Spring WS ve İstemci Tarafı

İstemci Tarafında Spring WS

- Spring WS kullanarak istemci tarafında web servis çağrıları yapmak için **WebServiceTemplate** ana sınıfı mevcuttur
- WebServiceTemplate **Source** nesnelerini gönderen ve response mesajlarını **Source** veya **Result** olarak işleyen metotlara sahiptir
- Ayrıca XML – nesne **dönüşümü** de yapabilir

İstemci Tarafında Spring WS

- WebServiceTemplate mesajları arka tarafta **WebServiceMessageSender** kullanarak göndermektedir
- HTTP, JMS, SMTP **transport** implemantasyonları vardır
- Default olarak **HttpUrlConnectionMessageSender** kullanılır

İstemci Tarafında Spring WS

- Apache Commons HttpClient tabanlı **HttpComponentsMessageSender**'da konfigüre edilerek kullanılabilir
  - Authentication, Http connection pooling ihtiyacı olan durumlar için uygundur

WebServiceTemplate Konfigürasyonu

WebServiceTemplate çalışabilmesi için bir MessageFactory nesnesine ihtiyaç duyar. SaajSoapMessageFactory veya AxiomSoapMessageFactory implemantasyonlarından birisi kullanılabilir.

```
<bean id="messageFactory"
        class="org.springframework.ws.soap.saaj.SaajSoapMessageFactory"/>

<bean id="webServiceTemplate"
        class="org.springframework.ws.client.core.WebServiceTemplate">

    <constructor-arg ref="messageFactory"/>

    <property name="defaultUri"
            value="http://localhost:8080/petclinic/ws/hello"/>
</bean>
```

Default bir URI belirtilebilir. Ayrıca web service isteği gönderilirken de URI parametre olarak verilebilir

WebServiceTemplate ve MessageSender Konfigürasyonu

```
<bean id="webServiceTemplate"
class="org.springframework.ws.client.core.WebServiceTemplate">

    <constructor-arg ref="messageFactory"/>

    <property name="defaultUri" value="http://localhost:8080/petclinic/ws"/>

    <property name="messageSender">
        <bean
class="org.springframework.ws.transport.http.HttpComponentsMessageSender">
            <property name="credentials">
                <bean class="org.apache.http
                        .auth.UsernamePasswordCredentials">
                    <constructor-arg value="user1:secret"/>
                </bean>
            </property>
        </bean>
    </property>
</bean>
```

Custom bir MessageSender implemantasyonu property olarak enjekte edilebilir. Örneğin credentials bilgisinin HTTP request'inde gönderilebilmesi için HttpComponentsMessageSender kullanılabilir

WebServiceTemplate ve OXM Konfigürasyonu

```
<bean id="messageFactory"
class="org.springframework.ws.soap.saaj.SaajSoapMessageFactory"/>

<bean id="webServiceTemplate"
class="org.springframework.ws.client.core.WebServiceTemplate">
    <constructor-arg ref="messageFactory"/>
    <property name="defaultUri"
        value="http://localhost:8080/petclinic/ws/hello"/>

    <property name="marshaller" ref="jaxbMarshaller"/>
    <property name="unmarshaller" ref="jaxbMarshaller"/>
</bean>

<oxm:jaxb2-marshaller id="jaxbMarshaller"
        context-path="com.javaegitimleri.petclinic.ws.model"/>
```

context-path attribute'una XML binding yapılmış Java sınıflarını içeren Paketlerin isimleri ":" ile ayrılarak verilir

WebServiceTemplate Kullanımı

```
String message =
      "<helloWorldRequest xmlns=\"http://www.java-egitimleri.com/greeting\">" +
         "<name>Kenan</name>" +
         "<age>39</age>" +
      "</helloWorldRequest>";


Source requestPayload = new StringSource(message);

Result responseResult = new StreamResult(System.out);


webServiceTemplate.sendSourceAndReceiveToResult(
            requestPayload, responseResult);
```

veya

```
webServiceTemplate.sendSourceAndReceiveToResult(
            "http://localhost:8080/petclinic/ws/hello",
            requestPayload, responseResult);
```

WebServiceTemplate Kullanımı

```
HelloWorldRequest request = new HelloWorldRequest();
request.setName("Kenan");
request.setAge(39);

HelloWorldResponse response = (HelloWorldResponse)
        webServiceTemplate.marshalSendAndReceive(request);

System.out.println(response.getGreeting());
```

XML – nesne dönüşümünün yapılabilmesi için WebServiceTemplate nesnesine uygun **Marshaller** ve **Unmarshaller** bean'larının tanıtılmış olması gerekir.

WebService MessageCallback

Web servis mesajına erişmeyi, SOAP header vb değerleri set etmeyi sağlayan bir callback mekanizması mevcuttur

```
WebServiceMessageCallback requestCallback =
                new WebServiceMessageCallback() {

    @Override
    public void doWithMessage(WebServiceMessage message)
                        throws IOException, TransformerException {
        ((SoapMessage)message).setSoapAction("urn:helloWS");
    }
};

webServiceTemplate.sendSourceAndReceiveToResult(
                    requestPayload, requestCallback, responseResult);
```

İletişim

- Harezmi Bilişim Çözümleri
- Kurumsal Java Eğitimleri
- http://www.java-egitimleri.com
- info@java-egitimleri.com