class: inverse, center, middle

# JAVAX-WEB - Webszolgáltatások megvalósítása Java platformon

## Tematika

* SOAP alapú webszolgáltatások fejlesztése Java API for XML Web Services (JAX-WS) használatával
* REST alapú webszolgáltatások Java API for RESTful Web Services (JAX-RS) használatával

## Források

* Martin Kalin: Java Web Services Up and Running (O'Reilly)
* The Java EE 8 Tutorial (https://javaee.github.io/tutorial/)
* Bill Burke: RESTful Java with JAX-RS (O'Reilly)
* Mark D. Hansen: SOA Using Java Web Services (Prentice Hall)

## Opcionális források

* Steve Graham, Doug Davis, ...: Building Web Services with Java: Making Sense of XML, SOAP, WSDL, and UDDI, 2nd Edition (Developer’s Library)
* Antonio Goncalves: Beginning Java EE 7 (Apress)
* Arun Gupta: Java EE 7 Essentials (O'Reilly)

## Webszolgáltatás

* W3C definíció: hálózaton keresztüli gép-gép együttműködést támogató szoftverrendszer
* Platform független
* Szereplők
  + Szolgáltatást nyújtó
  + Szolgáltatást használni kívánó
  + Registry

## SOAP alapú webszolgáltatások

* Szabványokon alapuló
  + SOAP
  + WSDL
  + UDDI
* Kapcsolódó szabványok
  + HTTP
  + XML, és kapcsolódó szabványok: XSD, névterek

## SOAP

* W3C által karbantartott szabvány

## SOAP üzenet

## Példa SOAP kérés

<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">  
 <soap:Header/>  
 <soap:Body>  
 <listEmployees xmlns="http://training360.com/empapp"/>  
 </soap:Body>  
</soap:Envelope>

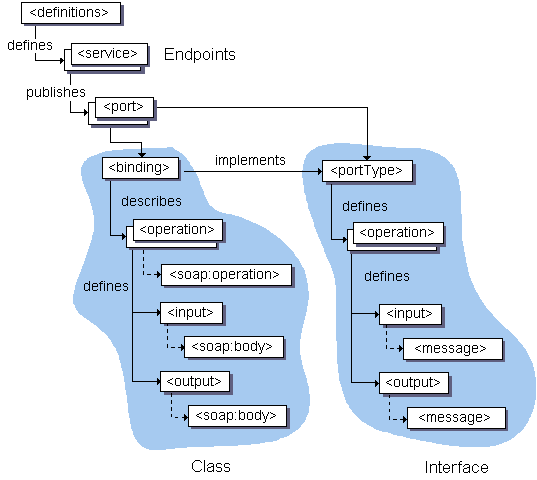
## Példa SOAP válasz

<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">  
 <soap:Body>  
 <listEmployeesResponse xmlns="http://training360.com/empapp">  
 <employee>  
 <id>1</id>  
 <name>John Doe</name>  
 </employee>  
 <employee>  
 <id>2</id>  
 <name>Jane Doe</name>  
 </employee>  
 </listEmployeesResponse>  
 </soap:Body>  
</soap:Envelope>

## WSDL

* Web Services Description Language
* WSDL dokumentum: több állományból

## WSDL felépítése



WSDL

## Példa WSDL állomány

<?xml version='1.0' encoding='UTF-8'?>  
<wsdl:definitions xmlns:xsd="http://www.w3.org/2001/XMLSchema"  
 xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"  
 xmlns:tns="http://training360.com/empapp"  
 xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"  
 xmlns:ns1="http://schemas.xmlsoap.org/soap/http"  
 name="EmployeeEndpointService"  
 targetNamespace="http://training360.com/empapp">  
  
 <!-- ... -->  
  
</wsdl:definitions>

## Példa WSDL állomány

<wsdl:types>  
 <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"  
 xmlns:tns="http://training360.com/empapp"  
 elementFormDefault="qualified"  
 targetNamespace="http://training360.com/empapp" version="1.0">  
  
 <xs:element name="listEmployees" type="tns:listEmployees"/>  
  
 <xs:element name="listEmployeesResponse" type="tns:listEmployeesResponse"/>  
  
 <xs:complexType name="listEmployees">  
 <xs:sequence/>  
 </xs:complexType>  
  
 <!-- ... -->  
  
 </xs:schema>  
</wsdl:types>

## Példa WSDL állomány

<wsdl:message name="listEmployees">  
 <wsdl:part element="tns:listEmployees" name="parameters">  
 </wsdl:part>  
</wsdl:message>  
<wsdl:message name="listEmployeesResponse">  
 <wsdl:part element="tns:listEmployeesResponse" name="parameters">  
 </wsdl:part>  
</wsdl:message>  
<wsdl:portType name="EmployeeEndpoint">  
 <wsdl:operation name="listEmployees">  
 <wsdl:input message="tns:listEmployees" name="listEmployees">  
 </wsdl:input>  
 <wsdl:output message="tns:listEmployeesResponse"   
 name="listEmployeesResponse">  
 </wsdl:output>  
 </wsdl:operation>  
</wsdl:portType>

