Application Designer

About this Documentation

Version 9.1.1

October 2018

ADABAS & NATURAL

This document applies to Application Designer Version 9.1.1 and all subsequent releases.

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

Document ID: CIT-CCITFIRSTSTEPS-911-20181006

Table of Contents

1 About this Documentation

Document Conventions

Convention	Description
Bold	Identifies elements on a screen.
`Monospace font`	Identifies service names and locations in the format *`folder.subfolder.service`*, APIs, Java classes, methods, properties.
`Italic`	Identifies: Variables for which you must supply values specific to your own situation or environment. New terms the first time they occur in the text. References to other documentation sources.
`Monospace font`	Identifies: Text you must type in. Messages displayed by the system. Program code.
{ }	Indicates a set of choices from which you must choose one. Type only the information inside the curly braces. Do not type the { } symbols.
|	Separates two mutually exclusive choices in a syntax line. Type one of these choices. Do not type the | symbol.
[ ]	Indicates one or more options. Type only the information inside the square brackets. Do not type the [ ] symbols.
...	Indicates that you can type multiple options of the same type. Type only the information. Do not type the ellipsis (...).

Online Information and Support

Software AG Documentation Website

You can find documentation on the Software AG Documentation website at **http://documentation.softwareag.com**. The site requires credentials for Software AG's Product Support site Empower. If you do not have Empower credentials, you must use the TECHcommunity website.

Software AG Empower Product Support Website

If you do not yet have an account for Empower, send an email to empower@softwareag.com with your name, company, and company email address and request an account.

Once you have an account, you can open Support Incidents online via the eService section of Empower at **https://empower.softwareag.com/**.

You can find product information on the Software AG Empower Product Support website at **https://empower.softwareag.com**.

To submit feature/enhancement requests, get information about product availability, and download products, go to **Products**.

To get information about fixes and to read early warnings, technical papers, and knowledge base articles, go to the **Knowledge Center**.

If you have any questions, you can find a local or toll-free number for your country in our Global Support Contact Directory at **https://empower.softwareag.com/public_directory.asp** and give us a call.

Software AG TECHcommunity

You can find documentation and other technical information on the Software AG TECHcommunity website at **http://techcommunity.softwareag.com**. You can:

- Access product documentation, if you have TECHcommunity credentials. If you do not, you will need to register and specify "Documentation" as an area of interest.
- Access articles, code samples, demos, and tutorials.
- Use the online discussion forums, moderated by Software AG professionals, to ask questions, discuss best practices, and learn how other customers are using Software AG technology.
- Link to external websites that discuss open standards and web technology.

Data Protection

Software AG products provide functionality with respect to processing of personal data according to the EU General Data Protection Regulation (GDPR). Where applicable, appropriate steps are documented in the respective administration documentation.

2 First Steps

This documentation is organized under the following headings:

About this Tutorial	How the completed "Hello World!" application will appear to the user. An overview of the basic steps that are required to create this application.
Starting the Development Workplace	How to start the development workplace.
Creating a Project	How to create the project that is to be used for this tutorial.
Getting Started with the Layout Painter	How to create a layout for your project. General information on how to use the Layout Painter.
Writing the GUI Layout	How to use the Layout Painter in order to create the GUI layout for the "Hello World!" application.
Setting Up Your Environment	How to set up Application Designer in your development environment for Java.
Writing the Adapter Code	How to define the class and how to use the Code Assistant in order to generate most of the adapter code. How to program the adapter code for the method using your development environment for Java and how to test the completed application. How to view the generated HTML page directly from the browser.
Some Background Information	Information on changing the adapter code. About name binding between controls and adapter, data exchange at runtime, and files and their locations. Some information for real development and on some tools provided in the development workspace.

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.

3 About this Tutorial

This tutorial provides an introduction to working with Application Designer's development workplace. It explains how to create a “Hello World!” application. This covers all basic steps you have to perform when creating pages with Application Designer: you create a layout file, you create an adapter class, and you run the application. This is about eighty percent of all you need to know. The remainder of the Application Designer documentation contains just details and some more sophisticated techniques, such as navigating between pages.

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

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:

This corresponds to the following XML layout:

```
<?xml version="1.0" encoding="UTF-8"?>
<page model="DummyAdapter">
    <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>
</page>
```

3. When you save your layout for the first time, an intelligent HTML page is generated.
4. Before you can start coding, you have to make specific definitions in Application Designer and in your development environment for Java (for example, Eclipse).
5. You will use Application Designer's Code Assistant to generate most of the adapter code.
6. You will then switch to your development environment in order to program the adapter code for the method `sayHello` which will be executed when you choose the **Say Hello** button of your application.

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

4 Starting the Development Workplace

This tutorial assumes that you are using the Tomcat servlet engine which comes with this installation.

