Advice for Using Application Designer During Development

This chapter contains useful configuration advices for developing applications with Application Designer. It covers the following topics:

- Do not Use the Web Application cis
- Select the Right Directory Location for Your Web Application
- Typical Steps for Upgrading Your Web Application
- Resource Files to be Shared in a Versioning System
- Integrating Application Designer into Your Build Process

Do not Use the Web Application *cis*

Do not place your development in the standard web application *cis*. Always use a parallel web application so that *cis* is the "Application Designer reference" aside your own development.

Advantage: if installing Application Designer upgrades, you are always sure that these will never directly touch your development system. You can install upgrades inside the *cis* directory, have a look at them and then explicitly decide to take them over into your development web application.

Important:
If you take over, always take over the whole reference web application, not parts of it.

Select the Right Directory Location for Your Web Application

By default, Tomcat administers its web applications within the directory *tomcat/webapps*. For each web application, there is one corresponding directory. For example, Application Designer is represented by directory *cis*.

You do not have to follow this structure. You can also decide to have your installation directory in a directory which is completely separated from your Tomcat installation.

There may be several reasons to do so:

- You may want to move the web application directory into your development project directories, i.e. you do not want to have your development objects distributed on several locations of your hard disk.
- You may want to have different Tomcat installations (for example, different versions) and use the same web application inside these Tomcat installations.

Example

Create the file *abcde.xml* in the directory *conf/Catalina/localhost*. Define the file in the following way:

```
<Context docBase="C:/Development/project/webui/webapp"
         privileged="true"
         antiResourceLocking="false"
         antiJARLocking="false">
</Context>
```

In this example, the web application *abcde* (this is the name of the file) is located in the directory *C:/Development/project/webui/webapp*. The directory that is defined by `docBase` is the web application directory which is internally structured in the following way:

```
webapp/cis
      /cisdemos
      ...
      ...
      /WEB-INF
      ...
```

Typical Steps for Upgrading Your Web Application

Software AG constantly publishes patched installation versions. With each patch, there is a brief documentation on the fixes that were done with the patch. The Application Designer build of a certain version always includes all patches of previous builds.

When installing a build, you have to follow a certain procedure in order to correctly take over the files of the new build and to save some information that you might have changed and which may get overwritten:

1. Start the installation wizard of the corresponding Application Designer build and select a temporary directory (*c:/temp/buildYYYYMMDD*) as the destination. As a consequence, the wizard will install a directory structure such as the following:

   ```
   c:/temp/buildYYYYMMDD
       /tomcat
           /webapps
               /cis
                   /cis
                   /cisdemos
                   ...
   ```

2. Make safety copies of all configuration files that Application Designer always delivers as part of the installation and that you may have changed. Application Designer offers a tool for this; see *Creating Safety Copies of Configuration Files* for more information.

3. Copy the content from the *cis* directory into your own web application.

4. Regenerate your layout definitions. You can do this

   - either by using the Layout Manager in the development workplace (open your project(s) and regenerate all layout definitions as described in the *Development Workplace* documentation under *Generating HTML Pages*) or by using Ajax Developer;

- or - and better: define a script (batch) in which you call the Application Designer class `com.softwareag.cis.gui.generate.HTMLGeneratorWholeDirectory` in order to regenerate all layout definitions of one project in one step. See also *Java API for HTML Page Mass Regeneration* in the *Development Workplace* documentation.

5. If you have defined your own style sheet using the Style Sheet Editor, proceed as described in the *Development Workplace* or Ajax Developer documentation under *Regenerating Your Own Style Sheet from the Style Sheet Template*.

Automate the Upgrade Using ANT

We strongly recommend that you use ANT for automating the upgrade of an installation. All steps (taking over the files and regenerating the layouts) can be easily defined using an ANT script. See *Using ANT for Upgrades* in which details are explained and a sample ANT script is provided.

Creating Safety Copies of the Configuration Files

In case you created your customized configuration files with the names *user_*.xml*, you simply have to make sure to keep your *user_*.xml* files. In addition, you have to make a safety copy of the file *web.xml*. For information on how to create custom configuration files with the names *user_*.xml*, see *Customizing Configuration Files*.

In case, you modified Application Designer’s default configuration files, you can create safety copies as described below.

A Configuration Manager is available for managing the core configuration files and which helps you to keep them consistent through upgrades of Application Designer. This tool is designed for "non-ANT-minded" developers. All "ANT-minded" developers should refer to the ANT way of upgrading your system.

We recommend using ANT. Nevertheless, the description of the Configuration Manager is provided below.

All configurations that are delivered with each Application Designer build and that you might modify on your own are critical when it comes to applying new builds. Use the Configuration Manager that comes with Application Designer as described below.

