Java technológiák – 12. előadás
Webszolgáltatások

ANTAL Margit

Sapientia - EMTE

2010

Az előadás célja

- Webszolgáltatás modell
- SOAP alapú webszolgáltatások
- REST típusú webszolgáltatások

Webszolgáltatás

A webszolgáltatás mechanizmust biztosít távoli eljáráshívásra, hasonlóan a következő technológiákhoz:

- CORBA – Common Object Request Broker Architecture (1990 – 2000)
- RMI – Remote Method Invocation (Java SE)
- RPC – Remote Procedure Call (pl. Network File System – RPC program)
- DCOM – Distributed Component Object Model – lassan átveszi a szerepét a .Net remoting és webszolgáltatások

Definíció

A webszolgáltatás egy PLATFORM, amely lehetővé teszi különböző hálózati alkalmazások együttműködését.

Webszolgáltatás tulajdonságok

- platform-, és nyelvfüggetlen ügyféltámogatás,
- elérhetőség: URI-vel azonosítható (logikai URI),
- egyszerű használat: a szolgáltatásleíró elégséges információt nyújt az igénybevételhez, nyelvfüggetlen módon. Pl. Amazon E-Commerce Service

```
http://webservices.amazon.com/AWSECommerceService/AWSECommerceService.wsdl
```

Webszolgáltatás-implementációk

Legelterjedtebbek

- REST - Representational State Transfer
- XML-RPC - Extensible Markup Language - Remote Procedure Call
- SOAP - Simple Object Access Protocol

Szolgáltatásorientált architektúra

SOA

Service Oriented Application

- SOA: egy architektúrális elv,
- webszolgáltatás: platform, amely a SOA elvet implementálja.

$\Rightarrow$ laza csatolású rendszerek

SOAP webszolgáltatás

SOAP elemek

- üzenet: a kliens és a szolgáltatás között az üzenetek XML formátumban haladnak,
- szolgáltatásleírás: XML formátumban tartalmazza a szolgáltatás műveleteinek leírását: paraméterek és típusaik, visszatérítési típus,
- szolgáltatás felderítés: $\approx$ Yellow Pages, egy regiszter,
- üzenetküldési mechanizmus: leggyakrabban HTTP protokollt használnak.

Szolgáltatások felderítése – UDDI

Egy regiszter (szolgáltatástár), amely lehetővéteszi:

- szolgáltatások (WSDL dokumentumok) tárolását,
- szolgáltatások keresését (a WSDL dokumentumban foglalt információk alapján).

SOA szereplők

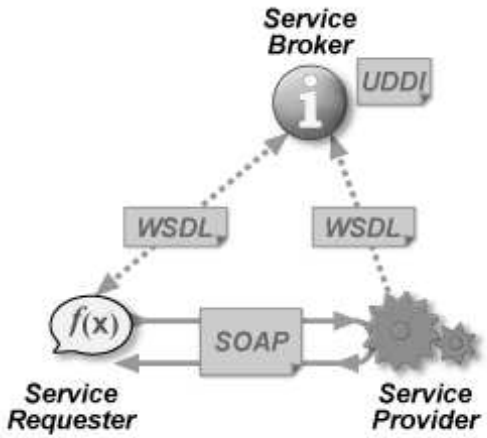


Figure:
http://upload.wikimedia.org/wikipedia/commons/4/4a/Webservices.png

Megjegyzés

Egyszerű rendszerek esetében a szolgáltatástárat (UDDI) nem érdemes megvalósítani. A szolgáltatás hívása a szolgáltatás leíró alapján megtehető.

JAX WS webszolgáltatás

```
package org.me;
import javax.jws.*;

@WebService()
public class MyService {

    @WebMethod(operationName = "add")
    public int add(@WebParam(name = "parameter1")
    int parameter1, @WebParam(name = "parameter2")
    int parameter2) {
        return parameter1 + parameter2;
    }
}
```

JAX WS webszolgáltatás-kliens

```
org.me.RandomizeWordsWSService service =
     new org.me.RandomizeWordsWSService();

org.me.RandomizeWordsWS port =
     service.getRandomizeWordsWSPort();

int parameter1 = 0;
int parameter2 = 0;
int result = port.add(parameter1, parameter2);
```

REST

Források

[1] Roger L. Costello, Building Web Services the REST Way http://www.xfront.com/sld001.htm
[2] Lars Vogel, RESTful Webservices with Java (Jersey / JAX-RS) - Tutorial http://www.vogella.de/articles/REST/

REST típusú webszolgáltatások

REST

Representational State Transfer

Tulajdonságok:

- architektúrális stílus, nem szabvány, de bizonyos szabványok használatára épül:
  - HTTP
  - URL
  - XML/HTML/GIF/JPEG/etc (Resource Representations)
  - text/xml, text/html, image/gif, image/jpeg, etc (MIME Types)
- a webszolgáltatást egy erőforrásnak tekinti, amelyet URL-el lehet azonosítani,

Representational State Transfer

Roy Fielding, PhD dissertation (2005)

”Representational State Transfer is intended to evoke an image of how a well-designed Web application behaves: a network of web pages (a virtual state-machine), where the user progresses through an application by selecting links (state transitions), resulting in the next page (representing the next state of the application) being transferred to the user and rendered for their use.”

Miért jó a REST?

- Az interneten minden erőforrásként kezelhető, és URL-el (fizikai vagy logikai) azonosítható.
- Az erőforrások elérését lehetővé teszi a HTTP protokoll.
- Segítségével skálázható, lazán csatolt rendszerek építhetők.

REST webszolgáltatás fejlesztése – 1. példa

- Szerveroldal: Java webalkalmazás: szervíz osztály (JAX-RS és Jersey osztálykönyvtárak)
- Kliensoldal:
  - Böngésző
  - Java konzol alkalmazás (JAX-RS és Jersey osztálykönyvtárak)

Szerveroldal: Szervíz osztály –1. példa

```
import javax.ws.rs.*;

//Sets the path to base URL + /hello
@Path("/hello")
public class Hello {

   // This method is called if TEXT_PLAIN is requested
   @GET
   @Produces(MediaType.TEXT_PLAIN)
   public String sayPlainTextHello() {
      return "Hello Jersey";
   }

   // This method is called if XML is requested
   @GET
   @Produces(MediaType.TEXT_XML)
   public String sayXMLHello() {
      return "<?xml version=\"1.0\"?>" + "<hello> Hello Jersey" + "</hello>";
   }

   // This method is called if HTML is requested
   @GET
   @Produces(MediaType.TEXT_HTML)
   public String sayHtmlHello() {
      return "<html> " + "<title>" + "Hello Jersey" + "</title>"
+ "<body><h1>" + "Hello Jersey" + "</body></h1>" + "</html> ";
   }
}
```

Szerveroldal: web.xml –1. példa

```
<servlet>
   <servlet-name>ServletAdaptor</servlet-name>
   <servlet-class>
      com.sun.jersey.spi.container.
                        servlet.ServletContainer
   </servlet-class>
   <load-on-startup>1</load-on-startup>
</servlet>
<servlet-mapping>
   <servlet-name>ServletAdaptor</servlet-name>
   <url-pattern>/resources/*</url-pattern>
</servlet-mapping>
```

Kliensoldal: böngésző

Böngésző

```
http://localhost:8084/App/resources/hello
```

Kérés

Melyik metódus fog meghívódni?

