



BITS Pilani
Pilani | Dubai | Goa | Hyderabad

Non-REST (SOAP) Web Services

Srikanth Gunturu

Guest Faculty
BITS, WILP

In this segment

Non-REST (SOAP) Web Services

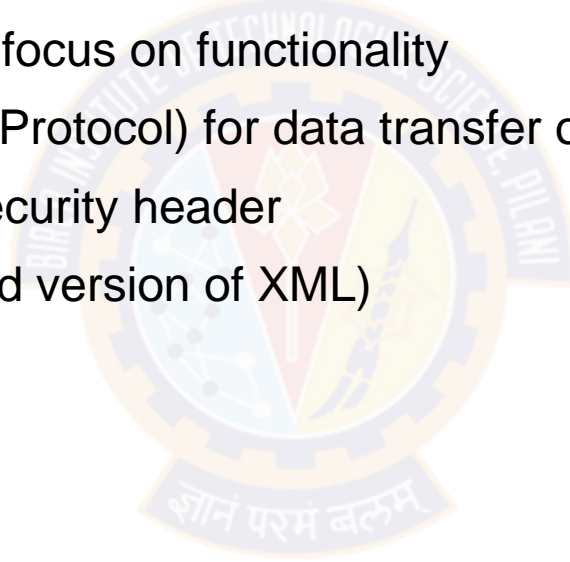
- Overview
- SOAP Messaging Protocol
- Web Services Description Language (WSDL)
- Java API for XML Web Services (JAX-WS)



Non-REST (SOAP) Web Services

Overview

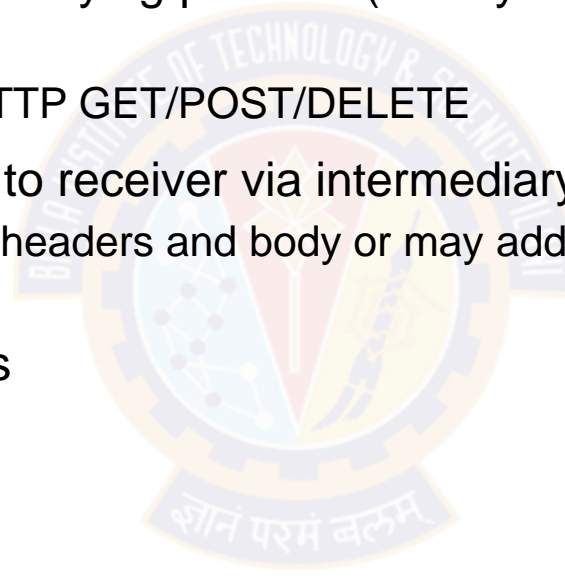
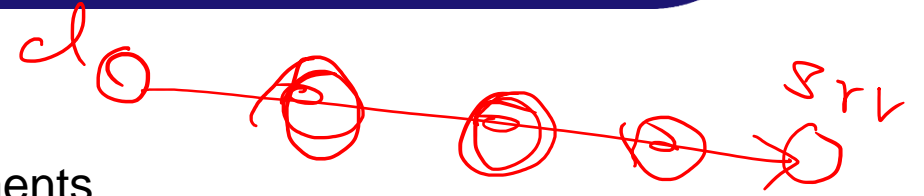
- First version of web services – released in late '90s
- Implemented as arbitrary web services grouped together based on business context
- Analogous to functional services – focus on functionality
- Use SOAP (Simple Object Access Protocol) for data transfer over HTTP/S
- Has embedded security via WS-Security header
- Supports XML (SOAP is a restricted version of XML)



Non-REST (SOAP) Web Services

SOAP Message Protocol - Overview

- Transfers structured information in the form of XML documents
- SOAP message is bound to the underlying protocol (mostly HTTP)
 - SOAP 1.1 – HTTP POST only
 - SOAP 1.2 (via web methods) – HTTP GET/POST/DELETE
- SOAP message goes from sender to receiver via intermediary nodes (SOAP nodes)
 - SOAP nodes may process SOAP headers and body or may add some additional headers for QoS / Security
- SOAP Message exchange patterns
 - Request-Reply
 - Response-only



