JAVA Eğitimleri

Spring ile SOAP/WSDL Tabanlı Web Servisleri

Spring WS Nedir?

- Contract first yaklaşımı benimsemiş SOAP/WSDL tabanlı WS geliştirmeyi sağlayan bir framework'tür
- En temel özelliği, **XML doküman/mesaj bakış açısı** ile düşünmeyi ön plana çıkarmasıdır
- **Gelen-giden veri** en önemli şeydir
- **Java** kodu daha geri plandadır ve bir **implementasyon detayıdır**

Contract First ve Contract Last Yaklaşımları

- **Contract first**
  - Önce WS kontratının yani WSDL'in oluşturulup, ardından bu kontrata uygun servisin implement edilmesini savunur
  - WSDL » Java code
- **Contract last**
  - Önce servisin implement edilip, ardından bu servis implemantasyonundan kontratın yani WSDL'in üretilmesini savunur
  - Java code **»** WSDL

Contract Last Yaklaşımın Problemleri

- Java **kodu değiştiği anda** WSDL'de değişebilir
- Ancak WSDL **iki farklı sistem arasındaki kontrat**tır, kodun değişmesi WSDL'in değişmesine **neden olmamalı**dır
- Java kodu üzerinden akacak **verinin boyutu**nun ve **kapsamı**nın kontrol edilmesi zor olabilir
- Sunucuda çalışan kodun içerisindeki **tiplerin bire bir karşılıkları** istemci tarafında olmayabilir

Spring WS ve Contract First Yaklaşım

- Spring WS'in “contract first” yaklaşımı kendi içinde **iki temel kısımdan** oluşur
  - Data contract (XSD)
  - Service contract (WSDL)
- Spring WS için XSD ve WSDL **başlangıç noktası**dır

Data Contract

- **Data Contract** ile gelen-giden **XML mesajların yapısı** belirlenir
- **XSD** kullanılarak oluşturulur
- Data contract'a uygun **XML mesajları** Spring WS tarafından alınır veya dönülür
- Bu mesajlara “**request**” ve “**response**” adı verilir

Service Contract

- Service contract ile de WS **operasyonlarının yapısı** tanımlanır
- **WSDL** ile belirlenir, ancak sıfırdan WSDL yazmaya gerek yoktur
- “XSD + convention”lar ile WSDL **otomatik üretilebilir**
- Bu işleme **otomatik wsdl publish** adı verilir

Extensible Markup Language (XML)

- XML, **verinin anlamı**nın verinin kendisi ile birlikte taşınmasıdır
- Veri **transferi ve saklanma** da kullanılır
- Veriye özel XML **elemanları** ve **attribute**'lar tanımlanır
- XML sadece **veri hakkında bir bilgi**yi ifade eder
- Bunun dışında o veri ile ilgili **başka herhangi bir şey** yapmaz

XML

- Verinin oluşturulması, sistemler arası transferi veya veritabanına kaydedilmesi, verinin gösterimi **farklı farklı sistemler** tarafından ele alınan işlemlerdir
- Verinin XML formatında olması bunlarla ilgili **herhangi bir kabiliyet**in olması anlamına gelmez
- XML bu açıdan **verinin ne olduğu**na odaklanır
- **HTML** ise **verinin gösterimi**ne odaklanır

XML Örneği

```
<vets>

   <vet id="101">
      <firstName>Ali</firstName>
      <lastName>Zor</lastName>
      <graduationYear>2011</graduationYear>
   </vet>

   <vet id="102">
      <firstName>Veli</firstName>
      <lastName>Uysal</lastName>
      <graduationYear>2005</graduationYear>
   </vet>

</vets>
```

XML Namespace Nedir?

```
<table>
  <tr>
    <td>Elma</td>
    <td>Armut</td>
  </tr>
</table>
```

HTML sayfa içerisindeki table elemanıdır

```
<table>
   <name>Yemek Masasi</name>
   <width>80</width>
   <length>120</length>
</table>
```

Bir masa bilgisini ifade eden table elemanıdır

```
<data>
    <table>
        <tr>
            <td>Elma</td>
            <td>Armut</td>
        </tr>
    </table>
</data>
```

```
<data>
    <table>
        <name>Yemek Masasi</name>
        <width>80</width>
        <length>120</length>
    </table>
</data>
```

Problem:
İki farklı table yapısının aynı mesajda ele alınması karışıklıklara yol açacaktır
Hangisi HTML table, hangisi masayı ifade eden table net değildir

Çözüm:
XML namespace tanımlarını kullanmaktır

XML Namespace Nedir?