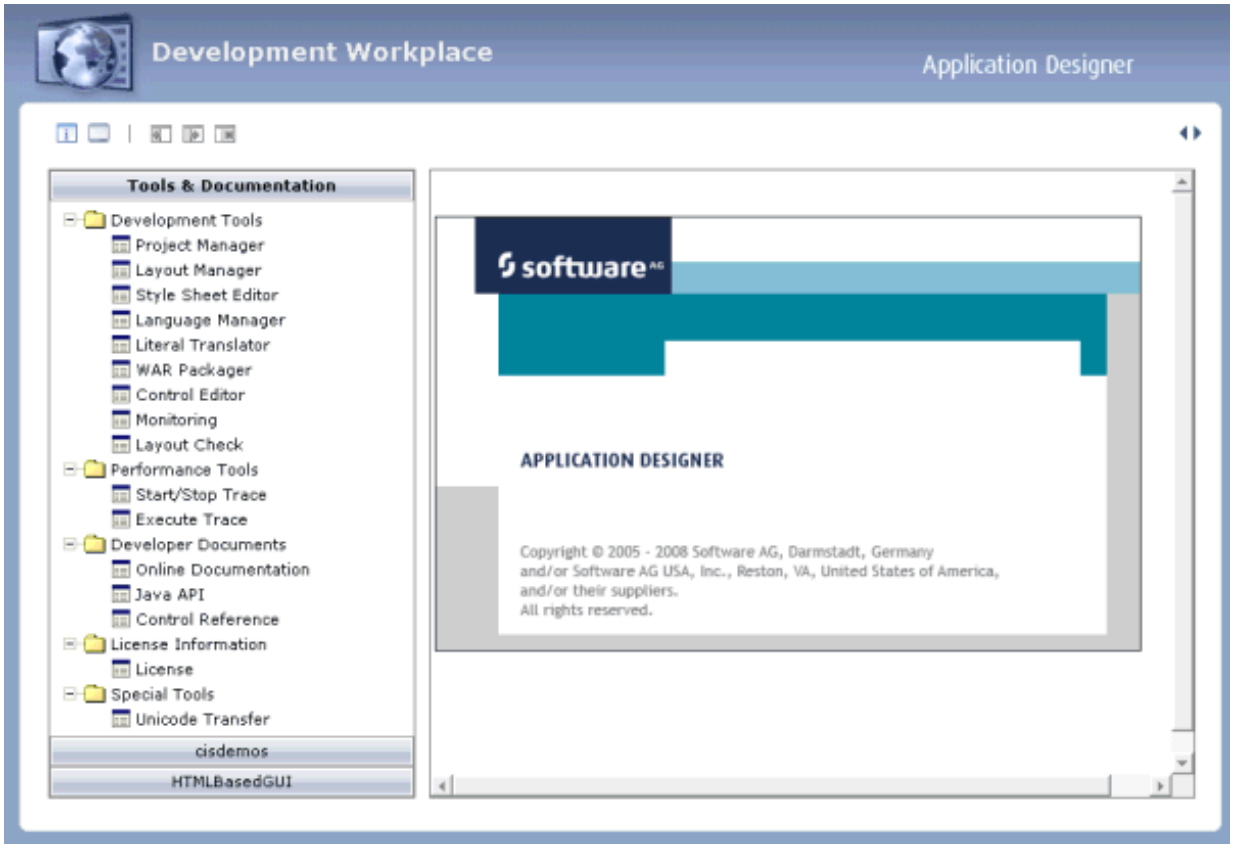
To start the development workplace

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

```
http://localhost:51000/cis/HTMLBasedGUI/workplace/ide.html
```

Note: If you have defined another port number during installation, enter this port number instead of the default port number 51000.

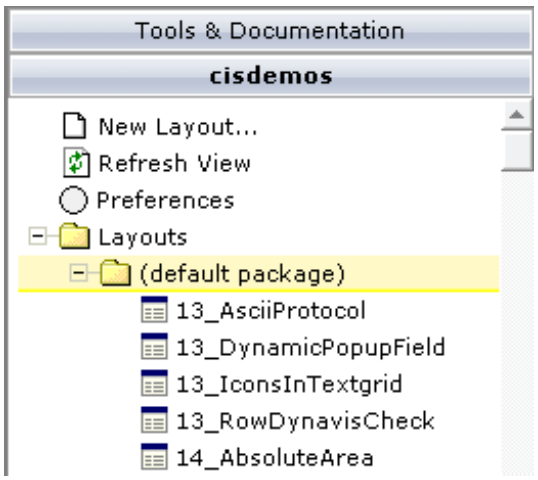
The development workplace is now shown in your browser.



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

5 Creating a Project

In the Application Designer environment, layouts and classes 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 "cisyourfirstproject".

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:

4 Enter "cisyourfirstproject" 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.

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***.

6 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.