Non-REST (SOAP) Web Services

SOAP Message

- SOAP message is enclosed in SOAP envelope
- Header is optional, Body is mandatory
- Envelope will specify namespace used
 - SOAP 1.1 – “xmlns:soap=http://schemas.xmlsoap.org/soap/envelope”
 - SOAP 1.2 – “xmlns:soap=http://www.w3.org/2003/05/soap-envelope”
- Encoding
 - SOAP Encoding – uses SOAP defined XML schema
 - Literal Encoding – uses XML schema (recommended)
- Message Styles
 - RPC – SOAP Encoded
 - RPC – Literal
 - Document - Literal
 - Document – Literal Wrapped

```
<soap: Envelope>  
  <soap:Header>  
    ...header stuff included here...  
  </soap:Header>  
  <soap:Body>  
    ...body stuff included here...  
  </soap:Body>  
</soap: Envelope>
```

```
<i xsi:type="xsd:int">7</i>  
<i>7</i>  
<hw: iElement>7</hw: iElement>  
<hw:i >7</hw:i>
```


Non-REST (SOAP) Web Services

SOAP Fault

- SOAP Fault – Contains error messages generated in processing

```
<soap: Fault>  
  <soap: Code>  
    <soap: Value>  
      ... put the appropriate code here. See Table 10.3 for codes  
    </soap: Value>  
    <soap: Subcode>  
      <soap: Value>  
        ...your application specific subcode. Could be something like "Timeout"  
        or "InvalidRequest"  
      </soap: Value>  
    </soap: Subcode>  
  </soap: Code>  
  <soap: Reason>  
    <soap: Text xml:lang='en'>... a text sentence (in this case in English) saying  
    more about the problem that occurred  
  </soap: Text>  
  </soap: Reason>  
</soap: Fault>
```

Predefined Codes

Version Mismatch

MustUnderstand

DataEncodingUnknown

Sender

Receiver

Subcode

Non-REST (SOAP) Web Services

SOAP Header

- SOAP Header – Attributes

- mustUnderstand

- Role

- Relay

```
<soap:Header>
```

```
<myschema:CurrentTransaction xmlns:myschema="http://myVeryOwnstuff/Transaction/"
```

```
  soap:role=predefined role URI (see Table 10.3) or custom, user defined Role
```

```
  soap:mustUnderstand="true"> ... insert current data for the current transaction here...
```

```
</myschema:CurrentTransaction>
```

```
</soap:Header>
```

Non-REST (SOAP) Web Services

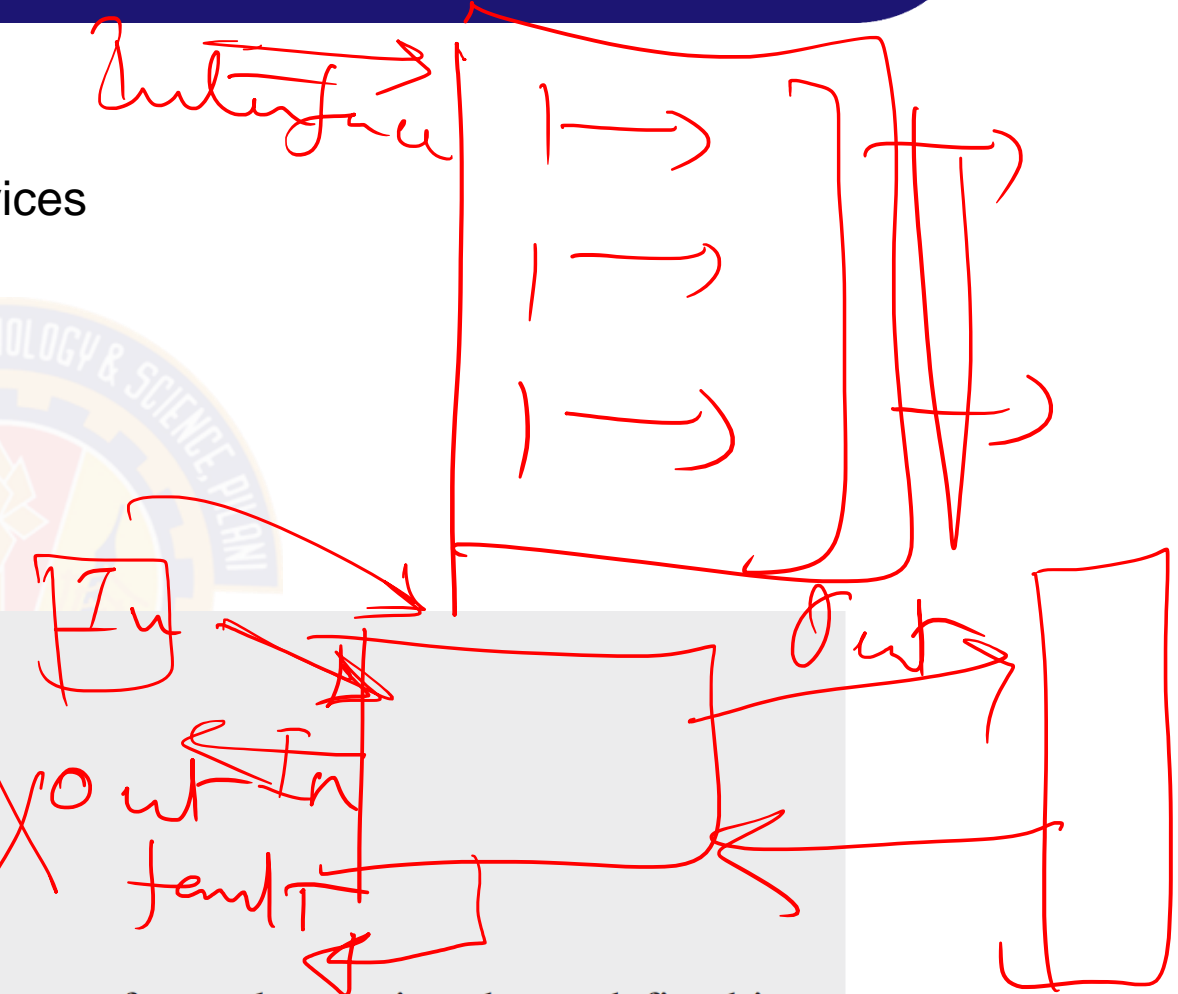
Web Services Description Language (WSDL)

