItem

Properties

Basic			
menuprop	Name of the adapter parameter that represents the control in the adapter.	Obligatory	
textid	Multi language dependent text that is displayed inside the control. The "textid" is translated into a corresponding string at runtime. Do not specify a "name" inside the control if specifying a "textid".	Optional	
align	Horizontal alignment of the control's content.	Optional	left center right
onlyoneline	If set to "true" then the DLMENU control only contains its top line - there is no second line below. Default is "false".	Optional	true false
comment	Comment without any effect on rendering and behaviour. The comment is shown in the layout editor's tree view.	Optional	

68 Non-Visual Controls and Hot Keys

This part describes some controls that do not have any visual effect to your screen, but provide some client functions to be applied to your page.

The information provided in this part is organized under the following headings:

- TIMER
- Extended Hot Key Management

69 TIMER

With a timer, you can regularly trigger a defined event sent by the client. For example, you can use a timer to regularly update information to be displayed inside your page.

The timer tag is accessible as a valid subnode inside the page tag.

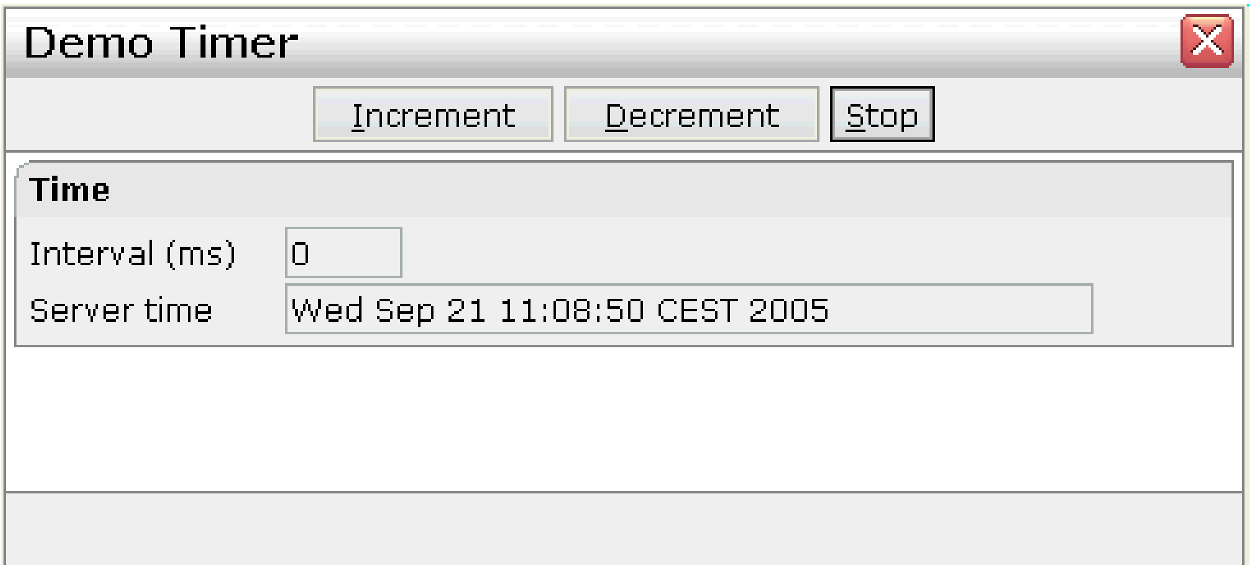
Specify either the `interval` or the `intervalprop` property in order to set the interval. In case of using a property for dynamically setting the interval, note the following:

- You can change the interval time at any time.
- You can stop the timer by setting the interval time to 0.

The following topics are covered below:

Example

The following screen displays a time stamp of the server. It is refreshed depending on the interval field. Increase/decrease the interval time by choosing the corresponding buttons.



