Spring, XBean, Servlets and more

Overview

You've probably skipped over the rest of the user's guide and landed here because you're only interested in Spring configuration, right? If so, read the Quick Start section before you dive into how to leverage Spring container management. It outlines, from start to finish, writing, configuring and exposing a simple SOAP Web service. More importantly, although the section doesn't say it, it is actually using the Spring framework under the hood, making it a very good starting point.

Using XBean and Spring

XFire uses a great project called XBean under the covers for its XML Configuration. XBean allows us to create a custom syntax and be able to intermix it with the spring syntax. Before you continue you should make sure you have Spring 1.2.4+. Now lets take a look at some examples.

Here is an example of how to declare a simple service:

And you can easily load it via XBean's Classpath application context:

As you can see, that registered our service easily enough. From here we can also intermix the Spring and XFire syntax:

To set properties on your Service you can do this:

```
<service>
  ... define your normal attributes ...
  <properties>
    <property key="mtom-enabled">true</property>
    <property key="myProperty">myValue</property>
  <properties>
</service>
```

Using XFire without XBean: The ServiceBean

The above examples with XBean are really just using the ServiceBean class behind the scenes. If you don't want to use the XBean ApplicationContext you can easily configure your service via the ServiceBean class:

You'll also want to import some standard XFire bean definitions from the classpath:

```
<import resource="classpath:org/codehaus/xfire/spring/xfire.xml"/>
```

This file contains things like definitions for the TransportManager, ServiceRegistry, and some simple ServiceFactorys.

If you're doing services over HTTP, you'll want to take a look at the XFireSpringServlet to expose these services.

To set properties on your Service you can do this:

```
<bean class="org.codehaus.xfire.spring.ServiceBean">
  ... define your normal attributes ...
  <property name="properties">
    <map>
      <entry>
        <key><value>mtom-enabled</value></key>
        <value>true</value>
      </entry>
      <entry>
        <key><value>myProperty</value></key>
        <value>myValue</value>
      </entry>
    </map>
  <property>
</bean>
```

Configuring the XFireSpringServlet

The XFireSpringServlet to expose your services over HTTP if you are not using the XFireConfigurableServlet and you are not using any of the Spring remoting features. Here is an example on how to use it:

```

```

The servlet gets at the application context like so:

```
ApplicationContext appContext =

WebApplicationContextUtils.getRequiredWebApplicationContext(servletConfig.
getServletContext());
```

If you need to provide your own XFire instance another way, extend XFireServlet and override the getXFire method.