- XML mesajındaki eleman ve attribute'ları **benzersiz biçimde ayrıştıran** bir URI'dır
- Genellikle kök elemanda tanımlanır

```
<data
xmlns:html="http://www.w3.org/1999/xhtml"
xmlns:furniture="http://www.furnitureworld.com">
    <html:table>
        <html:tr>
            <html:td>Elma</html:td>
            <html:td>Armut</html:td>
        </html:tr>
    </html:table>
    <furniture:table>
        <furniture:name>Yemek Masasi</furniture:name>
        <furniture:width>80</furniture:width>
        <furniture:length>120</furniture:length>
    </furniture:table>
</data>
```

xmlns:prefix=”URI” ile tanımlanır. Prefix sayesinde namespace'i uzun biçimde elemanların başına yazmaya gerek kalmaz

Default XML Namespace

- XML içindeki namespace'lerden bir tanesi **default** olarak tanımlanabilir

Default bir namespace tanımı yapılabilir
default namespace'in elemanları
prefix olmadan da kullanılabilir

```
<data
xmlns="http://www.w3.org/1999/xhtml"
xmlns:furniture="http://www.furnitureworld.com">
    <table>
        <tr>
            <td>Elma</td>
            <td>Armut</td>
        </tr>
    </table>
    <furniture:table>
        <furniture:name>Yemek Masasi</furniture:name>
        <furniture:width>80</furniture:width>
        <furniture:length>120</furniture:length>
    </furniture:table>
</data>
```

XML Şema (XSD) Nedir?

- **XML dokümanın/mesajın yapısı**nı tanımlar
- Doküman içerisinde yer alacak **eleman** ve **attribute**'ların neler olabileceğini, bunlar arasındaki yapısal ilişkileri belirtir
- Eleman ve attribute'ların **veri tipleri**ni, hangi sabit değerleri alabileceklerini veya default değerlerin neler olabileceğini belirtir
- Bir elemanın **çocuk elemanları**nın neler olabileceğini, sayısını ve sırasını belirtir
- Bir elemanın **içeriğinin olup olamayacağı**nı, text değer içerip içeremeyeceğini belirtir

XML Şema Örneği

```
<schema ...>

    <complexType name="vet">
        <sequence>
            <element name="firstName" minOccurs="1" maxOccurs="1"
type="string"/>
            <element name="lastName" minOccurs="1" maxOccurs="1"
type="string"/>
            <element name="graduationYear" minOccurs="1" maxOccurs="1"
type="integer"/>
            <element name="specialty" minOccurs="0"
maxOccurs="unbounded" type="ws:specialty"/>
        </sequence>
    </complexType>

    <complexType name="specialty">
        <sequence>
            <element name="name" minOccurs="1" maxOccurs="1"
type="string"/>
        </sequence>
    </complexType>

</schema>
```

XML Şema Tanımının Yapısı

Bir XSD tanımının kök elemanı her zaman için <schema>'dır

```
<?xml version="1.0" encoding="UTF-8"?>
<schema
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
    targetNamespace="http://www.java-egitimleri.com/vets"
    xmlns="http://www.java-egitimleri.com/vets"
    elementFormDefault="qualified">
...
</schema>
```

Şema içerisinde kullanılacak eleman ve veri tiplerinin XMLSchema namespace ve ön eki tanımlanır.