- A language for describing interfaces to web services
- Specified in XML format
- Message Exchange Patterns:
 - In-only, Robust In-Only
 - In-Optional-Out, In-Out, **Out-In**
 - Out-only, Robust Out-Only

```
<wsdl:description
  xmlns:wsdl="http://www.w3.org/ns/wsdl"
  ...other namespaces...>
  <wsdl:types>
    ...define a schema here...
  </wsdl:types>
  <wsdl:interface>
    <wsdl:fault name="myfault" element=...refer to element in schema defined in
    types... />
    <wsdl:operation name="myOwnOperation"
      pattern="http://www.w3.org/ns/wsdl/in-out">
```

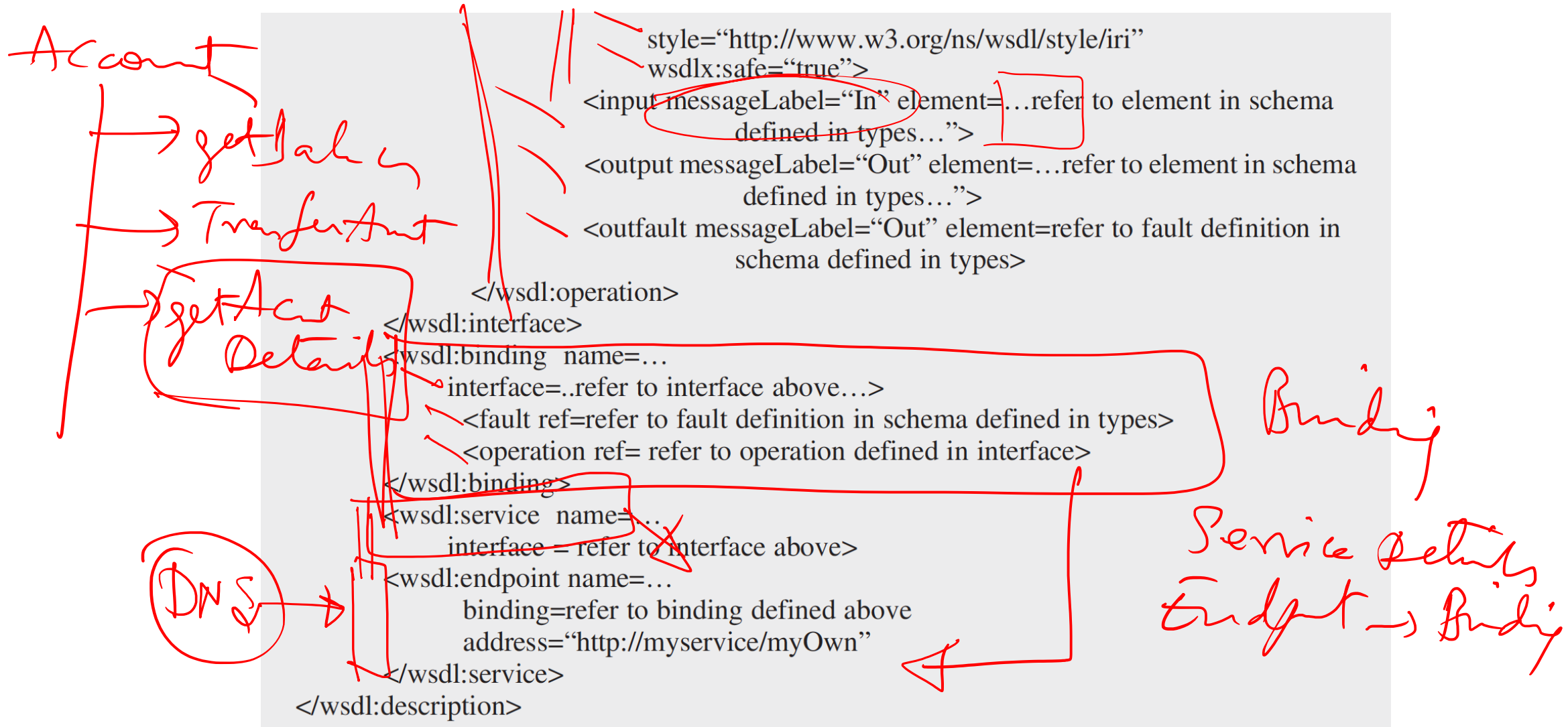
NS
Type

Interface
→ Fault
→ opes
→ In
→ Out



Non-REST (SOAP) Web Services

Web Services Description Language (WSDL) – contd.



Non-REST (SOAP) Web Services

JAVA API FOR XML WEB SERVICES (JAX-WS)

- Java API for creating SOAP web services, part of Java EE
- Server side code

Interface

```
package myHelloWorld;
import javax.jws.WebMethod;
import javax.jws.WebService;
import javax.jws.soap.SOAPBinding;
