ΕΛΛΗΝΙΚΗ ΔΗΜΟΚΡΑΤΙΑ
ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΡΗΤΗΣ

Εισαγωγή στην Επιστήμη και Τεχνολογία των Υπηρεσιών

Ενότητα 14: JAX-WS

Χρήστος Νικολάου
Τμήμα Επιστήμης Υπολογιστών

Με τη συγχρηματοδότηση της Ελλάδας και της Ευρωπαϊκής Ένωσης

Άδειες Χρήσης

- Το παρόν εκπαιδευτικό υλικό υπόκειται στην άδεια χρήσης **Creative Commons** και ειδικότερα

Αναφορά – Μη εμπορική Χρήση – Όχι Παράγωγο Έργο v. 3.0

(Attribution – Non Commercial – Non-derivatives)

- Εξαιρείται από την ως άνω άδεια υλικό που περιλαμβάνεται στις διαφάνειες του μαθήματος, και υπόκειται σε άλλου τύπου άδεια χρήσης. Η άδεια χρήσης στην οποία υπόκειται το υλικό αυτό αναφέρεται ρητώς.

Χρηματοδότηση

- Το παρόν εκπαιδευτικό υλικό έχει αναπτυχθεί στα πλαίσια του εκπαιδευτικού έργου του διδάσκοντα.
- Το έργο «**Ανοικτά Ακαδημαϊκά Μαθήματα στο Πανεπιστήμιο Κρήτης**» έχει χρηματοδοτήσει μόνο τη αναδιαμόρφωση του εκπαιδευτικού υλικού.
- Το έργο υλοποιείται στο πλαίσιο του Επιχειρησιακού Προγράμματος «Εκπαίδευση και Δια Βίου Μάθηση» και συγχρηματοδοτείται από την Ευρωπαϊκή Ένωση (Ευρωπαϊκό Κοινωνικό Ταμείο) και από εθνικούς πόρους.

Ευρωπαϊκή Ένωση
Ευρωπαϊκό Κοινωνικό Ταμείο

ΕΠΙΧΕΙΡΗΣΙΑΚΟ ΠΡΟΓΡΑΜΜΑ
ΕΚΠΑΙΔΕΥΣΗ ΚΑΙ ΔΙΑ ΒΙΟΥ ΜΑΘΗΣΗ
επένδυση στην κοινωνία της γνώσης
ΥΠΟΥΡΓΕΙΟ ΠΑΙΔΕΙΑΣ & ΘΡΗΣΚΕΥΜΑΤΩΝ, ΠΟΛΙΤΙΣΜΟΥ & ΑΘΛΗΤΙΣΜΟΥ
ΕΙΔΙΚΗ ΥΠΗΡΕΣΙΑ ΔΙΑΧΕΙΡΙΣΗΣ

ΕΣΠΑ 2007-2013
πρόγραμμα για την ανάπτυξη
ΕΥΡΩΠΑΪΚΟ ΚΟΙΝΩΝΙΚΟ ΤΑΜΕΙΟ

Με τη συγχρηματοδότηση της Ελλάδας και της Ευρωπαϊκής Ένωσης

XML
JAX-WS
635.444

David Silberberg

Lecture 25

Overview of JAX-WS

- JAX-WS stands for Java API for XML Web Services
  - Builds clients and services that communicate using XML
  - Enables developers to write message oriented and RPC-oriented web services
- Java programming language API for creating web services
  - Part of the Java EE platform from Sun Microsystems
  - Like the other Java EE APIs, JAX-WS uses *annotations* (introduced in Java SE 5)
  - Simplifies the development and deployment of web services and clients
- The Reference Implementation of JAX-WS
  - Part of project GlassFish – an open source Java EE application server
  - Called JAX-WS RI (For Reference Implementation)
  - Production quality implementation