## Példa WSDL állomány

<wsdl:binding name="EmployeeEndpointServiceSoapBinding" type="tns:EmployeeEndpoint">  
 <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>  
 <wsdl:operation name="listEmployees">  
 <soap:operation soapAction="" style="document"/>  
 <wsdl:input name="listEmployees">  
 <soap:body use="literal"/>  
 </wsdl:input>  
 <wsdl:output name="listEmployeesResponse">  
 <soap:body use="literal"/>  
 </wsdl:output>  
 </wsdl:operation>  
</wsdl:binding>  
<wsdl:service name="EmployeeEndpointService">  
 <wsdl:port binding="tns:EmployeeEndpointServiceSoapBinding"   
 name="EmployeeEndpointPort">  
 <soap:address   
 location="http://localhost:8080/empapp/EmployeeEndpoint"/>  
 </wsdl:port>  
</wsdl:service>

## Implementációs megközelítések

* Megközelítések
  + Bottom up: kód alapján
  + Top down: WSDL alapján
  + Meet in the middle
* Üzleti logika != interfész
  + Két szolgáltatás, két osztály
  + Megoldás: wrapper réteg

## JAX-WS

* Szabvány SOAP webszolgáltatások és kliensek fejlesztésére
* Java SE-ben is használható
* Java EE-ben
  + Web alkalmazásban: POJO (nem kell EJB konténer)
  + EJB: Stateless session bean (biztonság, tranzakció-kezelés)
* Épít a JAXB-re
* Támogatja a kód és WSDL alapú megközelítést is

## Implementációk

* JAX-WS Reference Implementation
  + https://javaee.github.io/metro-jax-ws/
* Apache Axis2
  + http://axis.apache.org/axis2/java/core/
* CXF
  + http://cxf.apache.org/

## Endpoint implementáció

@Stateless  
@WebService(targetNamespace = "http://training360.com/empapp")  
public class EmployeeEndpoint {  
  
 @WebMethod  
 @WebResult(name = "employee")  
 @ResponseWrapper(localName = "employees")  
 public List<Employee> listEmployees() {  
 // ...  
 }  
}

## SOAPUi

* SOAP és RESTful webszolgáltatások tesztelésére
* Ingyenes és kereskedelmi verzió
* Webszolgáltatás hívás
* Teszt csomagok, tesztesetek, asserttek
* Szimulálás és mockolás
* Terheléses tesztelés
* Futtatás parancssorból

## Szerver implementálása

* @WebService
  + serviceName, portName
  + targetNamespace
  + wsdlLocation
* @WebMethod - összes publikus publikálásra kerül, ezzel csak konfigurálni
  + operationName
  + action
  + exclude: nem lesz benne a WSDL-ben

## Paraméterek és visszatérési értékek

* @WebResult
  + name
  + targetNamespace
* @WebParam
  + name
  + targetNamespace
  + mode: IN / OUT / INOUT
* @RequestWrapper
* @ResponseWrapper

## Kliens implementálása

* Java API
* Generált osztályok

Parancssorból

wsimport -d <directory> -p <pkg> -s <directory> -target <version> -keep wsdl.wsdl

JAX-WS Maven Plugin, wsimport goal

## Kivételkezelés

* <Fault> tag
* Hibakezelés
  + RuntimeException leszármazott (unchecked)
  + Egyéb Exception leszármazott (checked), @WebFault annotáció
  + SOAPFaultException

SOAPFactory soapFactory = SOAPFactory.newInstance();  
SOAPFault soapFault = soapFactory.createFault(  
 "Cannot div parameter by 2.",  
 new QName("http://schemas.xmlsoap.org/soap/envelope/", "Client"));  
throw new SOAPFaultException(soapFault);

## WebServiceContext

* DI (@Resource annotációval)
* Hozzáférés pl. a HTTP stackhez, pl. header

@Resource  
private WebServiceContext webServiceContext;

Map<String, List<String>> headers = (Map<String, List<String>>) webServiceContext  
 .getMessageContext()  
 .get(MessageContext.HTTP\_REQUEST\_HEADERS);

