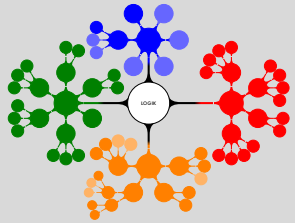


Tema 2

Servicios Web RESTful

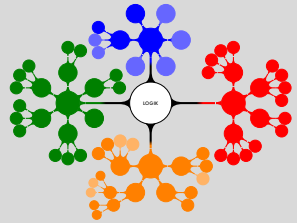
Interfaz *RESTful*

Asignatura *Desarrollo de Software* fecha 29 marzo 2016

Manuel I. Capel
Departamento de Lenguajes y Sistemas Informaticos
Universidad de Granada

Métodos HTTP y arquitecturas REST

- El estilo arquitectónico REST (“Representational State Transfer”)
- Objetivo: Diseño de componentes en un sistema hipermedia distribuido
  - Entidades REST: recursos: soporte, identificación y acceso
  - Interfaz de acceso
  - Negociación de contenidos
- Métodos HTTP
  - GET define un acceso de lectura al recurso
  - PUT crea un nuevo recurso o reemplaza el que haya (en URI indicado)
  - DELETE elimina recursos
  - POST envía datos a un URI específico y espera que el recurso allí los gestione



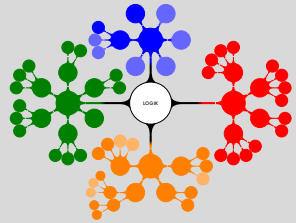
Métodos HTTP y arquitecturas REST – II

Servicios Web 'RESTful'

- Definición de la URL base para los servicios:
  static URI getBaseURI(){
  `UriBuilder.fromUri("http://localhost:8080/com.mio.jersey.first").build(); }`
- Programación de cliente configurado, que puede acceder al SW a través de la URI de base

```
com.sun.jersey.api.client.config.ClientConfig
    configuracion= new DefaultClientConfig();

com.sun.jersey.api.client.WebResource servicio=
  com.sun.jersey.api.client.Client.create(
      configuracion).resource(getBaseURI());
```

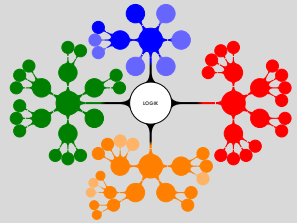
Métodos HTTP y arquitecturas REST – III

Establecimiento del intercambio de datos entre cliente y servicio

- Los tipos MIME soportados:

```
//Muestra el contenido del recurso como texto XML-
    plano
servicio.accept(MediaType.TEXT_XML).get(String.class)
    );
//Muestra el contenido del recurso como texto XML de
    aplicacion
servicio.accept(MediaType.APPLICATION_XML).get(String
    .class));
//Muestra el contenido del recurso como texto JSON
servicio.accept(MediaType.APPLICATION_JSON).get(
    String.class));
```

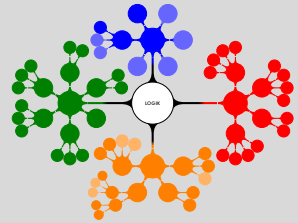
- Programar de la misma forma anterior el resto de las operaciones de REST que vayan a ser atendidas por el servicio.



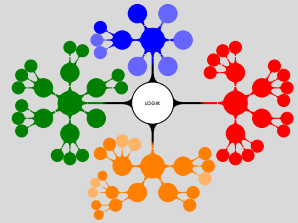
Arquitectura de Java para Ligadura XML

JAXB

- Arquitectura Java para XML
- Conversión de POJOs a la notación XML
- Java “mappings”: POJOs $<--->$ XML con APIs específicas
- Especificación estándar de JABX susceptible de varias implementaciones comerciales



Arquitectura de Java para Ligadura XML –II



Anotaciones de JAXB

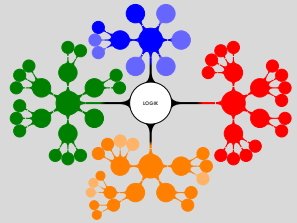
`@XmlRootElement(namespace = "espacionombre")`	raíz de un "árbol XML"
`@XmlType(propOrder = "campo2", "campo1",.. )`	orden escritura campos en el XML
`@XmlElement(name = "nuevoNombre")`	El elemento XML que será usado (*)

