ContextDeployer

Context Deployer

The ContextDeployer may be used to (hot)deploy an arbitrary Context or Web Application with Jetty specific configuration. To statically deploy only standard web applications at startup, use the WebAppDeployer.

Overview

Typically a ContextDeployer is defined in a jetty.xml file:

```
<Call name="addLifeCycle">
  <Arg>
    <New
class="org.mortbay.jetty.deployer.ContextDep
loyer">
      <Set name="contexts"><Ref
id="Contexts"/></Set>
      <Set
name="configurationDir"><SystemProperty
name="jetty.home"
default="."/>/contexts</Set>
      <Set name="scanInterval">1</Set>
    </New>
  </Arg>
</Call>
```

The ContextDeployer will scan the **configurationDir** directory at intervals of **scanInterval** seconds for xml descriptors that define contexts. Any contexts found are deployed to the passed **contexts** reference to a HandlerContainer (this is normally an instance of ContextHandlerCollection).

The deployment descriptors are in jetty xml format and define and configure individual contexts. A minimal example is:

```
<?xml version="1.0"  encoding="ISO-8859-1"?>
<!DOCTYPE Configure PUBLIC "-//Mort Bay
Consulting//DTD Configure//EN"
"http://jetty.mortbay.org/configure.dtd">
<Configure
class="org.mortbay.jetty.webapp.WebAppContex
t">
  <Set name="contextPath">/test</Set>
  <Set name="war"><SystemProperty
name="jetty.home"
default="."/>/webapps/test</Set>
</Configure>
```

This example creates an instance of org.mortbay.jetty.webapp.WebAppContext and sets the **contextPath** to be "/test" and the **resourceBase** to be "$jetty.home/webapps/test". Because the context used is a standard web application context, when started it will inspect the resourceBase for a WEB-INF/web.xml for further configuration.

The ContextDeployer is added to the server as a LifeCycle. This simply means that the deployer will be started and stopped with the server. Ie when server.start() is called, then start will also be called on the deployer.

Property value substitution

The ContextDeployer can automatically do property substitution on the context files that it deploys. You define a ConfigurationManager that manages the properties and pass this into the ContextDeployer. There is currently one implementation of the ConfigurationManager, and that is the `org.mortbay.jetty.deployer.FileConfigurationManager` that reads a properties file and makes available the property values to the ContextDeployer. Here's how you would configure the ContextDeployer:

```
<Call name="addLifeCycle">
  <Arg>
    <New
class="org.mortbay.jetty.deployer.ContextDep
loyer">
      <Set name="contexts"><Ref
id="Contexts"/></Set>
      <Set
name="configurationDir"><SystemProperty
name="jetty.home"
default="."/>/contexts</Set>
      <Set name="scanInterval">1</Set>
      <Set name="configurationManager">
        <New
class="org.mortbay.jetty.deployer.FileConfig
urationManager">
          <Set name="file"><SystemProperty
name="jetty.home"
default="."/>/foo.properties</Set>
        </New>
      </Set>
    </New>
  </Arg>
</Call>
```

Here's an example of the contents of the foo.properties file:

```
foo = /funkyapp
```

Here's how you would use this in a context xml file:

```
<?xml version="1.0"  encoding="ISO-8859-1"?>
<!DOCTYPE Configure PUBLIC "-//Mort Bay
Consulting//DTD Configure//EN"
"http://jetty.mortbay.org/configure.dtd">
<Configure
class="org.mortbay.jetty.webapp.WebAppContex
t">
  <Set name="contextPath"><Property
name="foo"/></Set>
  <Set name="war"><SystemProperty
name="jetty.home"
default="."/>/webapps/test</Set>
</Configure>
```

Contexts

Because the class of the context is defined in the Configure clause, any type of ContextHandler may be deployed with this mechanism, included base ContextHandlers, servlet Contexts, WebAppContexts or any class derived from them.

Hot deploy

If the scan interval is non-zero, the configuration directory is scanned at that interval (in seconds) for changes to the deployment descriptors. If a descriptor is added to the directory, the new context will be deployed. If a descriptor is touched/updated then it's context will be stopped, reconfigured and redeployed. If a descriptor is removed, then it's context will be stopped and removed from the server.

If the scan interval is zero, then the directory is only scanned at startup.

Other things to configure

This mechanism allows the most of the API available on ContextHandler or its derived classes to be called to configure the web application. Useful things to configure include:

- setAttribute
- setClassLoader
- setContextPath
- setVirtualHosts
- addServlet
- setConfigurations

- setExtraClassPath
- setDefaultsDescriptor
- setDescriptor
- setOverrideDescriptor
- setSystemClasses
- setServerClasses