▶ **To create safety copies of the configuration files**

1. Invoke the demo workplace.

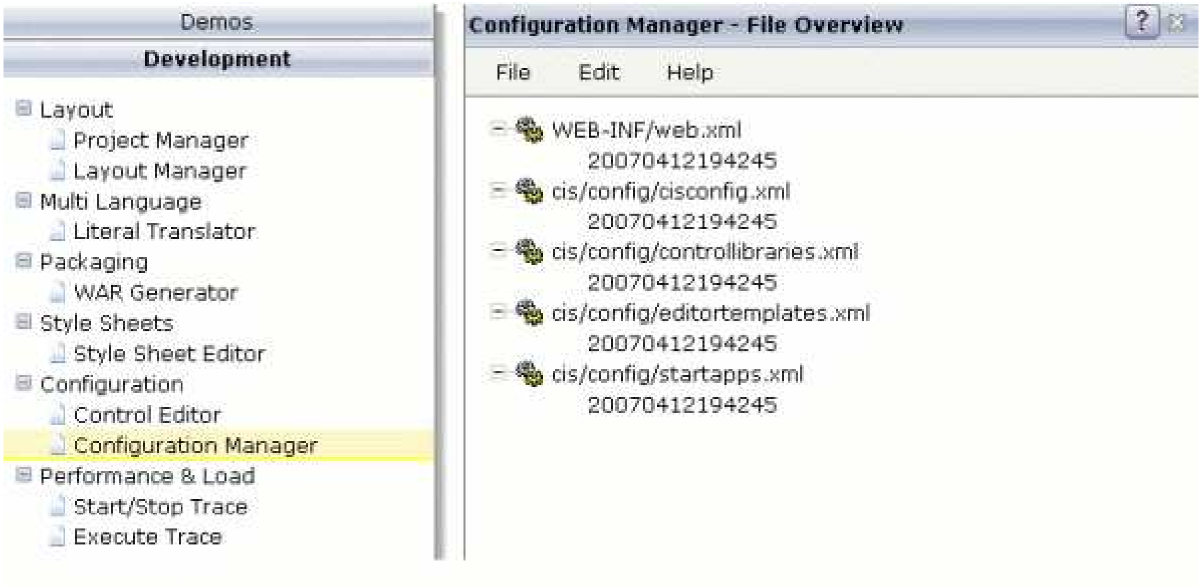
2. In the navigation frame, choose the **Development** button.

3. In the **Configuration** node, choose **Configuration Manager**.

   A list of all critical files is shown.

4. From the **File** menu, choose **Create Safety Copies**.

   The system automatically creates copies of the current files. The name of each copy includes a timestamp.