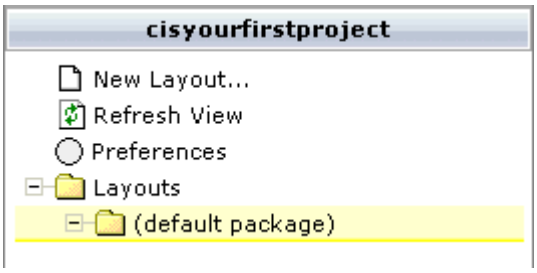
Creating a New Layout

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

To choose a layout template

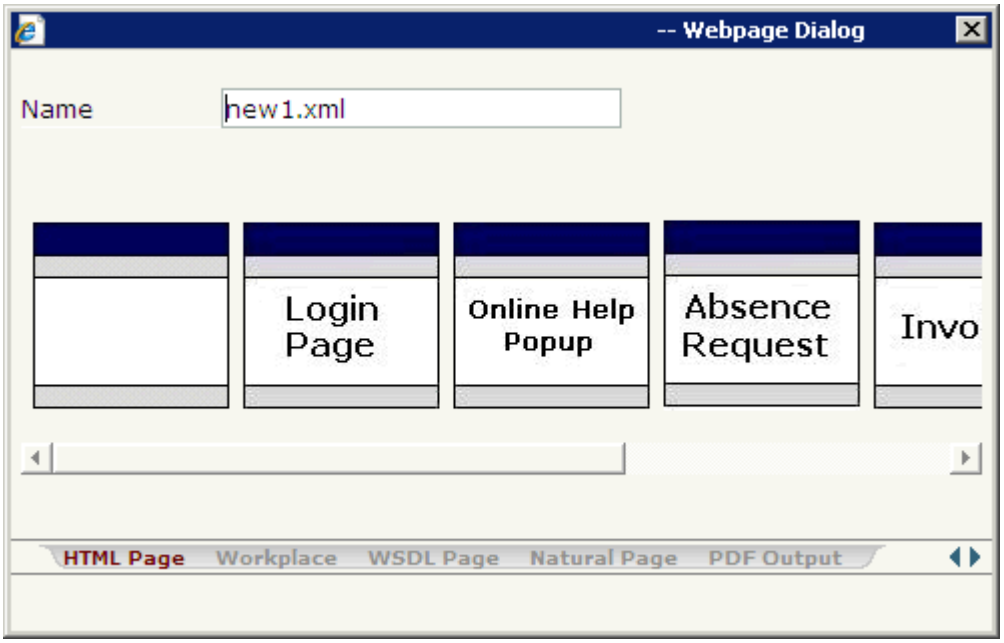
1 Choose the button for the project **cisyourfirstproject**.

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.

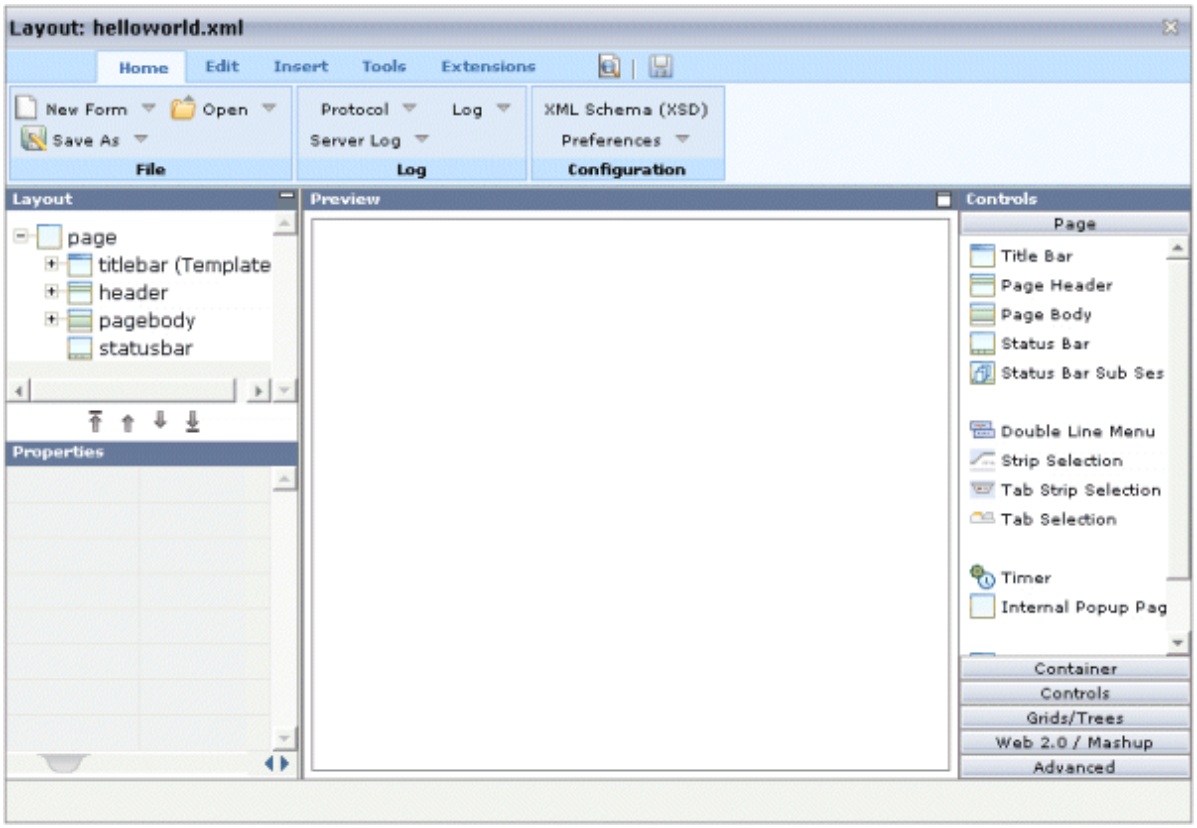


3 Enter "helloworld.xml" in the **Name** field.

This is the name of your layout definition.

4 Select the first template (when you move the mouse over this template, the tool tip "HTML 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).

  The Code Assistant, which will also be used in this tutorial, is also shown in the preview area.