Bu şema tarafından tanımlanacak XML elemanlarının ait olacakları namespace'i tanımlar

Target namespace ve ön eki tanımlanır

Burada tanımlanmış elemanların herhangi bir XML dokümanı içerisinde kullanılması durumunda namespace qualifikasyonuna sahip olmaları gerektiğini anlatır

Simple Element Örnekleri

```
<element name="firstName" type="string"/>
<element name="lastName" type="string"/>
<element name="graduationYear" type="integer"/>
<element name="color" type="string" default="red"/>
<element name="color" type="string" fixed="red"/>
```

Fixed veya default değerler de tanımlanabilir

Complex Element Örneği

```
<element name="vet">

    <complexType>

        <sequence>
            <element name="firstName" type="string" />
            <element name="lastName" type="string" />
            <element name="graduationYear" type="integer" />
        </sequence>
    </complexType>
</element>
```

<complexType> elemanı ile tanımlanır

Çocuk elemanların belirtilen sırada olması gerektiğini tanımlar

Çocuk elemanlar

Sequence dışında başka XSD indicator'leri de vardır:
all: çocuk elemanlar herhangi bir sırada olabilir, fakat Her biri sadece bir kere yer alabilir
choice: Çocuk elemanlardan sadece bir tanesi olabilir

Complex Element Örneği

Complex element tanımlarken diğer bir yöntemde type tanımını elemanın Dışında bir yerde yapmaktır. Buna global type adı verilir.

```
<element name="vet" type="tns:vetType"/>

<complexType name="vetType">
    <sequence>
        <element name="firstName" type="string" />
        <element name="lastName" type="string" />
        <element name="graduationYear" type="integer" />
        <element name="specialty"   type="tns:specialtyType"
                                    minOccurs="0" maxOccurs="unbounded"/>
    </sequence>
</complexType>

<complexType name="specialtyType">

    <simpleContent>

        <extension base="string"/>
    </simpleContent>
</complexType>
```

Elemanın kaç defa tekrarlayabileceğini belirtir

Complex element'in sadece text içerik barındırmasını sağlar

Attribute ve Mixed Content

```
<complexType name="vetType">

    <attribute name="id" type="string"/>

    <sequence>
        <element name="firstName" type="string" />
        <element name="lastName" type="string" />
        <element name="graduationYear" type="integer" />
    </sequence>
</complexType>
```

<attribute> elemanı ile attribute tanımlanır. İsmi, tipi, default değeri vs belirtilir

```
<xs:element name="letter">
    <xs:complexType mixed="true">
        <xs:sequence>
            <xs:element name="name" type="xs:string" />
            <xs:element name="orderid" type="xs:positiveInteger" />
            <xs:element name="shipdate" type="xs:date" />
        </xs:sequence>
    </xs:complexType>
</xs:element>
```

Element içerisinde hem çocuk elemanların Hem de text içeriğin birlikte olabileceğini belirtir

XML Dosyadan XSD'ye Referans

- XML içerisinde yapısını tanımlayan XSD'nin lokasyon bilgisi de belirtilebilir

xmlns:xsi genellikle XML ve XSD dosyalarının içerisinde kullanılan built-in attribute'lara ait namespace tanımıdır

```
<data
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:html="http://www.w3.org/1999/xhtml"
xmlns:furniture="http://www.furnitureworld.com"

xsi:schemaLocation="
http://www.w3.org/1999/xhtml http://www.w3.org/2002/08/xhtml/xhtml1-strict.xsd
http://www.furnitureworld.com furnitures.xsd">

...

</data>
```

XML dokümanının yapısını tanımlayan şema dosyalarının lokasyonu da built-in **schemaLocation** ile belirtilir. Böylece XML **parser** veya **editor** XML dokümanını XSD'ye göre **validate** etme imkanına kavuşur

Spring WS MessageDispatcherServlet

- **MessageDispatcherServlet** web servis çağrılarını HTTP üzerinden ele almayı sağlar
- **DispatcherServlet**'e benzer
- Asıl işi **MessageDispatcher**'a delege eder
- MessageDispatcher da XML mesajlarının **endpoint'lere dispatch** edilmesini sağlar

