มหาวิทยาลัยขอนแก่น
วิทยา จริยา ปัญญา KHON KAEN UNIVERSITY

JAX-WS

Asst. Prof. Dr. Kanda Runapongsa Saikaew
Department of Computer Engineering
Khon Kaen University
http://gear.kku.ac.th/~krunapon/xmlws

คณะวิศวกรรมศาสตร์ มหาวิทยาลัยขอนแก่น
FACULTY OF ENGINEERING KHON KAEN UNIVERSITY
1

Agenda

- What is JAX-WS?
- Quick overview of JAX-WS
  - Differences from JAX-RPC
- JAX-WS Programming Model
  - Layered programming model
  - Server side
  - Client side

คณะวิศวกรรมศาสตร์ มหาวิทยาลัยขอนแก่น
FACULTY OF ENGINEERING KHON KAEN UNIVERSITY
2

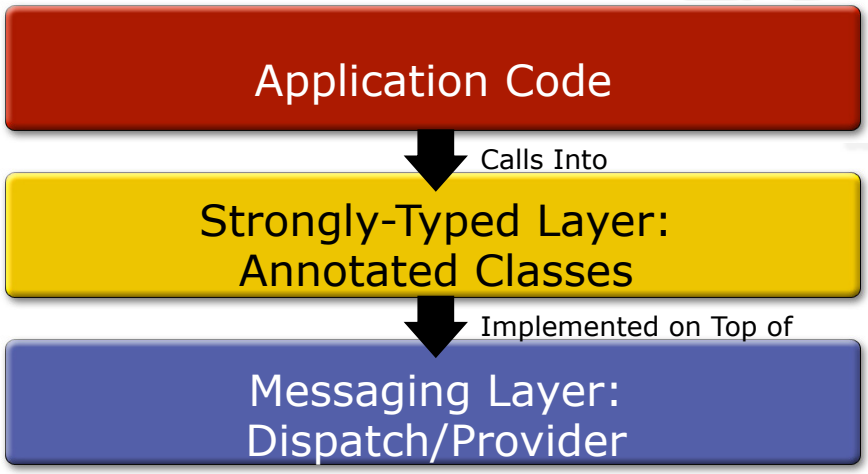
What is JAX-WS?

- JAX-WS stands for Java API for XML Web Services
- Technology for building web services and clients that communicate using XML
- JAX-WS allows developers to write
  - Message-oriented web services
  - RPC-oriented web services

Quick Overview of JAX-WS 2.0

- Simpler way to develop/deploy Web services
  - Plain Old Java Object (POJO) can be easily exposed as a Web service
  - No deployment descriptor is needed - use Annotation instead
  - Layered programming model
- Part of Java SE 6 and Java EE 5 platforms
- Integrated data binding via JAXB 2.0
- Protocol and transport independence

Layered Programming Model



Purpose of Each Layer

- Upper layer uses annotations extensively
  - Easy to use
  - Great toolability
  - Fewer generated classes
- Lower layer is more traditional
  - API-based
  - For advanced scenarios
- Most application will use the upper layer only

Two ways to create a Web Service

- Starting from a WSDL file (top-down approach)
  - Generate classes using *wsimport*
    - WS interface
    - WS implementation skeleton class
  - Add business logic to the WS implementation class
  - Build, deploy, and test
- Starting from a POJO (bottom-up approach)
  - Annotate POJO
  - Build and deploy
    - WSDL file generated automatically

Server-Side Programming Model:

Starting from POJO

1. Write a POJO implementing the service
2. Add @WebService annotation to it
3. Optionally, inject a WebServiceContext
4. Deploy the application
5. Point your clients at the WSDL
   - e.g. http://myserver/myapp/MyService?WSDL

คณะวิศวกรรมศาสตร์ มหาวิทยาลัยขอนแก่น
FACULTY OF ENGINEERING KHON KAEN UNIVERSITY

Example 1: Servlet-Based Endpoint

```
@WebService(targetNamespace =
"http://my.org/ns/")
public class Calculator {
    public int add(int a, int b) {
        return a+b;
    }
}
```

- @WebService annotation
  - All public methods become web service operations
- WSDL/Schema generated automatically
  - Default values are used

Example 2: EJB-Based Endpoint

```
@WebService(targetNamespace =
"http://my.org/ns/")
@Stateless
public class Calculator {
     @Resource
     WebServiceContext context;

    public int add(int a, int b) {
        return a+b;
    }
}
```

- It's a regular EJB 3.0 component, so it can use any EJB features
  - Transactions, security, interceptors...