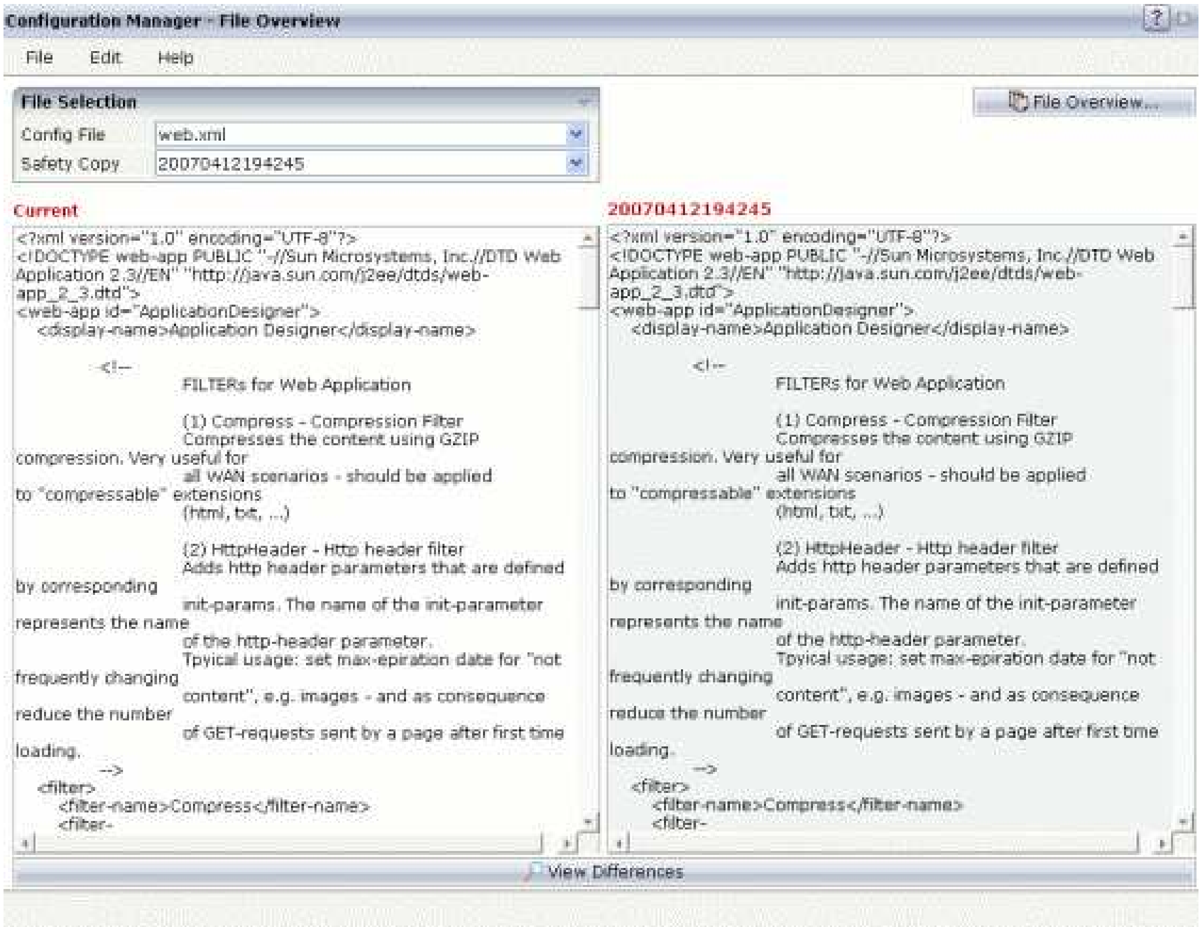


After having created the safety copies, you can, for example, upgrade your system to a new Application Designer build by copying/installing it over your existing application.

To check the differences after an upgrade

1. After the upgrade, start the Configuration Manager once again (as described above).
2. Invoke the context menu for a safety copy and choose the **Compare** command.

   The differences between the new version of the file and the saved version are shown.



Config File: web.xml

Safety Copy: 20070412194245

Current

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE web-app PUBLIC "-//Sun Microsystems, Inc.//DTD Web
Application 2.3//EN" "http://java.sun.com/j2ee/dtds/web-
app_2_3.dtd">
<web-app id="ApplicationDesigner">
   <display-name>Application Designer</display-name>

            <!--
                        FILTERs for Web Application

                        (1) Compress - Compression Filter
                        Compresses the content using GZIP
compression. Very useful for
                        all WAN scenarios - should be applied
to "compressable" extensions
                        (html, txt, ...)

                        (2) HttpHeader - Http header filter
                        Adds http header parameters that are defined
by corresponding
                        init-params. The name of the init-parameter
represents the name
                        of the http-header parameter.
                        Tpyical usage: set max-epiration date for "not
frequently changing
                        content", e.g. images - and as consequence
reduce the number
                        of GET-requests sent by a page after first time
loading.
            -->
   <filter>
      <filter-name>Compress</filter-name>
      <filter-
```

20070412194245

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE web-app PUBLIC "-//Sun Microsystems, Inc.//DTD Web
Application 2.3//EN" "http://java.sun.com/j2ee/dtds/web-
app_2_3.dtd">
<web-app id="ApplicationDesigner">
   <display-name>Application Designer</display-name>

            <!--
                        FILTERs for Web Application

                        (1) Compress - Compression Filter
                        Compresses the content using GZIP
compression. Very useful for
                        all WAN scenarios - should be applied
to "compressable" extensions
                        (html, txt, ...)

                        (2) HttpHeader - Http header filter
                        Adds http header parameters that are defined
by corresponding
                        init-params. The name of the init-parameter
represents the name
                        of the http-header parameter.
                        Tpyical usage: set max-epiration date for "not
frequently changing
                        content", e.g. images - and as consequence
reduce the number
                        of GET-requests sent by a page after first time
loading.
            -->
   <filter>
      <filter-name>Compress</filter-name>
      <filter-
```

3. You can edit the current file on the left side and take over changes from the right. Save the file by choosing **Save Current** from the **File** menu.

   Note:
   You can choose the **View Differences** button to better see what has changed.

Resource Files to be Shared in a Versioning System

When working in teams, you typically you use a central source versioning system (for example, CVS, Perforce, Visual Source Safe or others).

The resources to be synchronized are:

- **Adapter Java Sources**
  The location of the source files is up to you.

- **Layout Definitions**
  Each Application Designer project has a set of XML layout definitions that represent the layout of pages. The definitions are transferred to generated HTML by using the Application Designer development tools. The definitions are stored in the *xml* directory of a project.

- **Translation Information**
  If using the standard multi language management, literal translations are stored in comma-separated files in the project’s *multilanguage* directory.

- **Help Texts**
  If using the standard online help management, help texts are stored in the project's *help* directory.

- **Other Web Resources**
  Typically you have images and additional HTML pages that you use in your project. For example, your project has a subdirectory *images* that contains all GIF and JPG files.

- **Configuration Files**
  It may also make sense to share configuration files. For example, you may modify the *web.xml* file in order to access EJBs. In this case, you may centrally define the *web.xml* file and automatically synchronize it through the source code versioning system. The same may apply for all other configuration files.

Integrating Application Designer into Your Build Process

Typically you define the build process to consistently build up a system out of its resources. You may use simple scripts to do so, or you may use tools such as ANT.

See the Java API documentation for the following classes in order to find information on how to automate the generation of Application Designer layouts into HTML pages:

- `com.softwareag.cis.gui.generate.HTMLGenerator`

- `com.softwareag.cis.gui.generate.HTMLGeneratorWholeDirectory`