geronimo-application.xml

{scrollbar}

Overview

INLINE

The Geronimo deployment plan for an enterprise application, which is usually packaged as a EAR file, is called "**geronimo-application.xml**".

The **geronimo-application.xml** deployment plan is used in conjunction with the **application.xml** JAVA EE deployment plan to deploy enterprise applications consisting of Web Application WAR(s), EJB JAR(s), client application JAR(s), JCA connector JAR(s), and resource adapter module RAR(s) to the Geronimo application server. The **geronimo-application.xml** deployment plan is an optional file, but is typically used when deploying an EAR file. It is used to specify a moduleId for the deployed module, any third party dependencies, security options, and services.

Packaging

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

1. Embedded in an EAR file. In this case, a **geronimo-application.xml** file must be placed in the **/META-INF** directory of the EAR, which is the same place where the **application.xml** file must be located.
2. Maintained separately from the EAR 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.xml** but must adhere to the same schema.

Schema

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

http://geronimo.apache.org/schemas-2.1/docs/geronimo-application-2.0.xsd.html

Schema top-level elements

The root XML element in the **geronimo-application-2.0.xsd** schema is the **<application>** element. The top-level XML elements of the **<application>** root element are described in the sections below. The deployment plan should always use the Web application namespace, and it typically requires elements from Geronimo System and Geronimo Security namespaces. Additionally, it has an attribute to identify its name. A typical deployment for **geronimo-application.xml** can be presented as follows:

xmlgeronimo-application.xml Examplesolid <app:application xmlns:app="http://geronimo.apache.org/xml/ns/j2ee/application-2.0" xmlns:sys="
http://geronimo.apache.org/xml/ns/deployment-1.2" xmlns:sec="http://geronimo.apache.org/xml/ns/security-2.0" application-name="
SampleApplicationName"> ... </app:application>

<sys: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-2.1/docs/geronimo-module-1.2.xsd.html

The **<sys: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 built 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 however.

An example **geronimo-application.xml file** is shown below using the **<sys:environment>** elements:

xmlsolid<sys:environment> example <app:application xmlns:conn="http://geronimo.apache.org/xml/ns/j2ee/connector-1.2" xmlns:dep="http://geronimo.apache.org/xml/ns/deployment-1.2" xmlns:app="http://geronimo.apache.org/xml/ns/j2ee/application-2.0" application-name="SampleEAR"> <dep:environment> <dep:moduleId> <dep:groupId>sampleear</dep:groupId> <dep:artifactId>sample-ear</dep:artifactId> <dep:version>1.0</dep:version> <dep:type>ear</dep:type> </dep:moduleId> <dep:dependencies> <dep:dependency> <dep:groupId>org.apache.geronimo.configs</dep:groupId> <dep:artifactId>tomcat6</dep:artifactId> <dep:version>2.2-SNAPSHOT</dep:version> <dep:type>car</dep:type> </dep:dependency> <dep:dependency> <dep:groupId>default</dep:groupId> <dep:artifactId>geronim-web-4</dep:artifactId> <dep:version>1.0</dep:version> <dep:type>car</dep:type> </dep:dependency> </dep:dependencies> </dep:environment> </app:application>

<module>

The **<module>** uses the Geronimo default namespace for a geronimo-application.xml file that is described here:

- http://geronimo.apache.org/schemas-2.1/docs/geronimo-application-2.0.xsd.html

This element is used to define a single Java EE module and contains one of the following elements to identify the module:

- **<connector>** holds the location of a Java EE Connector module. Must match the same element in **application.xml**.

- **<ejb>** holds the location of an EJB module. Must match the same element in **application.xml**.

- **<java>** holds the location of a client application module. Must match the same element in **application.xml**.

- **<web>** holds the location of a web application module. Must match **<web-uri>** in **application.xml**.