- **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

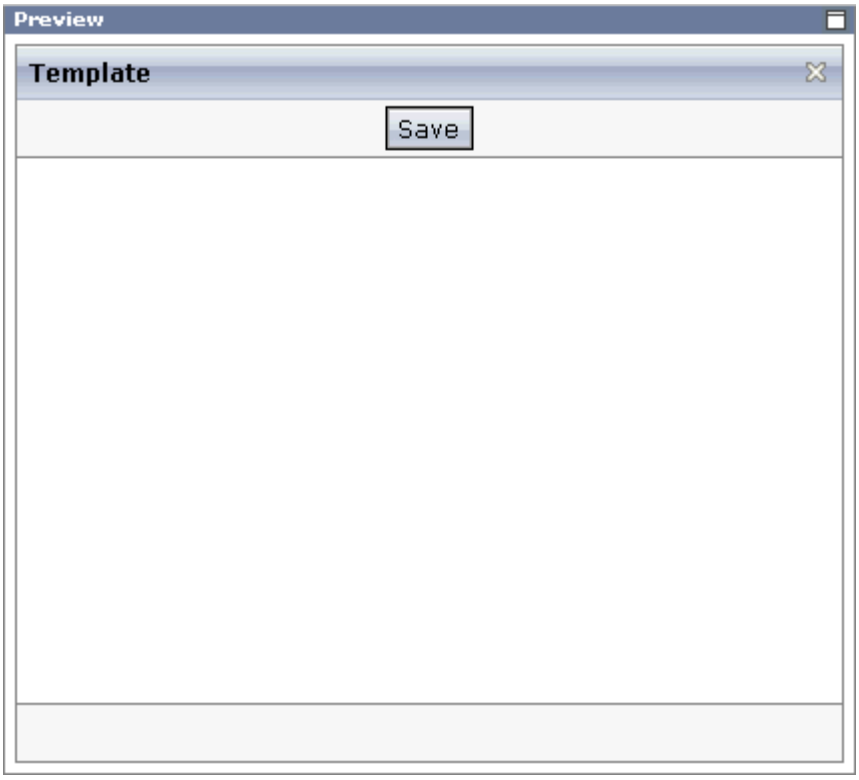
The layout tree inside the Layout Painter already contains some nodes that were copied from the template that you chose in the dialog in which you specified the name of the page. To see what the page looks like, preview the layout as described below.

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 a **Save** 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.

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.

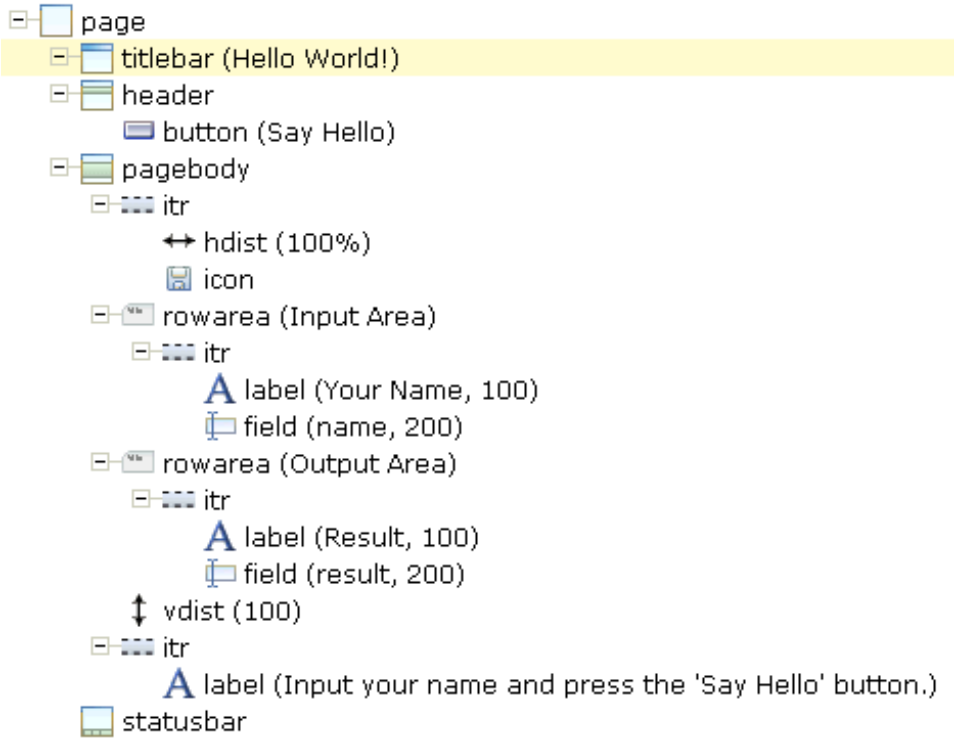
```
<page model="DummyAdapter">
    <titlebar name="Template">
    </titlebar>
    <header withdistance="false">
        <button name="Save">
        </button>
    </header>
    <pagebody>
    </pagebody>
```

```
    <statusbar withdistance="false">
    </statusbar>
</page>
```

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

7 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*.



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 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.

To specify the name for the title bar

1 In the layout tree, select the node **titlebar (Template)**.

The properties for this control are now shown in the properties area at the bottom. You can see the default entry "Template" 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!)**.

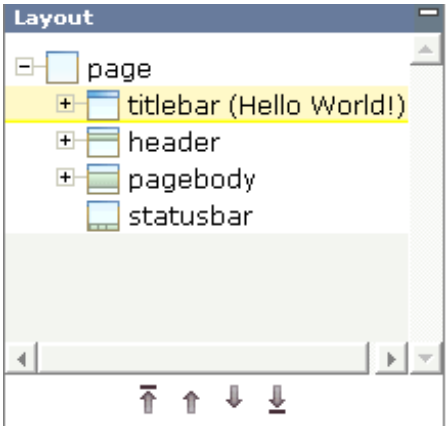
Note: Properties that are left blank are not shown in the XML code.

