geronimo-application-client.xml

{scrollbar}

Overview

INLINE

The Geronimo deployment plan for a Java EE application, which is usually packaged as a JAR file, is called "**geronimo-application-client.xml**".

The **geronimo-application-client.xml** deployment plan is an optional file, but is typically used when deploying a client application JAR file that uses the Geronimo client application container. It is used to specify a moduleId for the deployed module, any third party dependencies, a callback handler definition, any message destinations, resources, additional GBeans, and references to any external GBeans, EJBs, services, or resources.

Packaging

The **geronimo-application-client.xml** deployment plan can be packaged as follows:

1. Embedded in an JAR file. In this case, a **geronimo-application-client.xml** file must be placed in the **/META-INF** directory of the JAR.
2. Maintained separately from the JAR file. In this case, the path to the file must be provided to the appropriate Geronimo deployer (e.g., command-line or console). Note that in this case, the filename can be named something other than **geronimo-application-client.xml** but must adhere to the same schema.
3. Embedded in an application EAR file and referenced by an **<alt-dd>** element of the EAR deployment plan.

Schema

The **geronimo-application-client.xml** deployment plan is defined by the **geronimo-application-client-2.0.xsd** schema located in the **<geronimo_home> /schema/** subdirectory of the main Geronimo installation directory. The **geronimo-application-client-2.0.xsd** schema is briefly described here:

http://geronimo.apache.org/schemas-3.0/docs/geronimo-application-client-2.0.xsd.html

Schema top-level elements

The root XML element in the **geronimo-application-client-2.0.xsd** schema is the **<application-client>** element. The top-level XML elements of the **<appli cation-client>** root element are described in the sections below. The deployment plan should always use the application client namespace, and it typically requires elements from Geronimo System, Geronimo Naming, and Geronimo Security namespaces. A typical deployment for **geronimo-application-client.xml** can be presented as follows:

xmlsolidgeronimo-web.xml Example <client:application-client xmlns:client="http://geronimo.apache.org/xml/ns/j2ee/application-client-2.0" xmlns:sys=" http://geronimo.apache.org/xml/ns/deployment-1.2" xmlns:name="http://geronimo.apache.org/xml/ns/naming-1.2" xmlns:sec="http://geronimo.apache.org /xml/ns/security-2.0" xmlns:connector="http://geronimo.apache.org/xml/ns/j2ee/connector-1.2"> ... </client:/application-client>

<sys:environment>

The **<sys:client-environment>** XML element uses the Geronimo System namespace, which is used to specify the common elements for common libraries and module-scoped services for the client application JVM, and is described here:

- http://geronimo.apache.org/schemas-3.0/docs/geronimo-module-1.2.xsd.html

The **<sys:client-environment>** element contains the following elements:

- The **<moduleId>** element is used to provide the configuration name for the web application as deployed in the Geronimo server. It contains elements for the **groupId**, **artifactId**, **version** and module **type**. Module IDs are normally printed with slashes between the four components, such as **GroupID/ArtifactID/Version/Type**.
- The **<dependencies>** element is used to provide the configurations and third party libraries on which the web module is dependent upon. These configurations and libraries are made available to the web module via the Geronimo classloader hierarchy.
- The **<hidden-classes>** element can be used to provide some degree of control of the Geronimo classloader hierarchy, and mitigate clashes between classes loaded by the server and classes loaded by the web application. It is used to lists packages or classes that may be in a parent classloader, but must not be exposed to the web application. Since Geronimo is entirely open-source and utilizes many other open-source libraries it is possible that the server itself and the web application may have different requirements and/or priorities for the same open source project libraries. The **<hidden-classes>** element is typically used when the web application has requirements for a specific version of a library that is different than the version used by Geronimo itself. A simple example of this is when a web application uses, and most importantly includes, a version of the **Log4J** common logging library that is different than the version used by the Geronimo server itself. This might not provide the desired results. Thus, the **<hidden-classes>** element can be used to "hide" the Log4J classes loaded by all the parent classloaders of the web application module, including those loaded by and for the Geronimo server itself, and only the Log4J classes included with the web application library will get loaded.

- The **<non-overridable-classes>** element can also be used to provide some degree of control of the Geronimo classloader hierarchy, but in the exact opposite manner than provided by the **<hidden-classes>** element. This element can be used to specify a list of classes or packages which will **only** be loaded from the parent classloader of the web application module to ensure that the Geronimo server's version of a libary is used instead of the version included with the web application.