import javax.jws.soap.SOAPBinding.Style;

//Service Endpoint Interface
@WebService
@SOAPBinding(style = Style.RPC)
public interface HelloWorld {

    @WebMethod String HelloWorld(String name);
}
```

Impl

```
package myHelloWorld;
import javax.jws.WebService;

//Service Implementation
@WebService(endpointInterface = "myHelloWorld.HelloWorld")
public class HelloWorldImpl implements HelloWorld {

    @Override
    public String HelloWorld(String name) {
        System.out.println(name+" says hello");
        return "Hello World " + name;
    }
}
```

Non-REST (SOAP) Web Services

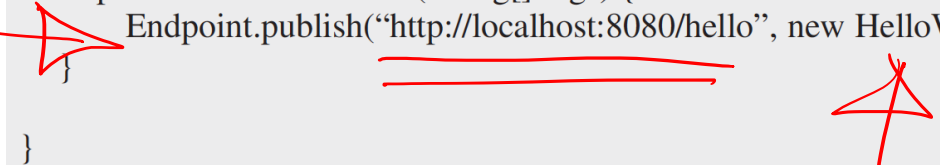
JAVA API FOR XML WEB SERVICES (JAX-WS)

- Service publisher

```
package myHelloWorld;
import javax.xml.ws.Endpoint;
import myHelloWorld.HelloWorldImpl;

//Endpoint publisher
public class HelloWorldPublisher{

    public static void main(String[] args) {
        Endpoint.publish("http://localhost:8080/hello", new HelloWorldImpl());
    }
}
```



Non-REST (SOAP) Web Services

JAVA API FOR XML WEB SERVICES (JAX-WS)