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.

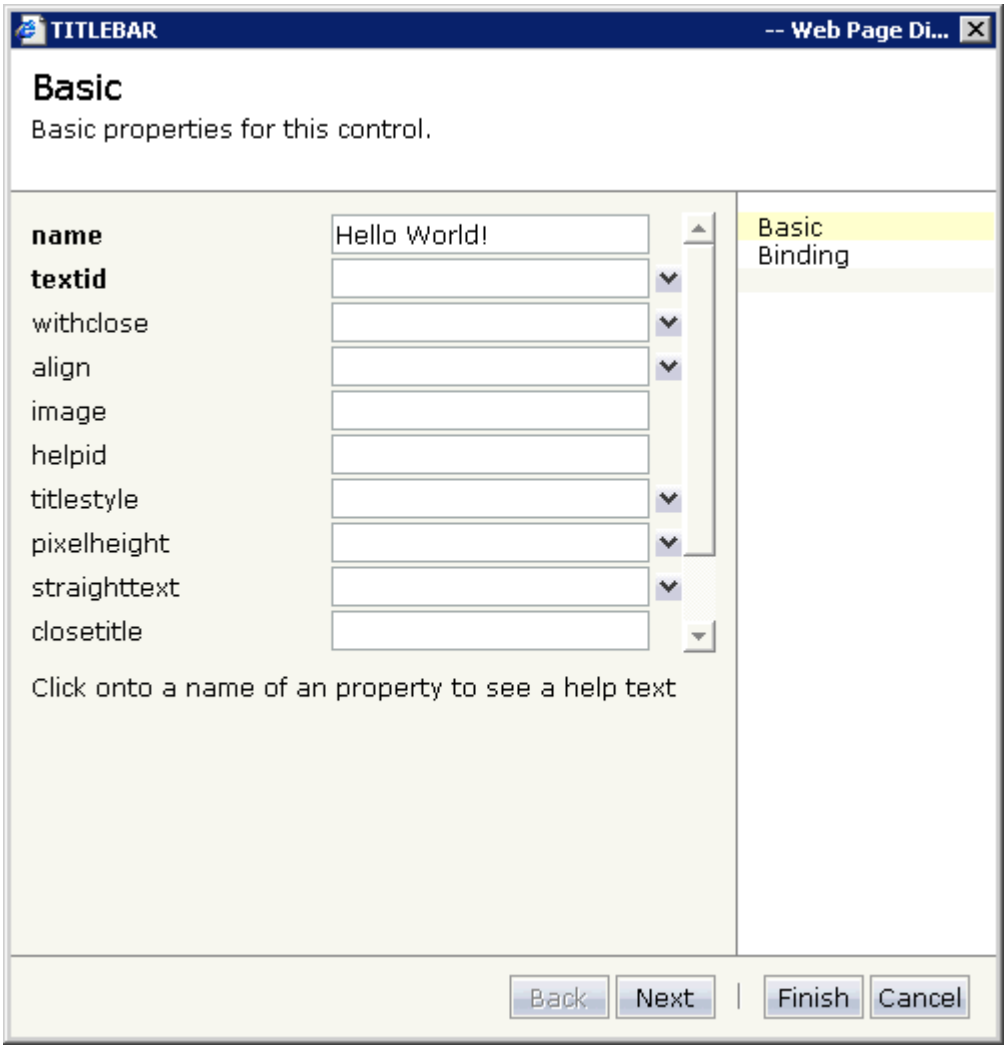
To use the Property Editor

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.



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.

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 "Save".

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

3 Specify the following properties:

Property	Value
`name`	Say Hello
`method`	sayHello

The method needs to be programmed in the adapter class. 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.

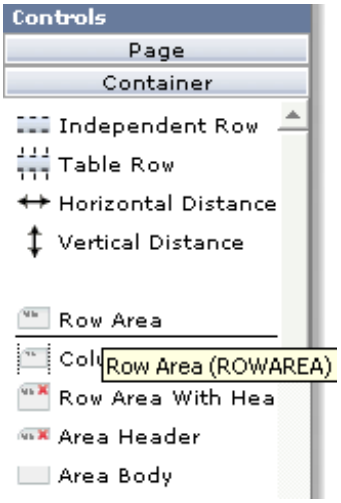
Each row area will contain an **Independent Row** control which in turn contains a **Label** and a **Field** control. These controls can be found in the **Controls** section of the controls palette.

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.

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.

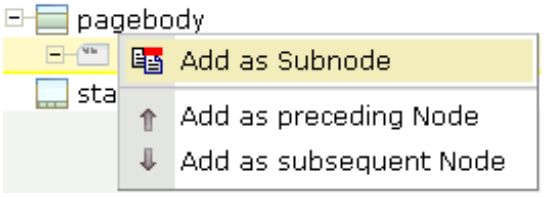
The row area is added as a subnode of the **pagebody** node. The new subnode is automatically selected so that you can maintain the properties of the row area directly in the properties area.

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**.

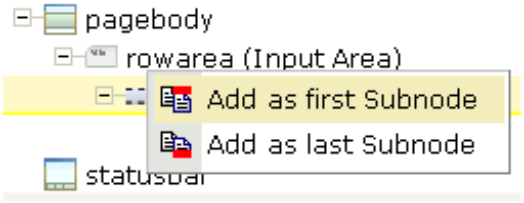
6 Open the **Controls** section of the controls palette.

