Appendix D - Class Loader Concepts

An explicit class loader management was introduced to support the following scenarios:

- Classes are automatically found in the context of Application Designer without specifying a `CLASSPATH` variable.
- Classes can be stored inside an application project directory - separated from other application projects.
- During development time, easily run new pages together with the latest classes without restarting the server.

This chapter explains the class loader concepts used inside Application Designer. It covers the following topics:

- Design Time - Runtime
- Class Loader Hierarchy
- Preparing for Runtime

Design Time - Runtime

The class loader concepts are designed to simplify the development of pages and their logical representations on the server side: adapters.

At runtime, they should only be used if you are not running in a cluster - i.e. if you do not distribute your application server on multiple nodes. When running in a cluster, classes should be located exactly there, where Java EE specifications allow them to be located. Inside the Application Designer configuration, you can select which mode you are running in - for details, see *Design Time Mode and Runtime Mode* in the *Configuration* documentation.

After explaining the class loader concepts in this chapter, at the end we explain what to do in order to change a design time environment into a runtime environment.

Class Loader Hierarchy

Application Designer runs as a web application inside a servlet engine - by default, the Tomcat servlet engine is used. The class loader used by the servlet engine is called "web application loader" in the following text.

The Application Designer environment itself is running in the context of the web application loader. This class loader is looking for classes as specified by the servlet engine. Therefore the Application Designer runtime must be accessable by this class loader. For Tomcat, this is achieved by placing the *cis.jar* file inside the *<installdir>/tomcat/webapps/ROOT/WEB-INF/lib* directory.

The following topics are covered below:

- Application Class Loader
- Initialisation of Your Application
- Guidelines for Development
- Classpath Extensions in cisconfig.xml
- Loading Resource Files

Application Class Loader

The application classes (adapter classes) are loaded by the class loader management of Application Designer. This class loader looks for Java classes as follows:

- All *.class* files inside the directory:

  <webapp>/softwareag/appclasses/classes

- All *.jar* files inside the directory:

  <webapp>/softwareag/appclasses/lib

- All *.class* files inside any application project under the directory:

  <webapp>/<project>/appclasses/classes

- All *.jar* files inside any application project under the directory:

  /<webapp>/<project>/appclasses/lib

- All classes that are referenced in the classpath extension that can be defined in the Application Designer configuration (*cisconfig.xml*).

Unlike normal class loader hierarchies, the application class loader always tries to resolve a class inside its application directories first. Only if the class is not found, the parent class loader is called - the web application loader. The benefit is that application classes are totally separated from the servlet engine classes - e.g. by using XML parser libraries. You are not bound to the parser delivered with the servlet engine.

Inside the Application Designer session management, a session is bound to an application class loader instance. Therefore the application class loader - which was instanciated when the session was created - is kept in the session during its whole life cycle. All objects created inside this session use this instance of the class loader.

In case of changing classes inside the *softwareag/appclasses* or the corresponding application-project subdirectories, you can force to create a new class loader used in all sessions which are created afterwards. This means, that you can upgrade your system without disturbing running sessions. Old sessions are still using their old classes; new sessions are using new classes.

The creation of a new instance of a class loader is triggered inside the monitoring tool. See *Monitoring* in the *Development Workplace* documentation.

By choosing the button **Use latest Version of Applications for new Sessions**, a new class loader instance is generated.

A new class loader instance can also be created during development inside the Layout Painter. See also the "Hello World!" example in the *First Steps* and its section *If you Change the Adapter*.

Initialisation of Your Application

Every time a new instance of a class loader generated, the initialisation process of your application is also performed. This guarantees that, for example, all static variables you may use internally can be correctly initialised by your initialisation procedure.

The initialisation of applications is described in the *Becoming a Member of the Startup Process* part of the *Special Development Topics*.

Guidelines for Development

The guidelines you have to follow during development are quite simple:

- Always put *all* your application/adapter classes inside the *softwareag/appclasses* directory or in the corresponding project directories. When using the project management (which is strongly recommended), store the classes in the project directories so that you can easily copy projects as self-containing units between different Application Designer installations.
- Do *not* put classes into the servlet engine’s class loader’s class path.
- Avoid class duplicates (a *.class* file in the */classes* subdirectory also contained in a *jar* file inside the */lib* subdirectory).
- Reload the classes by creating a new class loader instance. To see the effects re-logon. (The re-logon can be done by refreshing the browser.)

Classpath Extensions in *cisconfig.xml*

In the *cisconfig.xml* file, you can define the possibility to explicitly include defined directories or *jar*/*zip*/etc. files in the application class loader. The following example shows a *cisconfig.xml* file containing a class loader extension:

```
<cisconfig ...>
    <classpathextension path="c:/development/centralclasses/classes/"/>
    <classpathextension path="c:/development/centralclasses/libs/central.jar"/>
</cisconfig>
```

Consequence: you can also include classes that are located outside the web application’s directory structure into the application class loader of Application Designer.

Pay attention: if defining directories that contain *.class* files, then the path definition inside the classpath extension must end with a slash (/).

Loading Resource Files

The Application Designer application class loader does only load classes to be loaded into the Java virtual machine. It is not able to load resource files that you might access from your code.

Place resource files into the web application class loader, below the directory *<webapps>/WEB-INF/classes/* so that they are loaded in a correct way.

Preparing for Runtime

The following topics are covered below:

- Basics
- Example

Basics

As explained in the previous section, the Application Designer class loader concepts are very useful for design time purposes. What is the price? The Application Designer class loader finds its classes by accessing the file system. It uses for this reason the `cis.home` parameter inside the *<webapp>/WEB-INF/web.xml* file in order to know the file root directory of the web application.

At runtime - especially if your application server distributes the load on several physical nodes - this is dangerous: each node may have its own directory structure and you cannot specify one root directory anymore in which the web application is located.

Consequence: for running in these scenarios, you have to prepare your application accordingly - i.e. you have to place your classes at the places where the Java EE definition defines them to be located.

The normal directories to put classes in are:

- *<webapp>/WEB-INF/lib* for libraries (*.jar* files).
- *<webapp>/WEB-INF/classes* for single class files (*.class* files).

In addition, you must switch off the flag "useownclassloader" inside the *cisconfig.xml*. Consequently, the Application Designer application class loader will not be used at all - all classes are loaded by the web application loader.

Example

Example: let us assume that you have set up the Application Designer application project "projectxyz". The classes for this project are located in

- *<webapp>/projectxyz/appclasses/classes/*.class* and
- *<webapp>/projectxyz/appclasses/lib/*.jar*

so that the Application Designer class loader can reach them.

For changing to the runtime scenario, just copy the **.class* and **.jar* files from your project directory into the corresponding Java EE standard directories.