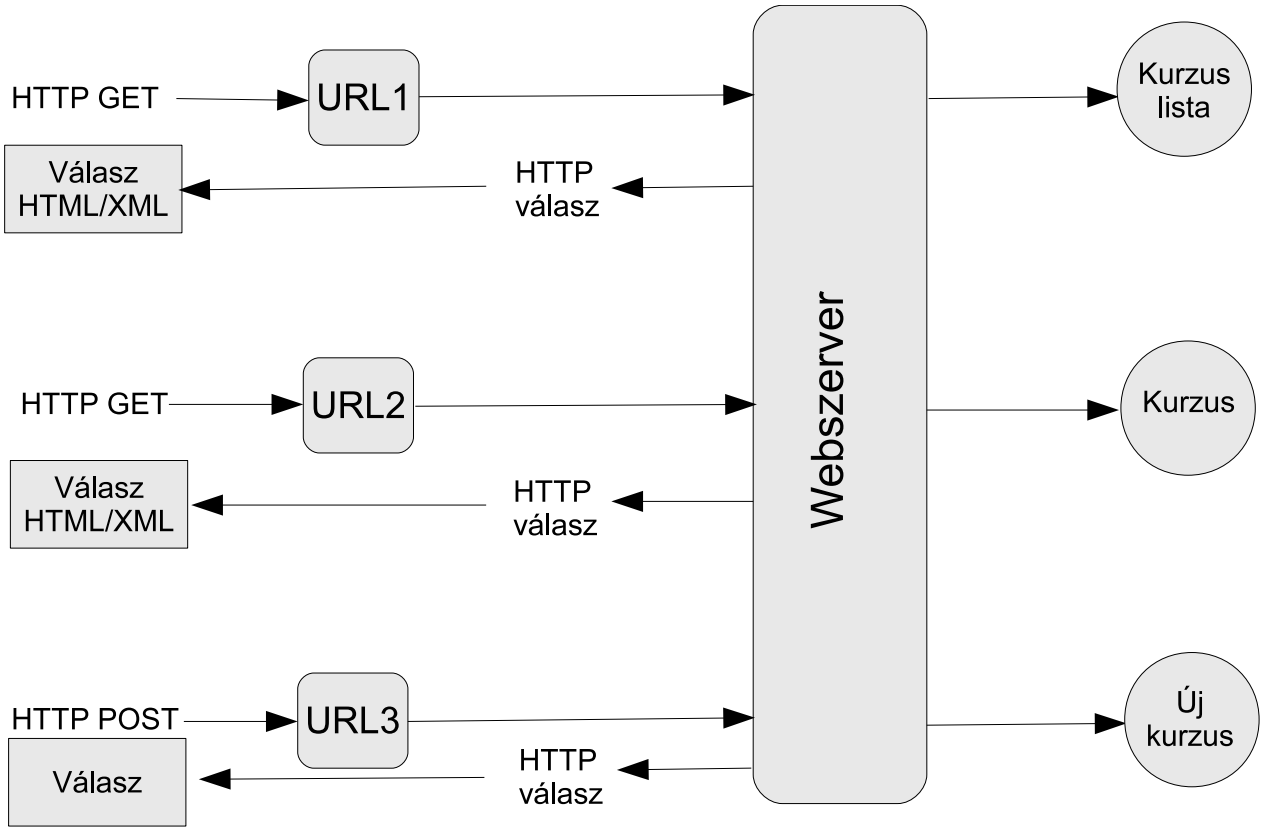
Kliensoldal: Java alkalmazás

```
import com.sun.jersey.api.client.*;
import com.sun.jersey.api.client.config.*;
import java.net.URI;
import javax.ws.rs.core.*;

public class Main {

    public static void main(String[] args) {
        ClientConfig config = new DefaultClientConfig();
        Client client = Client.create(config);
        WebResource service = client.resource(getBaseURI());
        System.out.println("HTTP Response: " +
                service.path("resources").path("hello").accept(
                MediaType.TEXT_PLAIN).get(ClientResponse.class).toString());
        //Get plain text
        System.out.println("PLAIN TEXT   : " +
                service.path("resources").path("hello").accept(
                MediaType.TEXT_PLAIN).get(String.class));
        //Get XML
        System.out.println("XML          : " +
                service.path("resources").path("hello").accept(
                MediaType.TEXT_XML).get(String.class));
        //Get HTML
        System.out.println("HTML         : " +
                service.path("resources").path("hello").accept(
                MediaType.TEXT_HTML).get(String.class));
    }

    private static URI getBaseURI() {
        return UriBuilder.fromUri(
                "http://localhost:8084/Webservice_REST").build();
    }
}
```

CRUD operations – HTTP metódusok

- CREATE – POST
- READ – GET
- UPDATE – PUT
- DELETE – DELETE

DISTEDU, the REST way – 2. példa



Kommunikáció a REST webszolgáltatással

Tulajdonképpen ez névvel ellátott erőforrások halmaza lesz.

Distedu REST webszolgáltatás

- HTTP GET – `http://ms.sapientia.ro:8080/distedu/list` – az összes tanfolyam adatai
- HTTP GET – `http://ms.sapientia.ro:8080/distedu/course/3` – a 3-as azonosítójú tanfolyam adatai
- HTTP POST – `http://ms.sapientia.ro:8080/distedu/add`, és a kérés törzse tartalmazná a tanfolyam adatokat

Kommunikáció a REST webszolgáltatással

A HTTP kérés tartalma – Content-type

- `text/html`
- `application/xml`
- `audio/x-wav`
- `image/gif`

REST webszolgáltatás implementáció

/distedu/add metódus

```
@POST
@Path("add")
@Consumes("application/xml")
public Response addCourse(){ ... }
```

/distedu/list metódus

```
@GET
@Path("list")
@Produces("application/xml")
public Response list( ){ ... }
```

Java REST webszolgáltatás vs. Java webalkalmazás

Hasonlóság

- Mindkettőt webkonténerbe kell telepíteni, *war* csomagformátumot használva.

Különbség

- a webszolgáltatás nem használja az állapotkezelési mechanizmusokat (pl. sütik)