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

Note: As long as the adapter class is not defined, the field is dimmed and it is not possible to enter information in this field. The adapter class will be defined later in this tutorial.

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.

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.

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.

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.

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.

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*.

8 Setting Up Your Environment

Before you start coding, you have to make specific definitions in Application Designer and in your development environment for Java.

Setting Up Application Designer

You have to define the source directory into which the adapter classes for your project are to be written. This directory will be used by the Code Assistant that you will use later in this tutorial.

To set up Application Designer

1 Create a new directory with the name "src" in the directory that has been created for your project; that is: go to the directory *<installdir>/tomcat/webapps/cis/cisyourfirstproject* and create the directory */src*.

2 From the **Home** tab of the Layout Painter, choose **Preferences**.

The following dialog appears:

3 In the **Source Directory** field, specify the path to the *src* directory you have just created.

Note: By default, the prefix "m_" is defined in this dialog. This prefix is used for all properties in the adapter class.

4 Choose the **Save and Apply** button.

Setting Up Your Development Environment for Java

Typically, development environments provide project management to define project settings.

To set up your development environment for Java

1 Create a new Java project.

It is recommended that you use the same project name as in your Application Designer project. For this tutorial, you should therefore create a Java project with the name "cisyourfirstproject".

2 Include the file *cis.jar* in the required libraries of your project. You find this file in the following Application Designer directory:

<installdir>/tomcat/webapps/cis/WEB-INF/lib

3 Include the file *servlet-api.jar* in the required libraries of your project. You find this file in the following Application Designer directory:

<installdir>/tomcat/common/lib

4 Set up the output path (i.e. the path where the compiled classes will be located) so that it points to the following directory:

<installdir>/tomcat/webapps/cis/cisyourfirstproject/appclasses/classes

You can now proceed with the next exercise: ***Writing the Adapter Code***.

9 Writing the Adapter Code

Defining the Class Name

As a last step of your layout definition, you will now define the name of the class which is the logical counterpart of your page.

To define the class name

1 In the layout tree, select the top node **page**.

You can see the default entry "DummyAdapter" for the `model` property.