The <module> element also contains an optional <alt-dd> element that specifies an optional URI to the post-assembly version of the deployment descriptor file for a particular Java EE module. If <alt-dd> is not specified, the deployer must read the deployment descriptor from the default location and file name required by the respective component specification.

xmlsolid<module> example <application xmlns="http://geronimo.apache.org/xml/ns/j2ee/application-2.0" xmlns:sys="http://geronimo.apache.org/xml/ns/deployment-1.2" application-name="MDBSampleEAR"> <module> <ejb>example-ejb.jar</ejb> <alt-dd>dds/my-ejb-geronimo-plan.xml</alt-dd> </module> <module> <web> <web-uri>example-web.war</web-uri> <context-root>example</context-root> </web> </module> </application>

Then the Geronimo deployment plan could include separate deployment plans for both modules like this:

xmlsolid<module> example <application xmlns="http://geronimo.apache.org/xml/ns/j2ee/application-2.0" xmlns:sys="http://geronimo.apache.org/xml/ns/deployment-1.2" application-name="MDBSampleEAR"> <module> <ejb>example-ejb.jar</ejb> <alt-dd>dds/my-ejb-jar.xml</alt-dd> </module> <module> <web> <web-uri>example-web.war</web-uri> <web-app xmlns="http://geronimo.apache.org/xml/ns/j2ee/web-1.0" xmlns:naming="http://geronimo.apache.org/xml/ns/naming-1.0"> <naming:resource-ref> <naming:ref-name>jms/AConnectionFactory</naming:ref-name> <naming:resource-link>MyConnectionFactory</naming:resource-link> </naming:resource-ref> </web-app> </web> </module> </application>

In this case, the EJB deployment plan is stored inside the EAR (in a directory called dds/ that also holds a replacement for ejb-jar.xml for that module). The Web application deployment plan is right there inside the EAR deployment plan, in its entirety.

<ext-module>

The **<ext-module>** uses the Geronimo default namespace for a geronimo-application.xml file that is described here:

- http://geronimo.apache.org/schemas-2.1/docs/geronimo-application-2.0.xsd.html

This element can be used to define a single external Jave EE module that is being deployed or redeployed and contains one of the following elements to identify the module:

- **<connector>** holds the location of a Java EE Connector module. Must match the same element in **application.xml**.

- **<ejb>** holds the location of an EJB module. Must match the same element in **application.xml**.

- **<java>** holds the location of a client application module. Must match the same element in **application.xml**.
- **<web>** holds the location of a web application module. Must match **<web-uri>** in **application.xml**.

The <ext-module> element also contains either an <internal-path> or an **<external-path>**. **<internal-path>** indicates that the module is packaged in the EAR and the path specified is relative to the enterprise application package main directory. **<external-path>** indicates that the module is not part of the enterprise application and must be located by matching the supplied pattern in a Geronimo repository.

xmlsolid<ext-module> example <application xmlns="http://geronimo.apache.org/xml/ns/j2ee/application-2.0" xmlns:sys="http://geronimo.apache.org/xml/ns/deployment-1.2" application-name="MDBSampleEAR"> <sys:environment> <sys:moduleId> <sys:groupId>default</sys:groupId> <sys:artifactId>MDBSampleEAR</sys:artifactId> <sys:version>1.0</sys:version> <sys:type>car</sys:type> </sys:moduleId> </sys:environment> <ext-module> <connector>TopicJMSSample</connector> <external-path> <sys:groupId>org.apache.geronimo.modules</sys:groupId> <sys:artifactId>geronimo-activemq-ra</sys:artifactId> <sys:version>2.1</sys:version> </external-path> <connector xmlns="http://geronimo.apache.org/xml/ns/j2ee/connector-1.2"> <resourceadapter> <!--how to connect to the JMS Server--> <resourceadapter-instance> <resourceadapter-name>TradeJMSResources</resourceadapter-name> <config-property-setting name="ServerUrl">tcp://localhost:61616</config-property-setting> <config-property-setting name="UserName">not needed</config-property-setting> <config-property-setting name="Password">not needed</config-property-setting> <workmanager> <gbean-link>DefaultWorkManager</gbean-link> </workmanager> </resourceadapter-instance> <!--defines a ConnectionFactory--> <outbound-resourceadapter> <connection-definition> <connectionfactory-interface>javax.jms.ConnectionFactory</connectionfactory-interface> <connectiondefinition-instance> <name>jms/TopicConnectionFactory</name> <implemented-interface>javax.jms.TopicConnectionFactory</implemented-interface> <connectionmanager> <xa-transaction> <transaction-caching/> </xa-transaction> <single-pool> <max-size>10</max-size> <min-size>0</min-size> <blocking-timeout-milliseconds>5000</blocking-timeout-milliseconds> <idle-timeout-minutes>0</idle-timeout-minutes> <match-one/> </single-pool> </connectionmanager> </connectiondefinition-instance> </connection-definition> </outbound-resourceadapter> </resourceadapter> <adminobject> <adminobject-interface>javax.jms.Topic</adminobject-interface> <adminobject-class>org.activemq.message.ActiveMQTopic</adminobject-class> <adminobject-instance> <message-destination-name>TextMessageTopic</message-destination-name> <config-property-setting name="PhysicalName">TextMessageTopic</config-property-setting> </adminobject-instance> </adminobject> </connector> </ext-module> </application>

