geronimo-ra.xml

{scrollbar}

Overview

Deploying a resource adapter in Geronimo requires a Geronimo plan. This may be an external plan or may be packed in the resource adapter rar file (NOT a jar file inside the rar file) as "**META-INF/geronimo-ra.xml**". The deployment plan is used in conjunction with the **ra.xml** Java EE deployment plan to deploy JCA connector RAR(s) to the Geronimo application server. It is used to specify a moduleId for the deployed module, any third party dependencies, outbound connection pooling parameters, overridden config-properties, admin objects such as jms destinations, and additional GBeans.

Packaging

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

1. Embedded in an JAR file. In this case, a **geronimo-ra.xml** file must be placed in the **/META-INF** directory of the JAR, which is the same place where the **ra.xml** file must be located.
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-ra.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-ra.xml** deployment plan is defined by the **geronimo-connector-1.2.xsd** schema located in the **<geronimo_home>/schema/** subdirectory of the main Geronimo installation directory. The **geronimo-connector-1.2.xsd** schema is briefly described here:

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

Schema top-level elements

The root XML element in the **geronimo-connector-1.2.xsd** schema is the **<connector>** element. The top-level XML elements of the **<connector>** root element are described in the sections below. The deployment plan should always use the Connector namespace, and it typically requires elements from Geronimo System namespace. A typical deployment for **geronimo-ra.xml** can be presented as follows:

xmlsolidgeronimo-ra.xml Example <conn:connector xmlns:conn="http://geronimo.apache.org/xml/ns/j2ee/connector-1.2" xmlns:dep="http://geronimo.apache.org/xml/ns/deployment-1.2"> ... </conn:connector>

<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-3.0/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 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.

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

xml<sys:environment> examplesolid <conn:connector xmlns:dep="http://geronimo.apache.org/xml/ns/deployment-1.2" xmlns:conn="http://geronimo.apache.org/xml/ns/j2ee/connector-1.2"> <dep:environment> <dep:moduleId> <dep:groupId>connector</dep:groupId> <dep:artifactId>ConnectorProj</dep:artifactId> <dep:version>1.0</dep:version> <dep:type>jar</dep:type> </dep:moduleId> <dep:dependencies> <dep:dependency> <dep:groupId>org.apache.geronimo.configs</dep:groupId> <dep:artifactId>sharedlib</dep:artifactId> <dep:type>car</dep:type> </dep:dependency> </dep:dependencies> </dep:environment> </conn:connector>

<resourceadapter>

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

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

This element is used to define a single JDBC connector or JMS connection factory. The **<resourceadapter-instance>** element provides resource adapter instance specific information like configuration properties and workmanager implementation. The **<outboundresource-adapter>** specifies information about an outbound resource adapter. The information includes fully qualified names of classes and interfaces required as part of the connector architecture specified contracts for connection management, level of transaction support provided, one or more authentication mechanisms supported and additional required security permissions. If there is no authentication mechanism specified as part of the resource adapter element, then the resource adapter does not support any standard security contract. The application server ignores the security part of the system contracts in this case.

xml<resourceadapter> examplesolid <connector xmlns="http://geronimo.apache.org/xml/ns/j2ee/connector-1.2"> <dep:environment xmlns:dep="http://geronimo.apache.org/xml/ns/deployment-1.2"> <dep:moduleId> <dep:groupId>console.dbpool</dep:groupId> <dep:artifactId>AuthorConnectionsPool</dep:artifactId> <dep:version>1.0</dep:version> <dep:type>rar</dep:type> </dep:moduleId> <dep:dependencies> <dep:dependency> <dep:groupId>org.apache.derby</dep:groupId> <dep:artifactId>derby</dep:artifactId> <dep:version>10.1.1.0</dep:version> <dep:type>jar</dep:type> </dep:dependency> </dep:dependencies> </dep:environment> <resourceadapter> <outbound-resourceadapter> <connection-definition> <connectionfactory-interface>javax.sql.DataSource</connectionfactory-interface> <connectiondefinition-instance> <name>AuthorConnectionsPool</name> <config-property-setting name="Password">APP</config-property-setting> <config-property-setting name="Driver">org.apache.derby.jdbc.EmbeddedDriver</config-property-setting> <config-property-setting name="UserName">APP</config-property-setting> <config-property-setting name="ConnectionURL">jdbc:derby:wroxauthors</config-property-setting> <connectionmanager> <local-transaction/> <single-pool> <max-size>10</max-size> <min-size>0</min-size> <match-one/> </single-pool> </connectionmanager> </connectiondefinition-instance> </connection-definition </outbound-resourceadapter> </resourceadapter> </connector>

<adminobject>

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

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

This element can be used to define a JMS topic or queue. The **<adminobject>** contains the following elements:

- The **<adminobject-interface>** element is used to specify the fully qualified name of the implemented Java interface of the admin object. One example of this is javax.jms.Topic.

- The **<adminobject-class>** element specifies the full qualified name of the Java class of the admin object.

- The **<adminobject-intstance>** element contains the configuration for this specific instance of the administered object type, with a unique name, and values for any configuration properties necessary for that administered object type. Two elements for defining the instance of the admin object are provided. The **<message-destination-name>** element can be referred to by other deployment plans by using the **<naming:message-destination>** element. This is also used as a unique object name of the GBean for the instance. The **<config-property-setting>** specifies the set of properties for the admin object instance.

xml<adminobject> examplesolid <connector xmlns="http://geronimo.apache.org/xml/ns/j2ee/connector-1.2"> <dep:environment xmlns:dep="http://geronimo.apache.org/xml/ns/deployment-1.2"> <dep:moduleId> <dep:groupId>console.dbpool</dep:groupId> <dep:artifactId>AuthorConnectionsPool</dep:artifactId> <dep:version>1.0</dep:version> <dep:type>rar</dep:type> </dep:moduleId> <dep:dependencies> <dep:dependency> <dep:groupId>org.apache.derby</dep:groupId> <dep:artifactId>derby</dep:artifactId> <dep:version>10.1.1.0</dep:version> <dep:type>jar</dep:type> </dep:dependency> </dep:dependencies> </dep:environment> <resourceadapter> <inbound-resourceadapter> <messageadapter> <messagelistener> <messagelistener-type>javax.jms.MessageListener</messagelistener-type> <activationspec> <activationspec-class>org.apache.activemq.ra.ActiveMQActivationSpec</activationspec-class> <required-config-property> <config-property-name>destination</config-property-name> </required-config-property> <required-config-property> <config-property-name>destinationType</config-property-name> </required-config-property> </activationspec> </messagelistener> </messageadapter> </inbound-resourceadapter> <adminobject> <adminobject-interface>javax.jms.Queue</adminobject-interface> <adminobject-class>org.apache.activemq.command.ActiveMQQueue</adminobject-class> <config-property> <config-property-

name>PhysicalName</config-property-name> <config-property-type>java.lang.String</config-property-type> </config-property> </adminobject> <adminobject> <adminobject-interface>javax.jms.Topic</adminobject-interface> <adminobject-class>org.apache.activemq.command.ActiveMQTopic</adminobject-class> <config-property> <config-property-name>PhysicalName</config-property-name> <config-property-type>java.lang.String</config-property-type> </config-property> </adminobject> </resourceadapter> </connector>

<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 connector module. 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.