- The **<inverse-classloading>** element can be used to specify that standard classloader delegation is to be reversed for this module. The Geronimo classloader delegation follows the Java EE 5 specifications, and the normal behavior is to load classes from a parent classloader (if available) before checking the current classloader. When the **<inverse-classloading>** element is used, this behavior is reversed and the current classloader will always be checked before looking in the parent classloader(s). This element is similar to the **<hidden-classes>** element since the desired behavior is to give the libraries packaged with the web application (i.e., in WEB-INF/lib) precedence over anything used by the Geroimo server itself.

- The **<suppress-default-environment>** element can be used to suppress inheritance of environment by module (i.e., any default environment built by a Geronimo builder when deploying the plan will be suppressed). If the **<suppress-default-environment>** element is specified then any default environment build by a builder when deploying the plan will be suppressed. An example of where this is useful is when deploying a connector on an app client in a separate (standalone) module (not as part of a client plan). The connector builder defaultEnvironment includes some server modules that won't work on an app client, so you need to suppress the default environment and supply a complete environment including all parents for a non-app-client module you want to run on an app client. This element should not be used for application clients however.

<sys:server-environment>

The **<sys:environment>** XML element uses the Geronimo System namespace, which is used to specify the common elements for common libraries and module-scoped services, and is described here:

- http://geronimo.apache.org/schemas-3.0/docs/geronimo-module-1.2.xsd.html

The **<sys:server-environment>** element contains the following elements:

- The **<moduleId>** element is used to provide the configuration name for the web application as deployed in the Geronimo server. It contains elements for the **groupId**, **artifactId**, **version** and module **type**. Module IDs are normally printed with slashes between the four components, such as **GroupID/ArtifactID/Version/Type**.

- The **<dependencies>** element is used to provide the configurations and third party libraries on which the web module is dependent upon. These configurations and libraries are made available to the web module via the Geronimo classloader hierarchy.

- The **<hidden-classes>** element can be used to provide some degree of control of the Geronimo classloader hierarchy, and mitigate clashes between classes loaded by the server and classes loaded by the web application. It is used to lists packages or classes that may be in a parent classloader, but must not be exposed to the web application. Since Geronimo is entirely open-source and utilizes many other open-source libraries it is possible that the server itself and the web application may have different requirements and/or priorities for the same open source project libraries. The **<hidden-classes>** element is typically used when the web application has requirements for a specific version of a library that is different than the version used by Geronimo itself. A simple example of this is when a web application uses, and most importantly includes, a version of the **Log4J** common logging library that is different than the version used by the Geronimo server itself. This might not provide the desired results. Thus, the **<hidden-classes>** element can be used to "hide" the Log4J classes loaded by all the parent classloaders of the web application module, including those loaded by and for the Geronimo server itself, and only the Log4J classes included with the web application library will get loaded.

- The **<non-overridable-classes>** element can also be used to provide some degree of control of the Geronimo classloader hierarchy, but in the exact opposite manner than provided by the **<hidden-classes>** element. This element can be used to specify a list of classes or packages which will **only** be loaded from the parent classloader of the web application module to ensure that the Geronimo server's version of a libary is used instead of the version included with the web application.

- The **<inverse-classloading>** element can be used to specify that standard classloader delegation is to be reversed for this module. The Geronimo classloader delegation follows the Java EE 5 specifications, and the normal behavior is to load classes from a parent classloader (if available) before checking the current classloader. When the **<inverse-classloading>** element is used, this behavior is reversed and the current classloader will always be checked before looking in the parent classloader(s). This element is similar to the **<hidden-classes>** element since the desired behavior is to give the libraries packaged with the web application (i.e., in WEB-INF/lib) precedence over anything used by the Geroimo server itself.

- The **<suppress-default-environment>** element can be used to suppress inheritance of environment by module (i.e., any default environment built by a Geronimo builder when deploying the plan will be suppressed). If the **<suppress-default-environment>** element is specified then any default environment build by a builder when deploying the plan will be suppressed. An example of where this is useful is when deploying a connector on an app client in a separate (standalone) module (not as part of a client plan). The connector builder defaultEnvironment includes some server modules that won't work on an app client, so you need to suppress the default environment and supply a complete environment including all parents for a non-app-client module you want to run on an app client. This element should not be used for applications clients however.