Demo Timer

Time

Interval (ms): 0

Server time: Wed Sep 21 11:08:50 CEST 2005

The XML layout definition is:

```
<page model="DemoTimerAdapter">
    <titlebar name="Demo Timer">
    </titlebar>
    <header withdistance="false">
        <button name="~~Increment" method="incrementTimer">
        </button>
        <button name="~~Decrement" method="decrementTimer">
        </button>
        <button name="~~Stop" method="stopTimer">
        </button>
    </header>
    <pagebody>
        <rowarea name="Time">
            <itr>
                <label name="Interval (ms)" width="100" asplaintext="true">
                </label>
                <field valueprop="interval" length="5" displayonly="true"
datatype="int">
                </field>
            </itr>
            <itr>
                <label name="Server time" width="100" asplaintext="true">
                </label>
                <field valueprop="serverTime" length="50" displayonly="true">
                </field>
            </itr>
        </rowarea>
    </pagebody>
    <statusbar withdistance="false">
    </statusbar>
    <timer intervalprop="interval">
    </timer>
</page>
```

In this example, the timer tag does not send a defined event but refreshes the screen. The timer interval is retrieved by the property `interval` of the adapter object.

Properties

Basic			
interval	Duration in milliseconds the timer waits between calling the adapter method defined in the METHOD property. Use this property to "hard code" the duration - or use INTERVALPROP to define the duration by an adapter property.	Sometimes obligatory	
intervalprop	Name of the adapter parameter that defines the timer interval duration. If 0 is passed then the timer is stopped.	Sometimes obligatory	
method	Name of the event that is sent to the adapter by the timer.	Obligatory	
comment	Comment without any effect on rendering and behaviour. The comment is shown in the layout editor's tree view.	Optional	

70 Extended Hot Key Management

Extended hot key management provides the following features:

- Possibility to define hot keys with certain controls.
- Possibility to define language dependent hot keys.

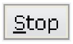
The following topics are covered below:

Direct Hot Key Definitions with Certain Controls

Some controls allow to directly specify hot keys within the text that is displayed inside the control. The controls that currently support this feature are:

- BUTTON
- MENU
- ROWTABAREA

Example: If you specify the button text to be "~~Stop", the button will look like this:



The text may both be directly maintained in the control (`name` property) or may come from the multi language management (`textid` property).

At the time, the hot key CTRL+ALT+S will be added to the page. The definition of hot keys in the texts of MENU controls or ROWTABAREA controls is done in the same way.

Caution: Application Designer does not check if hot keys are defined twice in a page.