JAX-WS Hides the Complexity of SOAP & WSDL

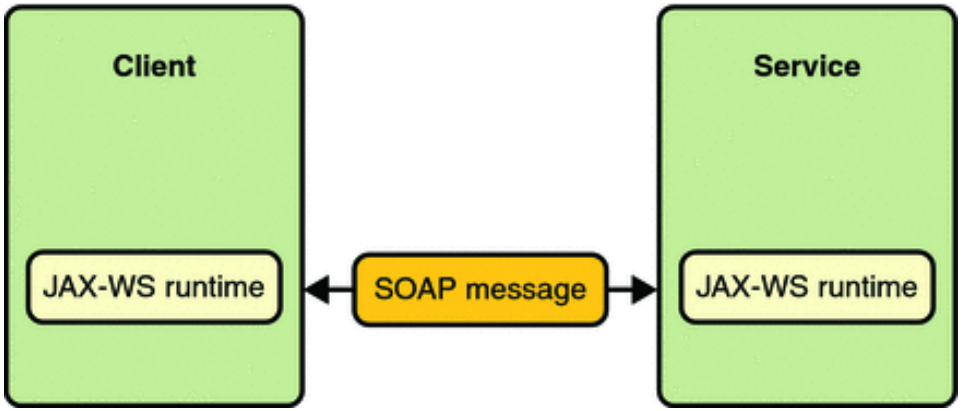
- SOAP is the XML-based protocol used
  - Envelope structure, encoding rules, and conventions for representing web service invocations and responses
  - Calls and responses are transmitted as SOAP messages via HTTP
- WSDL is the XML-based model for describing web services
- JAX-WS API hides the complexity of SOAP and WSDL
  - Server side development
    - Specify the web service operations by defining methods in a JAVA interface
    - Code the classes that implement those methods
  - Client side development
    - Create a proxy (a local object representing the service)
    - Invokes methods on the proxy.
  - Developer does not generate or parse SOAP messages
  - JAX-WS runtime system converts the API calls and responses to and from SOAP

Advantage of JAX-WS

- Platform independence provided by Java
- Java independence
  - JAX-WS clients can access web services not running on a Java platform
  - JAX-WS services can be called by non-Java clients
- Reason for Java independence of JAX-WS
  - Uses standards HTTP, SOAP, and the WSDL defined by the W3C
  - WSDL is an XML format for describing services as sets of endpoints operating on messages.

JAX-WS Architecture

- The architecture is a simple client/server architecture
- JAX-WS enables developers to write clients and services as java classes and methods
- JAX-WS technology manages communication between a web service and client.



Service Endpoint Interfaces

- Service
  - Java class that imports `javax.jws.WebService`
  - `@WebService` annotation defines the class as a web service endpoint
- Service endpoint implementation (SEI)
  - Java class
  - Defines the services methods that clients can call
- Service endpoint interface
  - Java Interface
  - Declares service interfaces that clients can call
  - Can be specified explicitly
    - Must specify endpointInterface elements to the `@WebService` annotation
  - Not required when building a JAX-WS endpoint
    - Implementation class implicitly defines an SEI
- The `wsgen` tool is applied to the endpoint implementation class to generate the web service artifacts that connect a web service client to the JAX-WS runtime

Basic Steps for Creating Client/Server

- Server
  - Code the implementation class
  - Compile the implementation class
  - Use `wsgen` to generate the artifacts required to deploy the service
  - Package the files into a WAR (Web ARchive) file
  - Deploy the WAR file
    - Web service artifacts used to communicate with clients are generated by the Application Server during deployment
- Client
  - Code the client class
  - Use `wsimport` to generate and compile the web service artifacts needed to connect to the service
  - Compile the client class
  - Run the client

JAX-WS Service Requirements

- Service class
  - Must import either the `javax.jws.WebService` or `javax.jws.WebServiceProvider`
  - May explicitly reference an SEI through the endpointInterface element of the `@WebService` annotation
    - If no endpointInterface is specified, an SEI is implicitly defined for the server class
  - Must not be declared `final`
  - Must not be abstract
  - Must have a default public constructor
  - Must not define the finalize method
  - May use the `javax.annotation.PostConstruct` or `javax.annotation.PreDestroy` annotations on its methods for life cycle event callbacks
- Service methods
  - Must be `public` and not be declared `static` or `final`
  - Exposed to clients
    - Must be annotated with `javax.jws.WebMethod`
    - Must have JAXB-compatible parameters and return types
- Construction and Destruction
  - The `@PostConstruct` method is called by the container before the implementing class begins responding to web service clients
  - The `@PreDestroy` method is called by the container before the endpoint is removed from operation