MessageDispatcherServlet Konfigürasyonu

```
<web-app>
    <servlet>
        <servlet-name>spring-ws</servlet-name>
        <servlet-class>
org.springframework.ws.transport.http.MessageDispatcherServlet
        </servlet-class>

         <init-param>
            <param-name>transformWsdlLocations</param-name>
            <param-value>true</param-value>
         </init-param>

        <load-on-startup>1</load-on-startup>
    </servlet>

    <servlet-mapping>
        <servlet-name>spring-ws</servlet-name>
        <url-pattern>/ws/*</url-pattern>
    </servlet-mapping>
</web-app>
```

MessageDispatcherServlet Konfigürasyonu

- Ayrıca **WEB-INF/spring-ws-servlet.xml** isimli Spring ApplicationContext dosyası oluşturulmalıdır

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:web-services="http://www.springframework.org/schema/web-services"
xmlns:context="http://www.springframework.org/schema/context"
xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans.xsd
http://www.springframework.org/schema/context
http://www.springframework.org/schema/context/spring-context-4.0.xsd
http://www.springframework.org/schema/web-services
http://www.springframework.org/schema/web-services/web-services.xsd">

<context:component-scan base-package="com.javaegitimleri.petclinic.ws" />

<web-services:annotation-driven />

</beans>
```

Spring WS namespace'inin **<web-services:annotation-driven/>** elemanı ile anotasyon tabanlı Spring WS konfigürasyonu devreye alınmış olur

Otomatik WSDL Publish

```
<?xml version="1.0" encoding="UTF-8"?>
<schema xmlns="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://www.example.org/greeting"
xmlns:tns="http://www.example.org/greeting" elementFormDefault="qualified">

<element name="helloWorldRequest">
    <complexType>
        <sequence>
            <element name="name" minOccurs="1" maxOccurs="1" type="string" />
            <element name="age" minOccurs="1" maxOccurs="1" type="int" />
        </sequence>
    </complexType>
</element>

<element name="helloWorldResponse">
    <complexType>
        <sequence>
            <element name="greeting" minOccurs="1" maxOccurs="1" type="string"/>
        </sequence>
    </complexType>
</element>

</schema>
```

Request soneki WS metot çağrısının input argümanlarını tanımlar

Response soneki ise WS metot çağrısının return tipini tanımlar

Request ve Response son eklerinden önceki kısım WS metot ismini oluşturur

Otomatik WSDL Publish

```
<beans ...>

    <web-services:dynamic-wsdl
        id="greeting"
        portTypeName="HelloWorld"