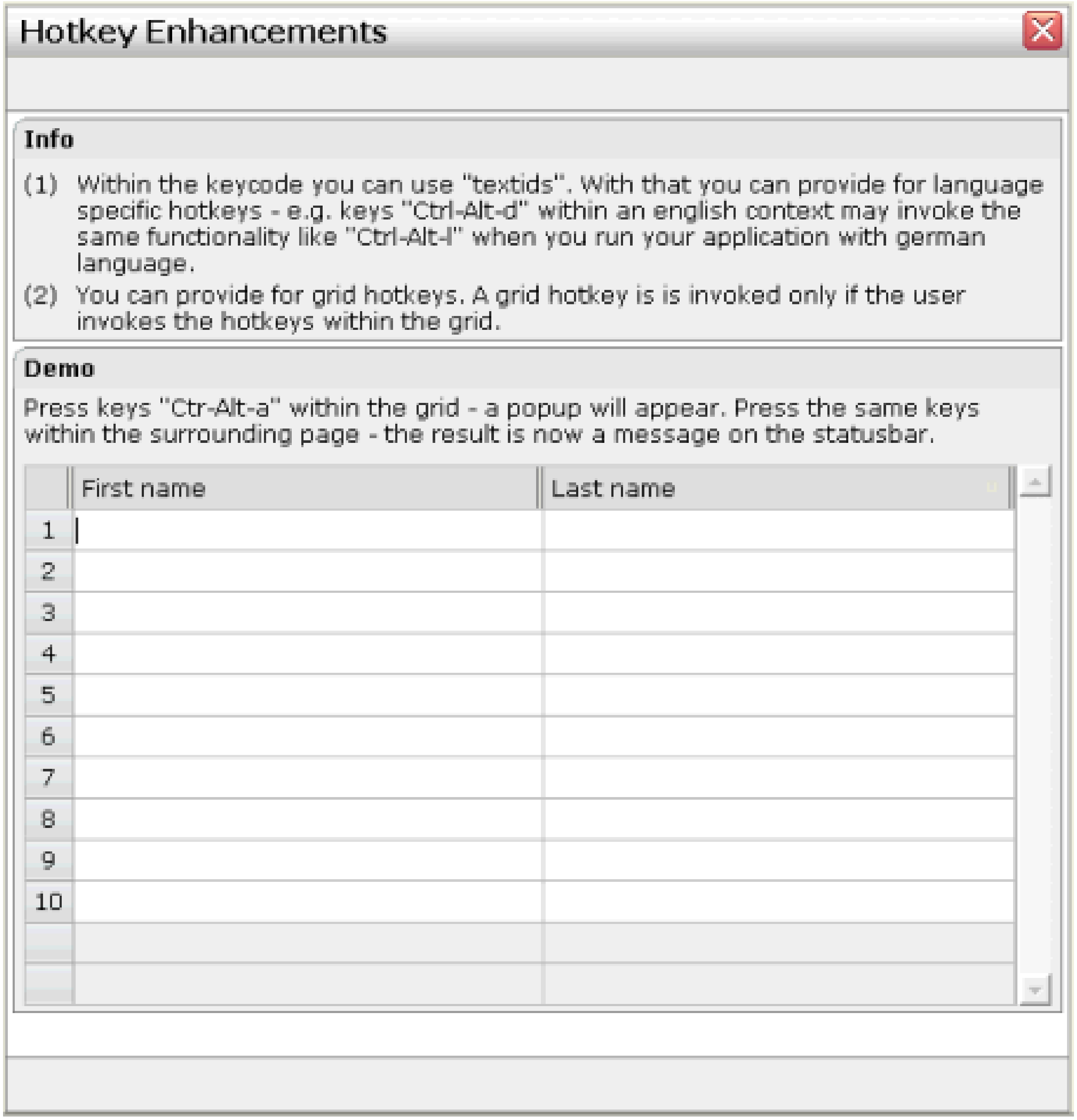
Why use CTRL+ALT as a default way to trigger the hot keys? This is because most of the simple ALT keys are already occupied by the browser.

Hot Key Definitions for Certain Controls

The controls PAGE, FIELD and ROWTABLEAREA2 support the property `hotkeys`.

The `hotkeys` property defines the active hot keys for the corresponding control. This means that you may have hot keys that are only valid inside a certain grid (ROWTABLEAREA2 control) or even inside a single FIELD, but are not valid inside the whole page (PAGE control).

Have a look at the following demo:



Hotkey Enhancements

Info

(1) Within the keycode you can use "textids". With that you can provide for language specific hotkeys - e.g. keys "Ctrl-Alt-d" within an english context may invoke the same functionality like "Ctrl-Alt-l" when you run your application with german language.

(2) You can provide for grid hotkeys. A grid hotkey is is invoked only if the user invokes the hotkeys within the grid.

Demo

Press keys "Ctr-Alt-a" within the grid - a popup will appear. Press the same keys within the surrounding page - the result is now a message on the statusbar.

	First name	Last name
1		
2		
3		
4		
5		
6		
7		
8		
9		
10		

If the user presses CTRL+ALT+A inside the grid, the hot key is managed by the grid. If the user presses the same key outside the grid, the hot key is processed by a corresponding definition on page level. The XML layout looks as follows:

```
<page model="com.softwareag.cis.test40.GridHotkeysAdapter"
translationreference="40_gridhotkeys"
      hotkeys="ctrl-alt-65;onCtrlAltAPage">
...
...
        <rowtablearea2 griddataprop="grid" rowcount="12" width="100%"
firstrowcolwidths="true"
                       hotkeys="ctrl-alt-$KEYCODE_A;onCtrlAltA">
...
...
```

The `hotkeys` property on PAGE, FIELD or ROWTABLEAREA2 is a semicolon-separated list containing the hot key itself and the method it is calling. There can be multiple hot key definitions for the same control. When maintaining this property, use the special dialog in the Layout Painter that appears for the `hotkeys` property.

You can either specify the key code of the hot key or a text ID that is to be translated by the multi language management.

索引