Coding the Service Endpoint Implementation Class

```
package helloservice.endpoint;// package name is helloservice

import javax.jws.WebService;  // required import

@WebService                   // required annotation
public class Hello {
   private String message = new String("Hello, ");

   public void Hello() {}     // default public constructor

   @WebMethod                 // required annotation
   public String sayHello(String name) {
        return message + name + ".";
   }
}
```

Building, Packaging, and Deploying the Service

- Build and package service
  - From a terminal window, go to the directory where the service is located
  - Type `ant` (calls the default target)
  - Automatically builds and packages the application into an WAR file (`helloservice.war`) located in the dist directory
- Deploy service
  - From a terminal window, go to the directory where the service is located
  - Start the Application Server (e.g., *Sun GlassFish Enterprise Server* – previously known as *Sun Java System Application Server*)
  - Type: `ant deploy`
- View WSDL
  - Type the URL `http://localhost:8080/`**`helloservice`**`/`**`hello`**`?WSDL` into a web browser
- All in one command – `ant all`
- Undeploy – `ant undeploy`

JAX-WS Client

Client performs the following steps:

- `javax.xml.ws.WebServiceRef` annotation declares a reference to a web service
- `@WebServiceRef` uses the `wsdlLocation` element to specify the URI of the deployed service's WSDL file

```
@WebServiceRef(wsdlLocation=
                          "http://localhost:8080/helloservice/hello?wsdl")
static HelloService service;
```

- Retrieves a proxy to the service, also known as a port, by invoking `getHelloPort` on the service.

```
Hello port = service.getHelloPort(); // implements SEI define by service
```

- Invokes the port's `sayHello` method with a `name` parameter

```
String response = port.sayHello(name);
```

JAX-WS Client Code

```
package simpleclient;

import javax.xml.ws.WebServiceRef;          // required import
import helloservice.endpoint.HelloService; // import the HelloService class
import helloservice.endpoint.Hello;         // import Hello class

public class HelloClient {

    @WebServiceRef(wsdlLocation="http://localhost:8080/
            helloservice/hello?wsdl")

    static HelloService service;            // HelloService variable

    public static void main(String[] args) {
        try {
            HelloClient client = new HelloClient();
            client.doTest(args);
        } catch(Exception e) {
            e.printStackTrace();
        }
    }
```

JAX-WS Client Code – continued

```
    public void doTest(String[] args) {
        try {
            System.out.println("Retrieving the port from the service: " +
            service);
            Hello port = service.getHelloPort();    // get a handle on the service

            System.out.println("Invoking the sayHello operation on the port.");

            String name;
            if (args.length > 0) {
                name = args[0];
            } else {
                name = "No Name";
            }

            String response = port.sayHello(name);  // call the service

            System.out.println(response);
        } catch(Exception e) {
            e.printStackTrace();
        }
    }
}
```