Customizing through Annotations

```
@WebService(name="CreditRatingService",
            targetNamespace="http://
example.org")
public class CreditRating {


@WebMethod(operationName="getCreditScore"
)
    public Score getCredit(
        @WebParam(name="customer")
Customer c) {
            // ... implementation code ...
    }
}
```


Java SE Client-Side Programming

1. Point a tool (NetBeans or wsimport) at the WSDL for the service

wsimport http://example.org/calculator.wsdl

2. Generate annotated classes and interfaces
3. Call new on the service class
4. Get a proxy using a get<ServiceName>Port method
5. Invoke any remote operations

Example: Java SE-Based Client

```
CalculatorService svc = new
CalculatorService();
Calculator proxy =
svc.getCalculatorPort();
int answer = proxy.add(35, 7);
```

- No need to use factories
- The code is fully portable
- XML is completely hidden from programmer

Java EE Client-Side Programming

1. Point a tool (NetBeans or wsimport) at the WSDL for the service

```
wsimport http://example.org/
calculator.wsdl
```

2. Generate annotated classes and interfaces
3. Inject a @WebServiceReference of the appropriate type
4. Invoke any remote operations

Example: Java EE-Based Client

```
@Stateless
public class MyBean {

   // Resource injection

@WebServiceRef(CalculatorService
.class)
   Calculator proxy;

    public int mymethod() {
        return proxy.add(35, 7);
}
```


@WebServiceRef

- The WebServiceRef annotation is used to define a reference to a web service and (optionally) an injection target for it
- It can be used to inject both service and proxy instances
- These injected references are not thread safe
- If the references are accessed by multiple threads, usual synchronization techniques can be used to support multiple threads

Annotations Used in JAX-WS

- JSR 181: Web Services Metadata for the Java Platform
- JSR 222: Java Architecture for XML Binding (JAXB)
- JSR 224: Java API for XML Web Services (JAX-WS)
- JSR 250: Common Annotations for the Java Platform

@WebService

- Marks a Java class as implementing a Web Service, or a Java interface as defining a Web Service interface.
- Attributes
  - endpointInterface
  - name
  - portName
  - serviceName
  - targetNamespace
  - wsdlLocation

@WebService Optional Elements

- endpointInterface: The complete name of the service endpoint interface defining the service's abstract Web Service contract
- name: The name of the Web Service
- portName: The port name of the Web service
- targetNamespace: the targetNamespace is used for the namespace for the wsdl:portType (and associated XML elements)

@WebMethod

- Customizes a method that is exposed as a Web Service operation
- The method is not required to throw java.rmi.RemoteException
- Attributes
  - action:The action for this operation
  - Exclude: Marks a method to NOT be exposed as a web method
  - operationName: Name of the wsdl:operation matching this method

@WebMethod Optional Elements

- action: The action for this operation
- exclude: Marks a method to NOT be exposed as a web method
- operationName: Name of the wsdl:operation matching this method

@WebParam

- Customizes the mapping of an individual parameter to a Web Service message part and XML element.
- Attributes
  - header
  - mode
  - name
  - partName
  - targetNamespace

@WebParam Optional Elements

- header: If true, the parameter is pulled from a message header rather than the message body
- mode:The direction in which the parameter is flowing (One of IN, OUT, or INOUT)
- name: Name of the parameter
- partName: The name of the wsdl:part representing this parameter
- targetNamespace: The XML namespace for the parameter

@WebResult

- Customizes the mapping of the return value to a WSDL part and XML element.
- Attributes
  - header
  - name
  - partName
  - targetNamespace

@WebResult Optional Elements

- header: If true, the parameter is pulled from a message header rather than the message body
- name: Name return value
- partName: The name of the wsdl:part representing this return value
- targetNamespace: The XML namespace for the returned value

Example

```
@WebService(targetNamespace = "http://
duke.example.org", name="AddNumbers")
@SOAPBinding(style=SOAPBinding.Style.RPC,
use=SOAPBinding.Use.LITERAL)
public interface AddNumbersIF {
   @WebMethod(operationName="add",
action="urn:addNumbers")
   @WebResult(name="return")
   public int addNumbers(
      @WebParam(name="num1")int number1,
      @WebParam(name="num2")int number2) throws
AddNumbersException;
}
//Use  javax.jws.soap.SOAPBinding
```

คณะวิศวกรรมศาสตร์ มหาวิทยาลัยขอนแก่น
FACULTY OF ENGINEERING KHON KAEN UNIVERSITY

Protocol and Transport Independence

- Typical application code is protocol-agnostic
- Default binding in use is SOAP 1.1/ HTTP
- Server can specify a different binding, e.g.
  - @BindingType(SOAPBinding.SOAP12HTTP_BINDING)

Protocol and Transport Independence

- Client must use binding specified in WSDL
- Bindings are extensible, expect to see more of them
  - e.g. SOAP/Java Message Service(JMS) or XML/SMTP

Example

```
@WebService
@BindingType(value=javax.xml.ws.soap.SOAPBinding.SOAP12HTTP_BINDING)
public class AddNumbersImpl {

    // More code
}
```


References

- Sang Shin, “JAX-WS 2.x Basics”, http://www.javapassion.com/webservices/#JAX-WS_2.0
- http://download.oracle.com/javase/6/docs/api/