xml<sys:client-environment> and < sys:server-environment> examplesolid <application-client xmlns=http://geronimo.apache.org/xml/ns/j2ee/application-client-2.0 xmlns:dep="http://geronimo.apache.org/xml/ns/deployment-1.2"> <dep:client-environment> <dep:moduleId> <dep:groupId>JavaEE</dep:groupId> <dep:artifactId>EXAMPLEClient</dep:artifactId> <dep:version>3.0</dep:version> <dep:type>car</dep:type> </dep:moduleId> </dep:client-environment> <dep:server-environment> <dep:moduleId> <dep:groupId>JavaEE</dep:groupId> <dep:artifactId>EXAMPLEClientServer</dep:artifactId> <dep:version>3.0</dep:version> <dep:type>car</dep:type> </dep:moduleId> </dep:server-environment> </application-client>

<naming:gbean-ref>

The **<naming:gbean-ref>** XML element uses the Geronimo Naming namespace, which is used to identify the common elements for resolving GBean references, resource references, and Web services references, and is described here:

- http://geronimo.apache.org/schemas-3.0/docs/geronimo-naming-1.2.xsd.html

It is used to map GBean references to GBeans in other applications using the ref-name provided in the GBean deployment descriptor.

<naming:ejb-ref>

The **<naming:ejb-ref>** XML element uses the Geronimo Naming namespace, which is used to identify the common elements for resolving EJB references, resource references, and Web services references, and is described here:

- http://geronimo.apache.org/schemas-3.0/docs/geronimo-naming-1.2.xsd.html

It is used to map EJB references to EJB's in other applications using remote home and remote interface. The application which contains the EJB being referenced should either be in same EAR or should be included in dependency list of this application. Also note as the EJB's referenced are in a different JVM all the Client interfaces should also be included in current application.

<naming:service-ref>

The **<naming:service-ref>** XML element uses the Geronimo Naming namespace, which is used to identify the common elements for resolving EJB references, resource references, and Web services references, and is described here:

- http://geronimo.apache.org/schemas-3.0/docs/geronimo-naming-1.2.xsd.html

It is used to map service references to service's in other applications. The application which contains the EJB being referenced should either be in same EAR or should be included in dependency list of this application.

<naming:resource-ref>

The **<naming:resource-ref>** XML element uses the Geronimo Naming namespace, which is used to identify the common elements for resolving EJB references, resource references, and Web services references, and is described here:

- http://geronimo.apache.org/schemas-3.0/docs/geronimo-naming-1.2.xsd.html

It is used to map resource references to resources's like JDBC resources, JMS resources, etc. configured outside the current application.

<naming:resource-env-ref>

The **<naming:resource-env-ref>** XML element uses the Geronimo Naming namespace, which is used to identify the common elements for resolving EJB references, resource references, and Web services references, and is described here:

- http://geronimo.apache.org/schemas-3.0/docs/geronimo-naming-1.2.xsd.html

It is used to map resource references to administrative objects deployed as a part of connectors.

<naming:message-destination>

The **<naming:message-destination>** XML element uses the Geronimo Naming namespace, which is used to identify the common elements for resolving EJB references, resource references, and Web services references, and is described here:

- http://geronimo.apache.org/schemas-3.0/docs/geronimo-naming-1.2.xsd.html

It is used to configure a JMS queue or topic which acts like a destination for the messages delivered.

<sec:default-subject>

The **<sec:default-subject>** element uses the Geronimo deployment namespace described here:

- http://geronimo.apache.org/schemas-3.0/docs/geronimo-security-2.0.xsd.html.

<realm-name>

The **<realm-name>** element uses the Geronimo deployment namespace described here:

- http://geronimo.apache.org/schemas-3.0/docs/geronimo-application-client-2.0.xsd.html

It names the security realm used for JAAS login.

<callback-handler>

The **<resource>** element uses the Geronimo deployment namespace described here:

- http://geronimo.apache.org/schemas-3.0/docs/geronimo-application-client-2.0.xsd.html

It specifies the name of a callback class provided by the application for JAAS authentication. This class must implement the javax.security.auth.callback.CallbackHandler interface and follow its specification, as this class will be used by the application client container to collect authentication information from the user.

<resource>

The **<resource>** element uses the Geronimo deployment namespace described here:

- http://geronimo.apache.org/schemas-3.0/docs/geronimo-application-client-2.0.xsd.html

It contains the definition of all the module-scoped connector resources. The connector resource can be both external and internal to the application client.

<sys:service>

The **<sys:service>** element uses the Geronimo deployment namespace described here:

- http://geronimo.apache.org/schemas-3.0/docs/geronimo-module-1.2.xsd.html

It is used to define GBean(s) that are configured and deployed with the application client. These additional Geronimo services will be deployed when the application is deployed (and stopped when the application is stopped). Normally, the implementation classes for these services are included at the server level and referenced using a dependency element.