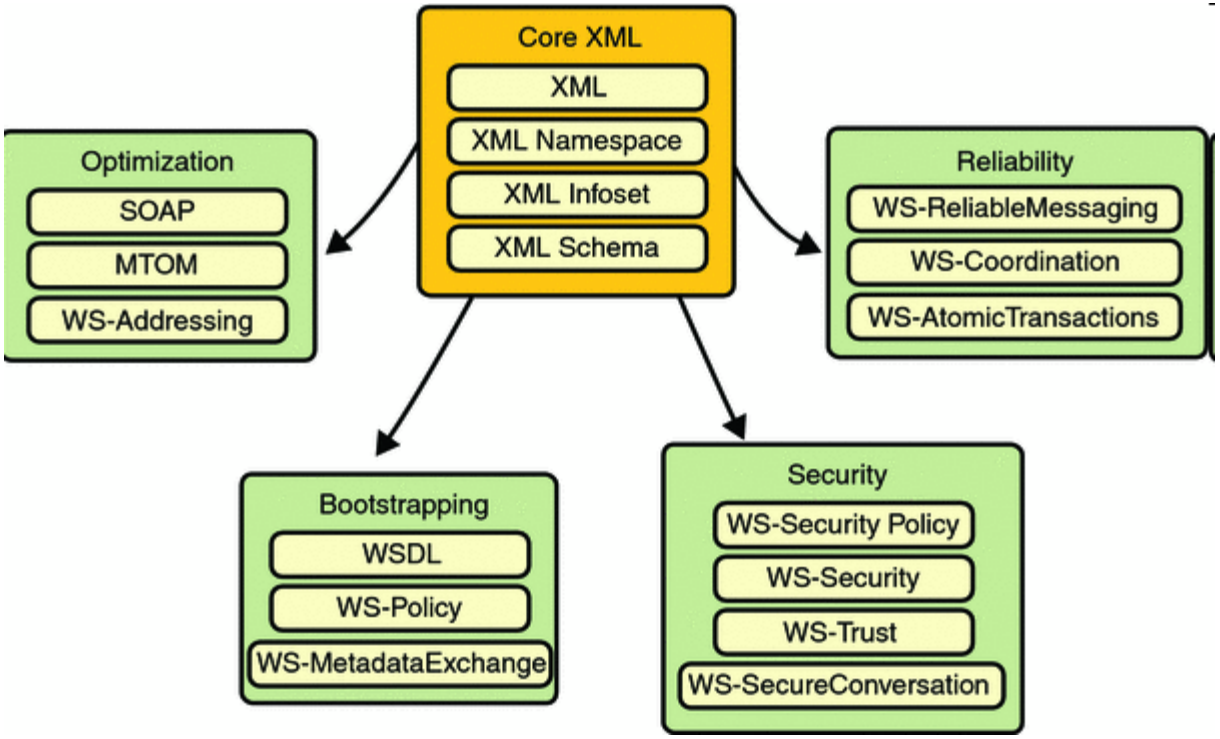
Building and Running the Client

- Make sure that `helloservice` is deployed
- From a terminal window, go to the directory containing the client code
- Type: `ant`
  - Calls the default target
  - Builds and packages the application into a JAR file – in this case `simpleclient.jar`
  - Places it into the dist directory
- To run the client, type: `ant run`
- JAX-WS uses JAXB to map Java programming language types to and from XML definitions
  - Insulates developers from the mappings
  - However, not every class in the Java language can be used as a method parameter or return type in JAX-WS (limited by JAXB)

Web Services – Bigger Picture

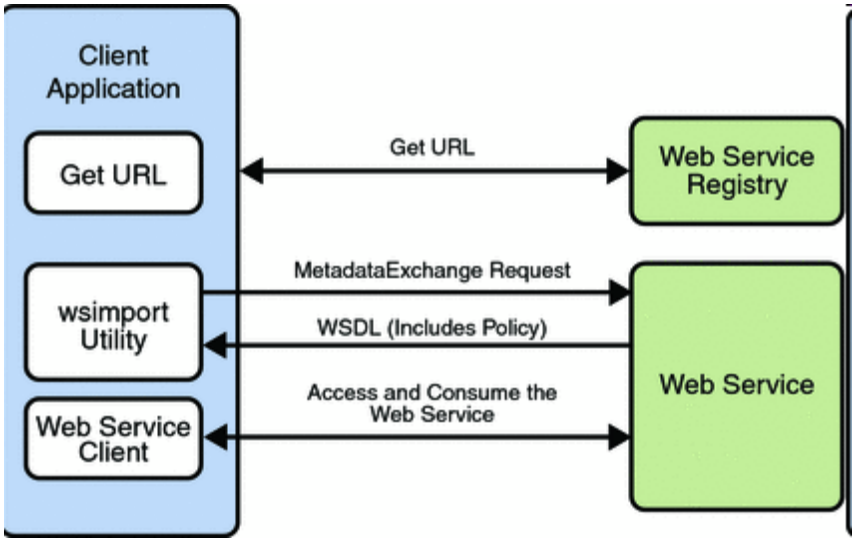
- Since 2005, Sun has worked closely with Microsoft to ensure interoperability of web services enterprise technologies
  - Security
  - Reliable messaging
  - Atomic transactions
- Metro, part of GlassFish, is a one-stop web services shop
  - Web Services portion is known as WSIT (Web Service Interoperability Technologies)
  - Implementation of a number of open web services specifications to support enterprise features
- WSIT augments JAX-WS capabilities