2 Specify the following property:

Properties	Value
`model`	HelloWorldAdapter

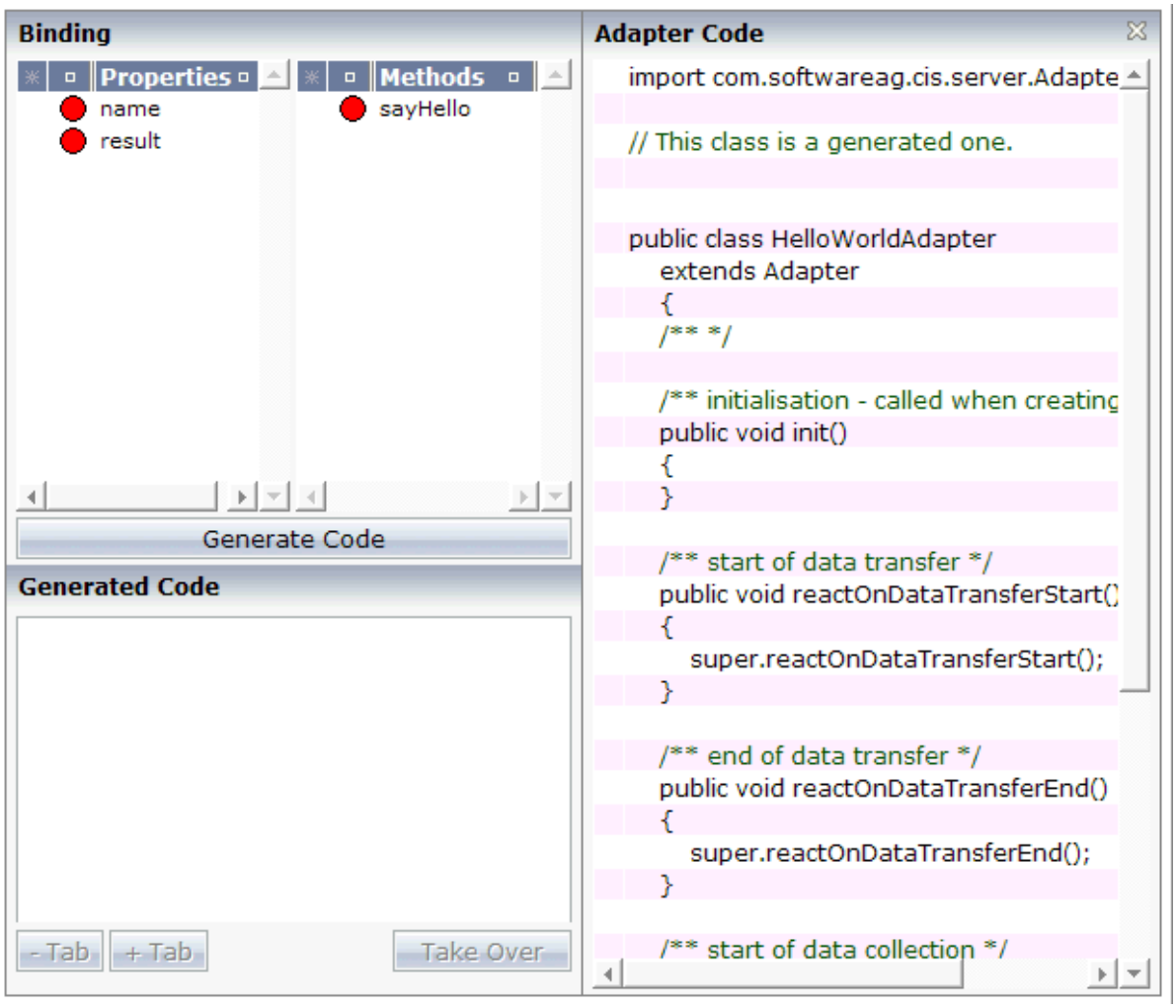
Generating the Code

You use the Code Assistant, which is part of the Layout Painter, to generate code for the referenced properties and methods.

To generate code

1 From the **Tools** tab of the Layout Painter, choose **Code Assistant**.

The following is shown in the preview area:



The properties and methods which cannot be found in the adapter code are indicated by red dots.

2 Select the property `name`.

3 Choose the **Generate Code** button.

The generated code appears below this button.

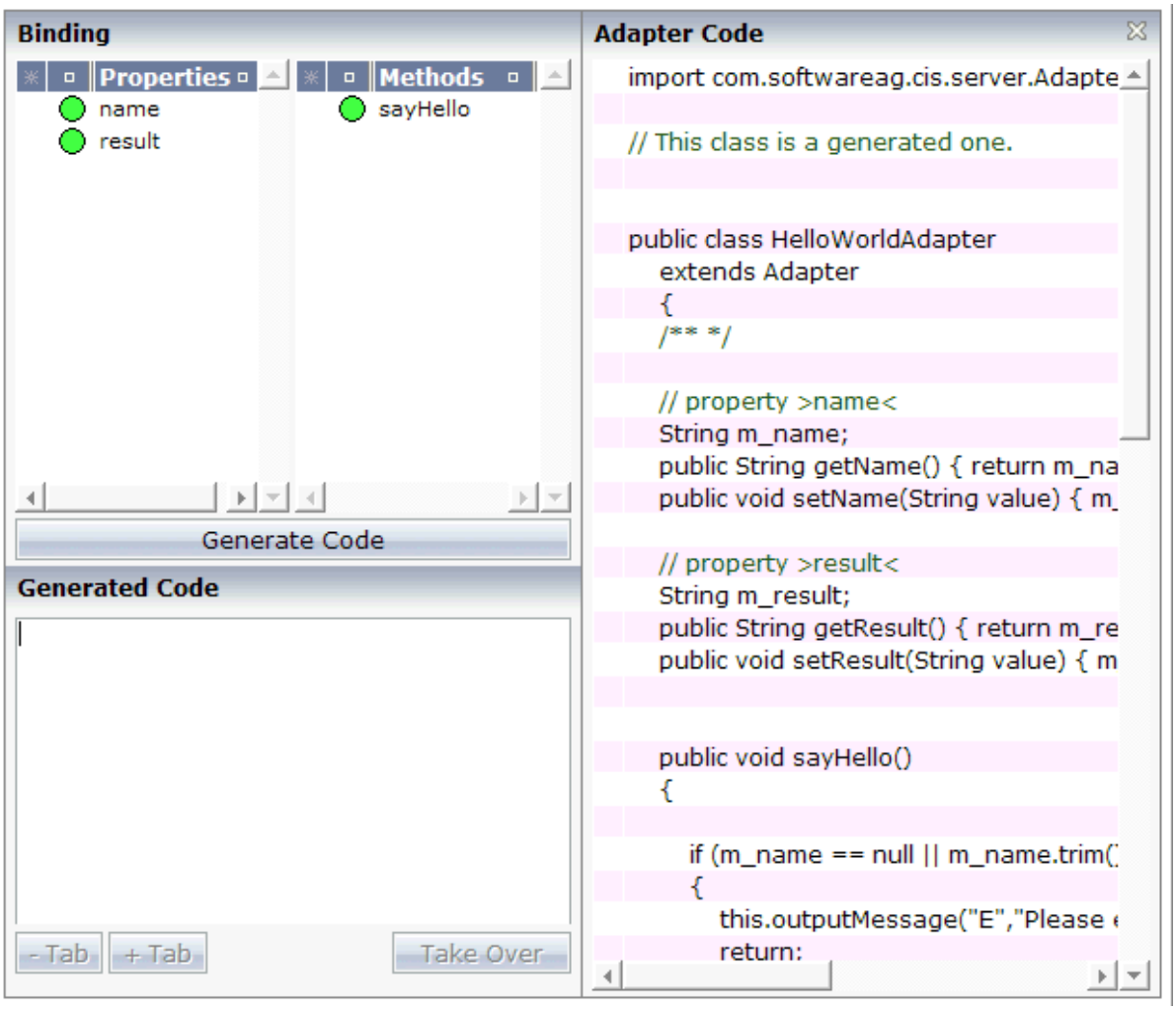
4 Select the position in the adapter code at which you want to insert the code.

5 Choose the **Take Over** button to insert the code at the selected position.

The color of the dot changes from red to green. This indicates that the property's `get` and `set` methods are now available in the adapter code.