        locationUri="/ws/hello">

        <web-services:xsd location="classpath:/greeting.xsd"/>

    </web-services:dynamic-wsdl>

</beans>
```

http://localhost:8080/petclinic/ws/greeting.wsdl adresinden erişilebilir

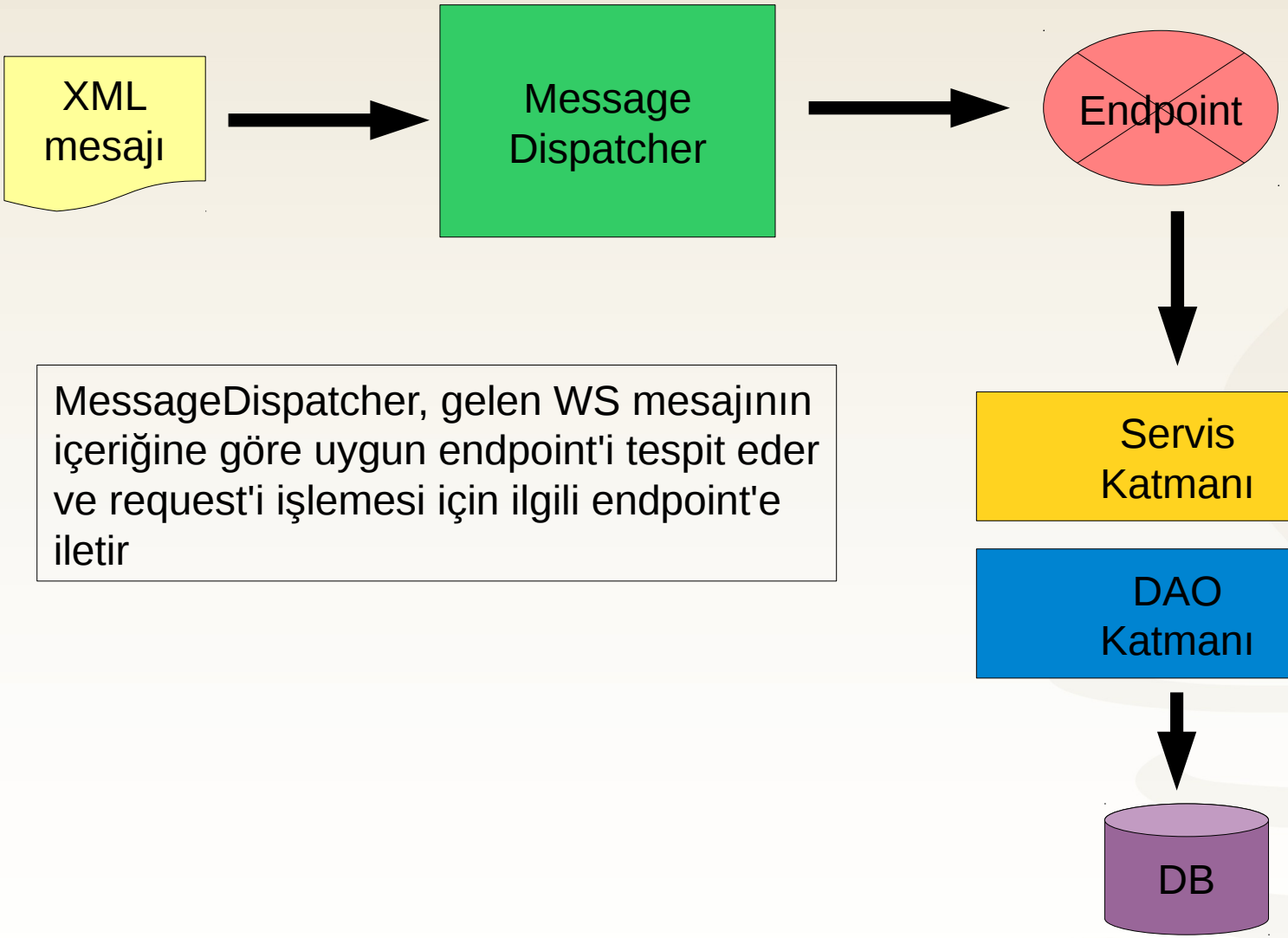
Servis ismi Spring WS tarafından "Service" son eki eklenerek elde edilir

WS erişim adresini belirler. Servlet transformWsdlLocations=true ise başına abosulte adres eklenir

XSD şema veya şemalar input olarak verilir ve inline olarak WSDL'e gömülür

<web-services:dynamic-wsdl> elemanının requestSuffix,responseSuffix, serviceSuffix gibi attribute'ları ile wsdl auto publish davranışını özelleştirmek mümkündür