WSIT Architecture



Bootstrapping and Configuration

- Consists of
  - Using a URL to access a web service
  - Retrieving its WSDL file
  - Using the WSDL file to create a web service client that can access and consume a web service



Bootstrapping and Configuration Steps

- Client acquires the URL for a web service that it wants to access
  - Many ways to acquire the URL
  - Perhaps look up the URL in a Web Services registry
- Client uses the URL and the `wsimport` tool to send a `WS-MetadataExchange Request` to access the web service and retrieve the WSDL file
  - WSDL file contains a description of the web service endpoint
  - Includes WS-Policy assertions that describe the security, reliability, transactional, etc., capabilities and requirements of the service
- Client uses the WSDL file to create the web service client
- Client accesses the web service

Message Optimization Technology

- When large binary objects (such as documents, images, music files, etc.) are encoded into XML format for inclusion in SOAP messages, even larger files are produced
- When a web service processes and transmits these large files over the network, the performance of the web service application and the network are negatively affected
  - Performance may degrade to a point that it is no longer useful
  - Network gets bogged down with more traffic than the allotted bandwidth can handle
- Message Optimization encodes the binary objects to optimize
  - SOAP application processing time
  - Bandwidth required to transmit the SOAP message over the network
  - Recommended if binary encoded XML documents are larger than 1KB

Reliable Messaging

- Quality of Service (QoS) technology reliable web services
  - Reliability is measured by a system's ability to deliver messages from point A to point B
  - Purpose is to ensure the delivery of application messages to web service endpoints
- Ensures that messages in a given message sequence are delivered once and, optionally, in the correct order
  - Recovers when messages are lost or out of sequence
  - Lost messages are retransmitted
  - Out of sequence messages are retransmitted in order
- Reliable Messaging used when
  - Communication failures result in the network being unavailable or connections being dropped
  - Application messages are being lost in transit
  - Application messages are arriving at their destination out of order and ordered delivery is a requirement
- Uses more memory (especially if the ordered delivery option is enabled) since messages must be stored (even after they are sent) until receipt is acknowledged

Security Technology

- WS-Security provides interoperable message content integrity and confidentiality
  - Even when messages pass through intermediary nodes before reaching their destination endpoint
  - WS-Security is in addition to existing transport-level security such as TSL (or its predecessor SSL)
- Enhances security by implementing WS-Secure Conversation
  - Enables client and server to establish a shared security context when a multiple-message-exchange sequence is initiated
  - Subsequent messages use derived session keys that increase the overall security while reducing the security processing overhead for each message
- Two additional features to improve security
  - *Web Services Trust*: Clients use SOAP messages to request security tokens that establish trusted communications
  - *Web Services Security Policy*: Services use security assertions that represent preferences and requirements for web service endpoints

Τέλος Ενότητας

Ευρωπαϊκή Ένωση
Ευρωπαϊκό Κοινωνικό Ταμείο
ΕΠΙΧΕΙΡΗΣΙΑΚΟ ΠΡΟΓΡΑΜΜΑ
ΕΚΠΑΙΔΕΥΣΗ ΚΑΙ ΔΙΑ ΒΙΟΥ ΜΑΘΗΣΗ
επένδυση στην κοινωνία της γνώσης
ΥΠΟΥΡΓΕΙΟ ΠΑΙΔΕΙΑΣ & ΘΡΗΣΚΕΥΜΑΤΩΝ, ΠΟΛΙΤΙΣΜΟΥ & ΑΘΛΗΤΙΣΜΟΥ
ΕΙΔΙΚΗ ΥΠΗΡΕΣΙΑ ΔΙΑΧΕΙΡΙΣΗΣ
ΕΣΠΑ 2007-2013
πρόγραμμα για την ανάπτυξη
ΕΥΡΩΠΑΪΚΟ ΚΟΙΝΩΝΙΚΟ ΤΑΜΕΙΟ
Με τη συγχρηματοδότηση της Ελλάδας και της Ευρωπαϊκής Ένωσης