6 Repeat the above steps for the `result` property and the `sayHello` method.

When you have finished all required steps, the page should look as follows:



7 Save your changes and close the Code Assistant.

Programming the Method

Now that the framework for the `sayHello` method has been generated in the adapter code, you have to program the method itself.

To program the method

1 Go to your development environment for Java.

2 Open the file *HelloWorldAdapter.java* which is located in the project directory */cisyourfirstproject/src*.

3 Add the line which is indicated in bold:

```
/** */
    public void sayHello()
    {
        // TODO Auto-generated method stub
     m_result = "Hello World, " + m_name + "!";
    }
```

4 Save your changes.

Testing the Completed Application

You will now check whether your application provides the desired result.

To test the application

1 Go back to your layout in the Layout Painter.

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

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

The page should perform correctly now, and successfully “talk” to the adapter you have just created:

Viewing the Page Outside the Layout Painter

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>/tomcat/webapps/cis/cisyourfirstproject*.

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.

For viewing the page directly, enter the following URL inside your browser:

```
http://localhost:51000/cis/servlet/StartCISPage?PAGEURL=/cisyourfirstproject/helloworld.html
```

If the page is not displayed, check the following:

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

Inside the URL, a servlet StartCISPage is invoked. As parameter `PAGEURL`, a certain value is passed: this value is built from the application project name and the page name. Calling the servlet is the generic way of opening pages which are created with Application Designer directly inside the browser.

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

10 Some Background Information

If you Change the Adapter

You can now add more functions to your “Hello World!” application by yourself. This means that you will both work with the layout description inside the Layout Painter as well as change the code of the adapter.

Changing the adapter's code requires that you replace your existing class files in the *<installdir>/tomcat/webapps/cis/cisyourfirstproject/appclasses/classes* directory by new ones. What does this mean for the runtime environment? The runtime does not see these recompiled classes because it already loaded the existing classes. So there is no difference how the application performs by default.

However, you can explicitly force the server to reload all application classes and to use the newest class version. The server is clever enough to keep old classes for existing sessions, which means that users who are already logged on will not be disturbed. But new sessions will get the newest classes.

The task to update the classes is directly triggered from the Layout Painter by choosing the save button. This tells the runtime to use new classes for new sessions. Since the preview area always opens a new session, it will be up-to-date whenever you choose the preview button.

Therefore, you can develop continuously, without restarting the GUI-server Tomcat environment.

For fast readers:

The class loader management mentioned above is designed to be used at design time. At runtime, especially if running in non-Tomcat environments, you should use the application server's web class loader to load the adapter classes. Details are provided in other sections of the Application Designer documentation.

Name Binding between Controls and Adapter

Which are the critical parts when building the “Hello World!” application?

- The PAGE control in the layout points to the class name of the adapter (property `model`).
- 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 method `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?

- The browser accesses (within the preview area of the Layout Painter) the intelligent HTML page generated from the XML layout description. The HTML page is loaded. Depending on the settings of your browser, the HTML page is picked from the page buffer of the browser after the first access.
- The page is loaded and JavaScript statements are executed to send a command to the server, requesting the data content of the page.
- The server's session management finds out that a new instance of the adapter class has to be created. It calls the default constructor - without any parameters - of the adapter class and registers it within its session management.
- The server calls all `get` methods of the adapter and collects the results in a streamed string. The string is sent back as a response to the browser.
- The page inside the browser receives the response and distributes the new content to the controls. The controls are updated using JavaScript statements working with the DOM interface.
- The user provides some input, for example, enters the name. The content change is registered inside the page.
- 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 as a streamed string and are sent to the server including the information that a method has to be called.
- The server receives the request and finds by its session management the corresponding adapter object. First, it pushes all data changes using the `set` methods into the adapter class and calls the method.
- The method changes the data inside the adapter.
- The server calls all `get` methods of the adapter and collects the result into a streamed string. The string is sent back as a response to the browser.
- 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 class.

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>/tomcat/webapps/cis/cisyourfirstproject*. The */cis* directory is the directory of the web application instance. The */cisyourfirstproject* directory is the directory that has been created for your new project

- The XML layout definition is kept in the */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 Java class files are kept in the directory */appclasses/classes*. There is also a directory */appclasses/lib* where you can find *jar* files to be imported into your application.
- In the directory */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.

The directory locations described in this section are the default locations. See also *Directories* in the *Development Workplace* documention.

Application Project Management

Use the project "cisyourfirstproject" only for your first steps. For real development, create your own projects. Working inside your own projects, you can be sure that your project files will never be touched by any Application Designer updates. Therefore, patches or enhancements to Application Designer can be installed by just copying them over the existing Application Designer installation.

Keep all your files inside the project directory during the development cycle. Even if this might work correctly: for example, do not put *jar* files into the server (Tomcat) directories, but place them into the corresponding */appclasses/lib* directory of your project.

There are some rare cases in which you have to position *jar* files explicitly inside the Tomcat environment: for example, if classes of a *jar* file access native libraries (such as DLLs) by the JNI interface. They can only be loaded once by the class loader. Because both Tomcat and Application Designer have class loaders which may load a class multiple times, you have to position these classes, for

example, into the /*tomcat*/*lib* directory. However, these are the exceptions. Do not put any classes outside of the Application Designer project directories unless there is a reason for it.

Usage of Enhanced Development Tools

You may already have noticed that there are various other tools inside the development workplace. See the *Development Workplace* documentation for detailed information on how to use these tools.