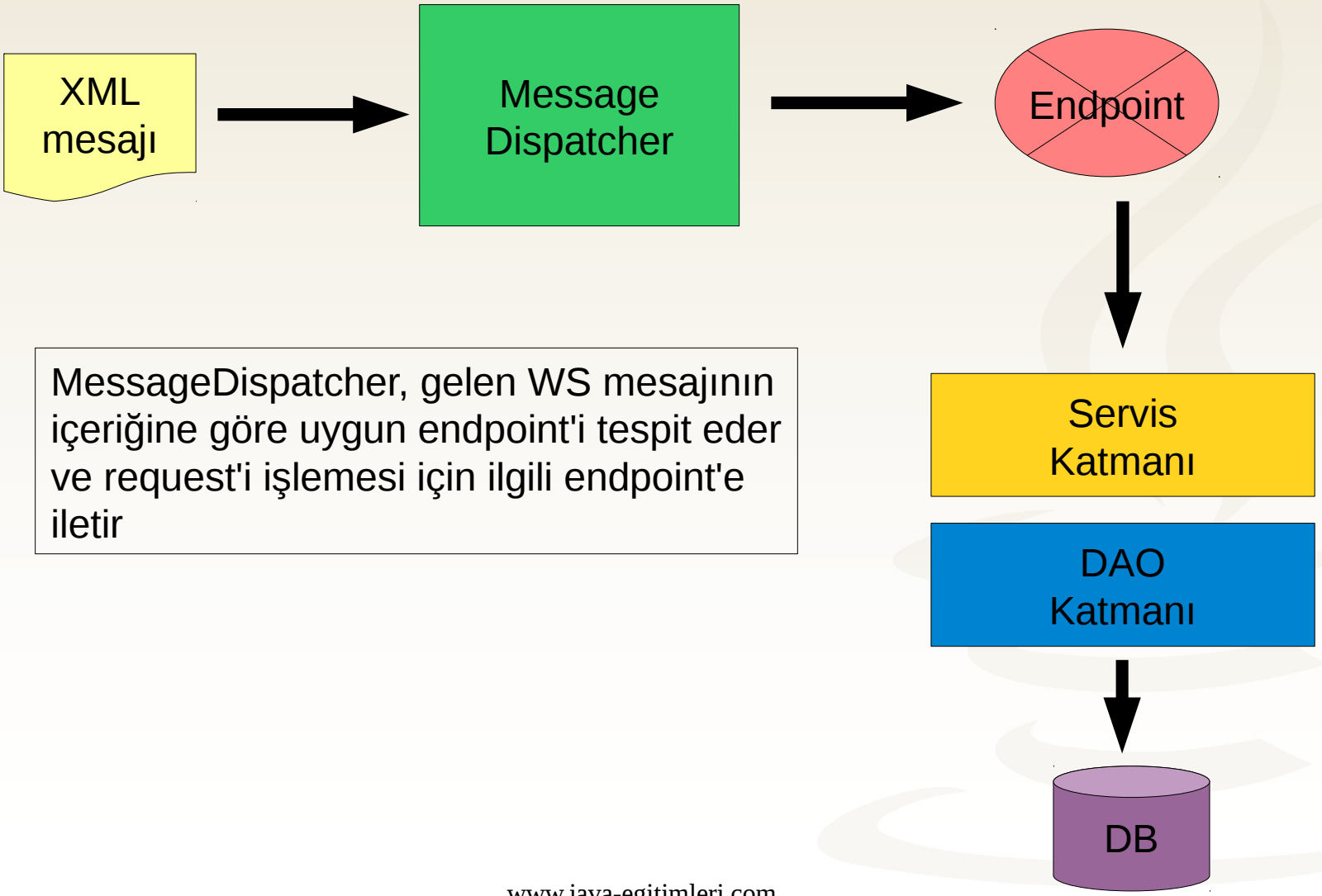
Spring WS ve Sunucu Tarafı

harezmi bilişim çözümleri

Spring WS MessageDispatcherServlet

- **MessageDispatcherServlet** Spring WS'in çalışma zamanındaki **giriş noktası**dır
- 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

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 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>
```

Spring WS 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

Java Tabanlı Spring WS Konfigürasyonu

- XML konfigürasyon dosyasına alternatif olarak Spring WS konfigürasyonu **Java konfigürasyon sınıfı**nda da yapılabilir