Spring WS MessageDispatcher Mimarisi

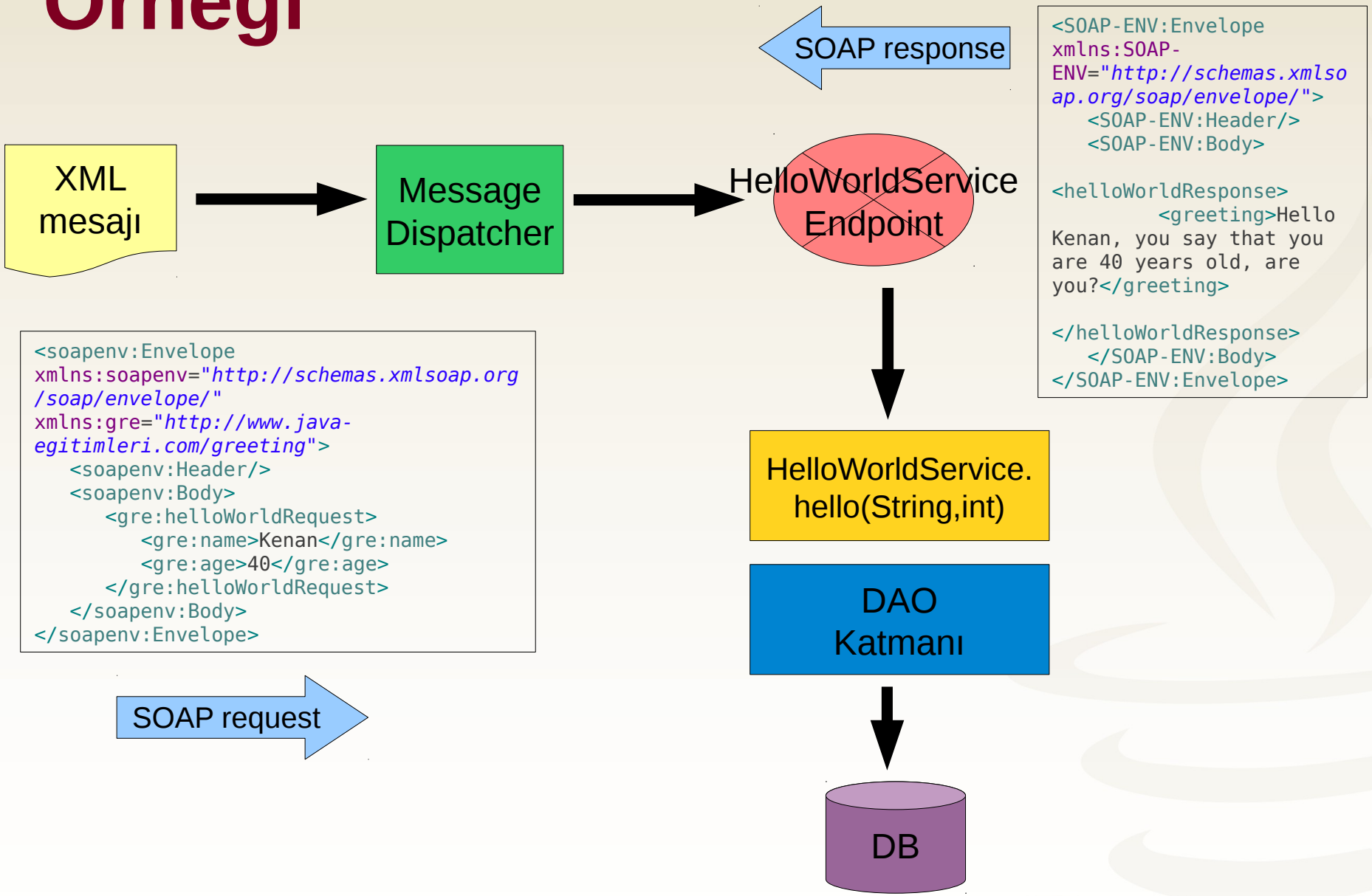


MessageDispatcher, gelen WS mesajının içeriğine göre uygun endpoint'i tespit eder ve request'i işlemesi için ilgili endpoint'e iletir

Spring WS Endpoint

- WS çağrısı ile gönderilen SOAP **mesajlarının işlendiği** yerdir
- SOAP mesajı işlenir ve servis katmanındaki **iş mantığı** çalıştırılır
- Servis katmanından **dönen sonuç** da SOAP cevabına dönüştürülerek istemci tarafına iletilir
- Metot parametreleri ve return değerinin dönüşüm işleminde genellikle **XML – nesne transformasyonu** söz konusudur

Spring WS Endpoint Örneği