## Request context

* A port BindingProvider példány

BindingProvider bindingProvider = (BindingProvider) echo;  
Map<String, List<String>> headers = new HashMap<>();  
headers.put("foo", Arrays.asList("bar"));  
bindingProvider.getRequestContext().put(MessageContext.HTTP\_REQUEST\_HEADERS, headers);

## Tesztelhetőség – publikálás

* Java 6 API – Endpoint.publish
* Embedded web container
* Standalone container

## Tesztelhetőség - hívás

* Kliens
* SAAJ/SOAPMessage
* HTTP/XML (Parser, XPath, XMLUnit)

## RESTful webszolgáltatások

* Roy Fielding: Architectural Styles and the Design of Network-based Software Architectures, 2000
* Representational state transfer
* Egyedileg címezhető erőforrások (resource)
* Erőforrások különböző formátumokban elérhetőek
* Erőforrásokon végzett standard (CRUD) műveletek
* Állapotmentes (stateless)

## RESTful webszolgáltatások

* Létező technológiák: URI, HTTP, XML, JSON
* AJAX világ segítette az elterjedését
* Web Application Description Language (WADL) – nem elterjedt

## HTTP újrafelhasználása

* URL
* HTTP metódusok használata (GET, PUT, POST, DELETE)
* Mime-type
* URL paraméterek
* Fejlécek
* Státuszkódok

## JAX-RS

* Java EE része, Java SE-ben is használható
* Root resource classes: POJO, EJB
* Resource methods: URI & operation to method mapping

## Deployment

@ApplicationPath("api")  
public class RestConfig extends Application  
{  
}

## JAX-RS annotációk

* @Path
* @Produces
* @Consumes
* @GET, @POST, @PUT, @DELETE, @HEAD és @OPTIONS, @HttpMethod
* @PathParam, @QueryParam, @MatrixParam, @CookieParam, @HeaderParam és @FormParam

## Resource

@Path("employees")  
public class EmployeeResource {  
  
 @Inject  
 private EmployeeServiceBean employeeServiceBean;  
  
 @GET  
 @Produces(MediaType.APPLICATION\_JSON)  
 public List<Employee> listEmployees() {  
 return employeeServiceBean.findEmployees();  
 }  
  
 @GET  
 @Path("{id}")  
 @Produces(MediaType.APPLICATION\_JSON)  
 public Employee findEmployeeById(@PathParam long id) {  
 return employeeServiceBean.findEmployeeById(id);  
 }  
  
}

## Resource

@Path("employees")  
public class EmployeeResource {  
  
 // ...  
  
 @POST  
 @Consumes(MediaType.APPLICATION\_JSON)  
 @Produces(MediaType.APPLICATION\_JSON)  
 public Response saveEmployee(Employee employeeData) {  
 return Response  
 .status(201)  
 .entity(employeeServiceBean.saveEmployee(employeeData.getName()))  
 .build();  
 }  
  
}

## Postman

* API fejlesztési életciklus támogatás
* RESTful webszolgáltatás hívások
* Egyszerre több környezet támogatása
* JavaScripttel szkriptelhető
* Automatikus tesztesetek
* Mock server
* Dokumentáció generálás
* Tesztesetek futtatása parancssorban

## Provider

* MessageBodyReader, Writer
* ContextResolver
* ExceptionMapper

## @Context annotáció

* A következőkbe injektálhatók
  + Resource
  + Provider
  + Application leszármazott

## @Context annotáció

* A következők injektálhatók
  + Application (önmagába nem)
  + UriInfo
  + HttpHeaders
  + Request
  + SecurityContext
  + Providers

## JAX-RS content handler

* MessageBodyReader, Writer
* StreamingOutput
* InputStream, Reader
* File
* byte[]
* String, char[] - @Produces charset
* MultivaluedMap<String, String>
* Source

## JAXB

* @XmlRootElement, @XmlType
* JAXBElement
* application/xml, text/xml, vagy application/\*+xml
* ContextResolver JAXBContext példányosításához, paraméterezéséhez

## JAX-RS 2.0 újdonságok

* Client API
* Async
* HATEOAS (Hypermedia)
* Annotations
* Validation
* Filters and Handlers
* Content negotiation

## Teljesítmény - XML

* Factory reuse
* Parser pool
* JAXBContext reuse
* Memóriaigény
* Validation
* Schema cache
* DTD – custom entity resolver
* Partial processing
* Selectiong right API: DOM, SAX, StaX, JAXB

## Teljesítmény - WS

* Message size
* Schema
* Endpoint implementation
* Handlers
* Fast infoset
* HTTP compression
* Reuse client proxy