Nota(*): Sólo necesita ser utilizado si el nombre es diferente del nombre que tiene en JavaBeans

Java Specification Request (JAX-RS)

Java Specification Request (JSR) 311

- JAX-RS: Soporte REST para aplicaciones y servicios
- Utiliza anotaciones para seleccionar la “parte REST” de las aplicaciones de Java
- Implementación–referencia de JSR 311 se llama *Jersey*
  - Sirve para implementar servicios Web RESTful dentro de un contenedor de servlets (Tomcat, por ejmplo)
  - URL de base del servlet:
    `http://localhost:8080/nombre-a-mostrar/patron-url/camino-para-el-resto-de-la-clase`
- JAX-RS apoya la creación de XML y JSON a través de la Arquitectura Java para ligadura con XML (JAXB).

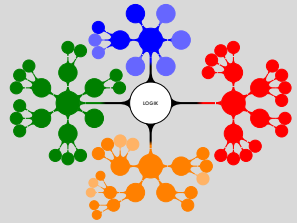


Anotaciones

Anotaciones más importantes de JAX–RS

- `@PATH(mi_camino)`
- `@POST`
- `@GET`
- `@PUT`
- `@DELETE`
- `@Produces(TiposMedia.TEXT_PLAIN[Más tipos])`

  $\in \{$“*application/xml*”, “*application/json*”, “*application_plain*”$\}$
- `@Consumes(tipo, [más tipos])`
- `@PathParam`

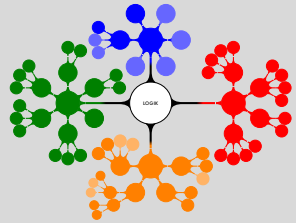


DAO

Definition

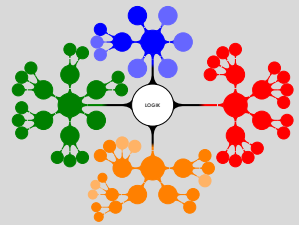
Un DAO o “objeto de acceso a datos” es un objeto que proporciona una interfaz abstracta a algún tipo de base de datos u otro mecanismo de persistencia.

- El DAO nos proporciona algunas operaciones sobre datos específicos sin que resulten visibles para las aplicaciones los detalles de la base de datos que actúa, a bajo nivel, como soporte de dichas operaciones.
- También se proporciona una correspondencia entre las llamadas a operaciones desde una aplicación a la *capa de persistencia* del servicio Web.



DAO Todo

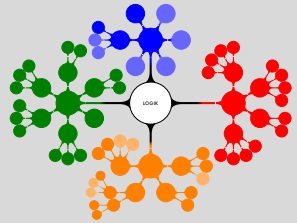
```
import java.util.HashMap;
import java.util.Map;
//importar el modelo del dominio de datos
public enum TodoDao {
  INSTANCE;//para implementar el patron singleton.
  private Map<String, Todo> proveedorContenidos = new
      HashMap<String, Todo>();
  private TodoDao() {
    Todo todo = new Todo("1", "Aprender REST");
    todo.setDescripcion("Leer http://lsi.ugr.es/ist/
        Documentos/practica2/supuesto_web_app.htm");
    proveedorContenidos.put("1", todo);
    todo = new Todo("2", "Aprender algo sobre REST");
    todo.setDescripcion("Leer todo el material de http://
        lsi.ugr.es/ist");
    proveedorContenidos.put("2", todo);  }
  public Map<String, Todo> getModelo(){
    return proveedorContenidos;  }
}
```

Dominio de datos

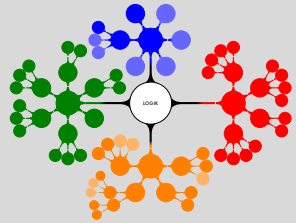
```
@XmlRootElement
public class Todo{
  private String id;
  private String resumen;
  private String descripcion;

  public Todo(){
  }
  public Todo (String id, String resumen){
    this.id = id;
    this.resumen = resumen;
  }
  public String getId() {
    return id;
  }
  public void setId(String id) {
    this.id = id;
  }
 ...
}
```

