

# EN TITT PÅ XFIRE

Magnus Mickelsson

Jag gillar inte Webservices. Så, då var det sagt.

Eller, jag avskyr inte Webservices som koncept, men den komplexitet som de ofta medför i applikation, byggsript, testning med mera.

I min mening blir saker och ting lättare att förvalta ju enklare och mer intuitiva de är, utan att för den skull förlora i kvalitet och flexibilitet. Detta är ett av grundkoncepten i agilt tänkande.

Jag blev därför intresserad när jag fick höra talas om ramverket Xfire, som tar ett annat grepp än den traditionella, tunga J2EE-synen på Webservices. Ramverket är Open Source (gratis) och kommer från Codehaus, vilka också handhar ramverken Xstream och OpenEJB.

Xfire använder sig av annotations på vanliga, enkla Java-klasser – utan att kräva kodgenerering eller WSDL-filer. Anrop till Xfire-webservices tolkas av en Servlet som i sin tur applicerar anropet på den Java-böna som programmeraren skapat och annoterat. WSDL-definitionen fås genom att Xfire i runtime tolkar Java-bönan och genererar en WSDL som svar på en vanlig HTTP-request till samma Servlet.

Nyttan blir att vi kan skapa enkla Java-klasser för att hantera våra Webservices, vilkas logik kan testas med vanliga enhetstester. Inga stubbar, ingen WSDL – ramverket hanterar detta åt oss.

Som en ren bonus finns det tester gjorda som då (oktober 2005) visade på att Xfire var 2-6 gånger snabbare än Axis för motsvarande meddelanden, en skillnad som växte ju större meddelandena blev. Xfire funkar dessutom bra tillsammans med Spring, ett ramverk som blivit ett populärt alternativ till ”klassisk” J2EE.

Lovande alltså, men hur ser det ut i praktiken? Ett exempel kanske kan få saker och ting att klarna något.

```
package se.citerus.webapp.service;  
import javax.jws.WebService;
```

```
@WebService(  
    serviceName = "HelloService.ws",  
    endpointInterface = "se.citerus.webapp.service.HelloWs"  
)  
  
public class HelloWsImpl implements HelloWs {  
    public String world() {  
        return "Hello World!";  
    }  
}
```

Detta är en väldigt enkel webservice. Värt att notera är den annotation (som lyder under JSR-181, Web Services Metadata for the Java Platform) som lagts till definierar vad servicen ska heta och vilket interface som används för kopplingen.

I detta fall ser interfacet ut så här:

```
package se.citerus.webapp.service;  
import javax.jws.WebService;  
  
@WebService  
public interface HelloWs {  
    public String world();  
}
```

För att kunna köra detta exempel krävs vidare en web-container, t.ex. Tomcat, samt att konfigurationen för denna anpassas att registrera Xfire's servlet som ska tolka anrop till HelloService.ws till vår enkla javaböna.

I detta exempel använder jag Spring som sammanhållande container, och konfigurationen i Spring blir som följer:

```
<!DOCTYPE beans PUBLIC  
    "-//SPRING//DTD BEAN//EN"  
    "http://www.springframework.org/dtd/spring-beans.dtd">  
  
<beans>  
    <!-- Define bean to handle JSR 181 Web Service annotations -->  
    <bean id="webAnnotations"  
        class="org.codehaus.xfire.annotations.jsr181.Jsr181WebAnnotations"/>  
    <!-- Define handler that maps incoming requests to the correct Web Service -->
```

```
<bean id="handlerMapping"
      class="org.codehaus.xfire.spring.remoting.Jsrl181HandlerMapping">
  <property name="typeMappingRegistry">
    <ref bean="xfire.typeMappingRegistry"/>
  </property>
  <property name="xfire">
    <ref bean="xfire"/>
  </property>
  <property name="webAnnotations">
    <ref bean="webAnnotations"/>
  </property>
</bean>

<!-- Define simple URLHandler to pass all requests to the JSR181 XFire handler -->
->
<bean class="org.springframework.web.servlet.handler.SimpleUrlHandlerMapping">
  <property name="urlMap">
    <map>
      <entry key="/*">
        <ref bean="handlerMapping"/>
      </entry>
    </map>
  </property>
</bean>

<!-- Include the default XFire beans that the framework provides -->
<import resource="classpath:org/codehaus/xfire/spring/xfire.xml"/>

<!-- Annotated XFire Web Services below -->
<!--
  As class, put the implementation class.
  Mappings as to which bean receives which Webservice call is found
  in the Annotations of the respective bean.
-->
<bean id="annotatedHello" class="se.citerus.webapp.service.HelloWsImpl"/>
</beans>
```

Puh, det är en del konfiguration som krävs för en endaste HelloWorld, i sann J2EE-tradition. Men, det sköna är att för att lägga till en ytterligare webservice behövs bara en ny rad i formen av:

```
<bean id="annotatedHello" class="se.citerus.webapp.service.HelloWsImpl"/>
```

Inte så pjåkigt.

Vidare behöver vi registrera i vår Servlet-container (i web.xml) att anrop ska dirigeras till vår Xfire-servlet:

```
...
<context-param>
  <param-name>contextConfigLocation</param-name>
  <param-value>classpath:org/codehaus/xfire/spring/xfire.xml</param-value>
</context-param>
<listener>
  <listener-
class>org.springframework.web.context.ContextLoaderListener</listener-class>
</listener>

<!--
  Webservice dispatcher using XFire via Spring
  Refer to xfire-servlet.xml to see Bean definitions related to this Servlet.
-->
<servlet>
  <servlet-name>xfire</servlet-name>
  <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-
class>
  <load-on-startup>1</load-on-startup>
</servlet>

<!-- Map .ws requests to XFire Web Services -->
<servlet-mapping>
  <servlet-name>xfire</servlet-name>
  <url-pattern>*.ws</url-pattern>
</servlet-mapping>
...
```

Med detta är vi nöjda och klara. Paketera följande som en web-applikation och gå till den adress applikationen har, t.ex.

<http://127.0.0.1:8080/helloworld/xfire/HelloService.ws?WSDL> för att få se den WSDL som Xfire genererar vid åtkomst.

För att underlätta testandet inkluderar vi nedan en länk till en ZIP-fil med ett Ant-drivet projekt som kan användas som exempel.

Ant-scriptet skapar ett WAR-arkiv (under build/deliveries/) som kan deployas. Surfa till index.jsp under webapplikationen så finns där en länk till den Xfire-genererade WSDL som gäller för vår webservice. Själva webservice-anropet demonstreras via att exekvera Ant-scriptet med target "run-helloworld" - notera dock att detta kräver en deployad WAR i en körande server.

Koden är testad i Tomcat 5.5, men borde fungera i alla Java 5.0-kompatibla Servlet-containers.

Hoppas ni fått en inblick i en värld med lite enklare Webservices, i alla fall i min mening. Till dess vi ses nästa gång, Magnus. ■ ■ ■

#### LÄR DIG MER OM XFIRE

- <http://www.citerus.se/pnehm/3-2006/xfire.zip>



**Magnus Mickelsson är konsult på Citerus AB och arbetar som systemarkitekt och utvecklare. Han har ett brinnande intresse för hur man kan förbättra sin utvecklingsprocess, både ur ett verktygs- och metodperspektiv. Vill du diskutera Xfire med honom? Maila på [magnus punkt mickelsson at citerus punkt se](mailto:magnus.punkt@mickelsson.se).**