- Creating client with wsimport
- Client Stub

wsimport -keep http://localhost:8080/hello?wsdl -d .

```
import myhelloworld.HelloWorld;
import myhelloworld.HelloWorldImplService;
public class HelloWorldClient {
    public static void main(String[] args) {
        HelloWorldImplService myHelloWorld = new HelloWorldImplService();
        HelloWorld myinterface = myHelloWorld.getHelloWorldImplPort();
        //Note the format of the operation call "helloWorld".
        //This matches the format in the wsimport-generated HelloWorld.java file.
        String response = myinterface.helloWorld(args[0]);

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

Non-REST (SOAP) Web Services

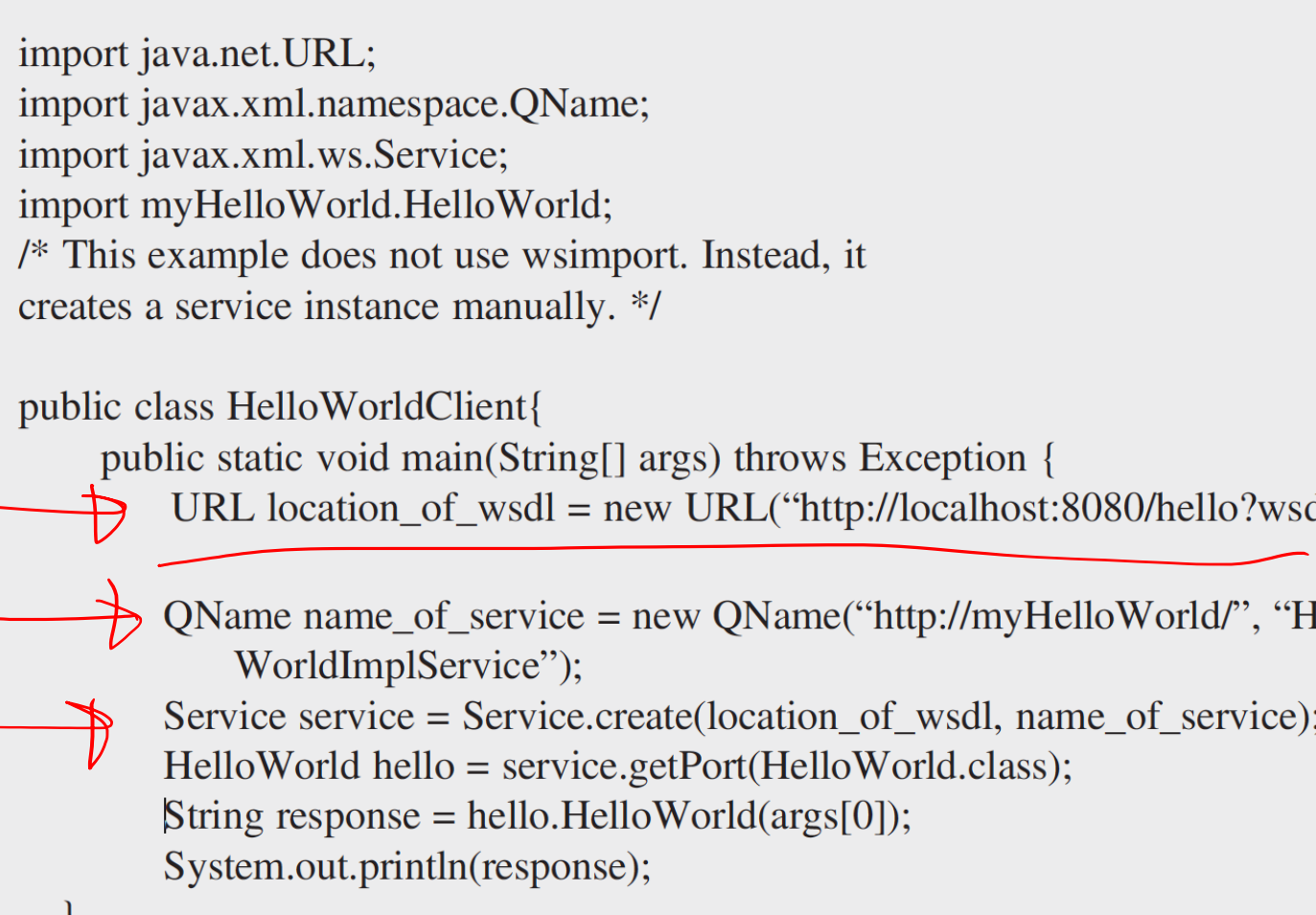
JAX-WS – Client code

- Dynamic Proxy Client

```
package myHelloWorld;

import java.net.URL;
import javax.xml.namespace.QName;
import javax.xml.ws.Service;
import myHelloWorld.HelloWorld;
/* This example does not use wsimport. Instead, it
creates a service instance manually. */

public class HelloWorldClient{
    public static void main(String[] args) throws Exception {
        URL location_of_wsdl = new URL("http://localhost:8080/hello?wsdl");
        QName name_of_service = new QName("http://myHelloWorld/", "Hello
WorldImplService");
        Service service = Service.create(location_of_wsdl, name_of_service);
        HelloWorld hello = service.getPort(HelloWorld.class);
        String response = hello.HelloWorld(args[0]);
        System.out.println(response);
    }
}
```





Thank You!

In our next session:
RESTful Web Services