```
@Configuration
@ComponentScan(basePackages="com.javaegitimleri.petclinic.ws")
@EnableWs
public class SpringWsConfiguration{
    ...
}
```

Otomatik WSDL Publish

- XSD içerisinde WS **request ve response mesajlarının** yapısı, **veri tipleri** vs tanımlanmaktadır
- XSD içerisinde kullanılan **bir takım convention**'lar ile otomatik olarak **WSDL üretilmesi ve yayımlanması** da mümkündür

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

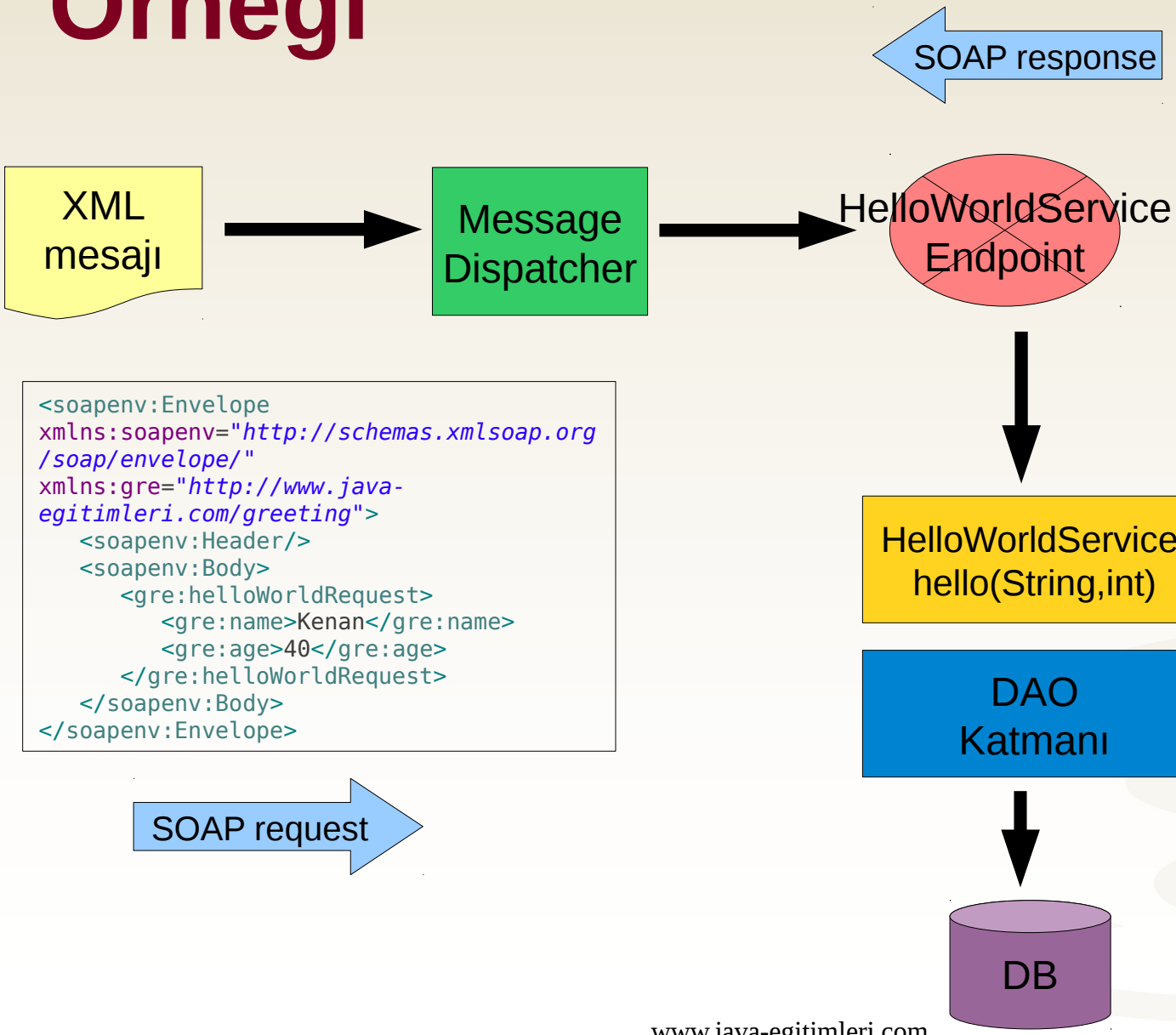
Servis ismi “Service” son eki eklenerek elde edilir

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 davranışı özelleştirmek mümkündür

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