```
<SOAP-ENV:Envelope
xmlns:SOAP-
ENV="http://schemas.xmlsoap.org/soap/envelope/">
   <SOAP-ENV:Header/>
   <SOAP-ENV:Body>

<helloWorldResponse>
         <greeting>Hello
Kenan, you say that you
are 40 years old, are
you?</greeting>

</helloWorldResponse>
   </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

```
<soapenv:Envelope
xmlns:soapenv="http://schemas.xmlsoap.org
/soap/envelope/"
xmlns:gre="http://www.java-
egitimleri.com/greeting">
   <soapenv:Header/>
   <soapenv:Body>
      <gre:helloWorldRequest>
         <gre:name>Kenan</gre:name>
         <gre:age>40</gre:age>
      </gre:helloWorldRequest>
   </soapenv:Body>
</soapenv:Envelope>
```

Spring WS Endpoint Örneği

@Component anotasyonundan türer, bean'in bir web servis endpoint olduğunu belirtir

Metodun bir web servis request'inin handler'ı olduğunu belirtir. Bir endpoint'de birden fazla handler olabilir

javax.xml.transform.Source sınıfıdır. SOAP mesaj response'unu oluşturmakta kullanılır

org.w3c.dom.Element sınıfıdır. XML mesajına Erişim sağlar

SOAP mesaj içeriğinin metot input arg veya return değeri olmasını sağlar

```
@Endpoint
public class HelloWorldServiceEndPoint {
    @Autowired
    private HelloWorldService helloWorldService;

    @PayloadRoot(
        localPart = "helloWorldRequest",
        namespace = "http://www.java-egitimleri.com/greeting")
    public @ResponsePayload Source
            hello(@RequestPayload Element request) {

        String name = request.getElementsByTagNameNS(
            "http://www.java-egitimleri.com/greeting","name")
            .item(0).getTextContent();
        String age = request.getElementsByTagNameNS(
            "http://www.java-egitimleri.com/greeting","age")
            .item(0).getTextContent();

        return new StringSource("<helloWorldResponse><greeting>"
            + helloWorldService.hello(name, Integer.parseInt(age))
            + "</greeting></helloWorldResponse>");
    }
}
```

Spring WS Endpoint Örneği

JAVA Eğitimleri

Namespace tanımı yapmayı sağlar. Metot, sınıf Veya paket düzeyinde tanımlanabilir

SOAP mesaj payload içerisinden XML eleman veya attribute değerlerini doğrudan extract etmeyi ve metoda parametre geçmeyi sağlar

```
@Endpoint
public class HelloWorldServiceEndPoint {

    @Autowired
    private HelloWorldService helloWorldService;

    @PayloadRoot(
        localPart = "helloWorldRequest",
        namespace = "http://www.java-egitimleri.com/greeting")
    @Namespace(prefix = "ns", uri = "http://www.java-egitimleri.com/greeting")
    public @ResponsePayload Source
                hello(@XPathParam("//ns:name") String name,
                      @XPathParam("//ns:age") double age) {
        return new StringSource("<helloWorldResponse><greeting>"
            + helloWorldService.hello(name, (int)age)
            + "</greeting></helloWorldResponse>");
    }
}
```

Metot input argümanlarında sadece Xpath tarafından desteklenen veri tipleri kullanılabilir: Double/double, Boolean/boolean, String, org.w3c.dom.Node ve org.w3c.dom.NodeList

Spring WS Endpoint Örneği

```
@Endpoint
public class HelloWorldServiceEndPoint {

    @Autowired
    private HelloWorldService helloWorldService;

    @PayloadRoot(
        localPart = "helloWorldRequest",
        namespace = "http://www.java-egitimleri.com/greeting")
    public @ResponsePayload HelloWorldResponse
                    hello(@RequestPayload HelloWorldRequest request) {
        String message =
                helloWorldService.hello(request.getName(),request.getAge());
        HelloWorldResponse response = new HelloWorldResponse();
        response.setGreeting(message);
        return response;
    }
}
```

XML şema kullanılarak JAXB transformer ile üretilen Java sınıflarıdır. Bu sınıflarda JAXB anotasyonları mevcuttur. HelloWorldRequest ve HelloWorldResponse sınıfları XmlRootElement anotasyonu ile işaretlenmiştir. JAXB dönüşümü IDE içerisinden veya build aracı ile tetiklenebilir.

İletişim

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