<sec:security>

The **<sec:security>** uses the Geronimo applicaiton namespace described here:

- http://geronimo.apache.org/schemas-2.1/docs/geronimo-application-2.0.xsd.html

It is used to map roles specified in the EAR file to roles or principals in the security realm that will be used when deploying the module. If this optional element is present, all web and EJB modules must make the appropriate access checks as outlined in the JACC spec. This element groups the security role mapping settings for the application. **<app:security>** contains the following elements:

- The **<doas-current-caller>** optional element may be set to true or false (default). If set to true, any work done by the application will be performed as the calling Subject, instead of "as the application server". This can be used to hook into the Java JVM security sandbox (for example, to only allow trusted users to access the server filesystem). It is not ususally necessary, as the application-level security features are typically sufficient. When it is enabled, you may want to adjust the security policy used for the server to control certain permissions by subject.
- The **<use-context-handler>** optional element may be set to true or false (default). If set to true, the installed JACC policy contexts will use PolicyContextHandlers.
- The **<default-role>** element is used by the the Deployer to assign method permissions for all of the unspecified methods, either by assigning them to security roles, or by marking them as unchecked. If the value of default-role is empty, then the unspecified methods are marked unchecked.
- The **<description>** element holds the description.
- The **<credential-store-ref>** element holds the pattern for matching a module.
- The **<default-subject>** element provides a description, realm, and id.
- The **<role-mappings>** element holds the information mapping roles declared in the **application.xml** deployment descriptor to specific principals present in the security realms available to Geronimo.

xmlsolid<sec:security> example <application xmlns="http://geronimo.apache.org/xml/ns/j2ee/application-2.0" xmlns:sys="http://geronimo.apache.org/xml/ns/deployment-1.2" xmlns:sec="http://geronimo.apache.org/xml/ns/security-2.0" application-name="SampleSecurityEAR"> <dep:environment> <dep:moduleId> <dep:groupId>sampleear</dep:groupId> <dep:artifactId>sample-ear</dep:artifactId> <dep:version>1.0</dep:version> <dep:type>ear</dep:type> </dep:moduleId> <dep:dependencies> <dep:dependency> <dep:groupId>org.apache.geronimo.configs</dep:groupId> <dep:artifactId>tomcat6</dep:artifactId> <dep:version>2.2-SNAPSHOT</dep:version> <dep:type>car</dep:type> </dep:dependency> <dep:dependency> <dep:groupId>default</dep:groupId> <dep:artifactId>geronim-web-4</dep:artifactId> <dep:version>1.0</dep:version> <dep:type>car</dep:type> </dep:dependency> </dep:dependencies> </dep:environment> <sec:security use-context-handler="false" doas-current-caller="true" default-role="admin1"> <sec:role-mappings> <sec:role role-name="admin-role"> <sec:description>ability to do everything</sec:description> </sec:role> <sec:role role-name="user-role"> <sec:description>limited access</sec:description> </sec:role> </sec:role-mappings> </sec:security> </application>

<sys:service>

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

- http://geronimo.apache.org/schemas-2.1/docs/geronimo-module-1.2.xsd.html

It is used to define GBean(s) that are configured and deployed with the application. 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.