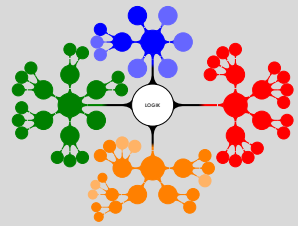
Recurso

```
import javax.ws.rs.Consumes;
import javax.ws.rs.DELETE;
import javax.ws.rs.GET;
import javax.ws.rs.PUT;
import javax.ws.rs.Produces;
import javax.ws.rs.core.Context;
import javax.ws.rs.core.MediaType;
import javax.ws.rs.core.Request;
import javax.ws.rs.core.Response;
import javax.ws.rs.core.UriInfo;
import javax.xml.bind.JAXBElement;
```

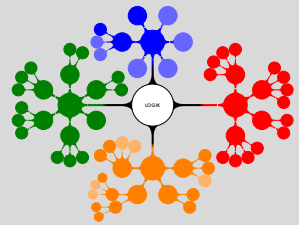
Recurso II

```
public class TodoResource {
  @Context
  UriInfo uriInfo;
  @Context
  Request peticion;   String id;
  public TodoResource(UriInfo uriInfo, Request peticion,
      String id) {
    this.uriInfo = uriInfo;
    this.request = peticion;
    this.id = id;  }
  @GET //Integracion de aplicaciones
  @Produces({MediaType.APPLICATION_XML, MediaType.
      APPLICATION_JSON})
  public Todo getTodo() {
    Todo todo = TodoDao.INSTANCE.getModelo().get(id);
    if(todo==null)
      throw new RuntimeException("Get: Todo con " + id +
          " no se ha encontrado");
    return todo; }
  //.....
```

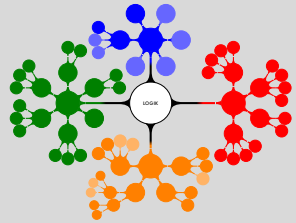
Recurso III

```
@GET  // Para el navegador
@Produces(MediaType.TEXT_XML)
public Todo getTodoHTML() {
  Todo todo = TodoDao.INSTANCE.getModelo().get(id);
  if(todo==null)
    throw new RuntimeException("Get: Todo con " + id +
        " no se ha encontrado");
  return todo;  }
@PUT
@Consumes(MediaType.APPLICATION_XML)
public Response putTodo(JAXBElement<Todo> todo) {
  Todo c = todo.getValue();
  return putAndGetResponse(c); }
@DELETE
public void deleteTodo() {
  Todo c = TodoDao.INSTANCE.getModelo().remove(id);
  if(c==null)
    throw new RuntimeException("Delete: Todo con " + id
        +  " no se ha encontrado"); }
```

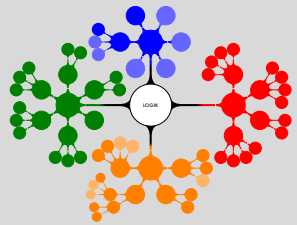
Recurso IV

```
private Response putAndGetResponse(Todo todo) {
    Response res;
    if(TodoDao.INSTANCE.getModelo().containsKey(todo.getId
        ())) {
      res = Response.noContent().build();
    } else {
      res = Response.created(uriInfo.getAbsolutePath()).
          build();
    }
    TodoDao.INSTANCE.getModelo().put(todo.getId(), todo);
    return res;
  }
```

Recurso V

```
//Para enviar datos al servidor como un formulario Web
@POST
@Produces(MediaType.TEXT_HTML)
@Consumes(MediaType.APPLICATION_FORM_URLENCODED)
public void newTodo(@FormParam(``id'') String id,
                    @FormParam(``resumen'') String resumen,
                    @@FormParam(``descripcion'' String
                        descripcion,
                    @Context HttpServletResponse
                        servletResponse)throws IOException) {
    Todo todo= new Todo(id, resumen);
    if(descripcion != null){
      todo.setDescripcion(descripcion);
    }
    TodoDao.INSTANCE.getModel().put(id, todo);
    servletResponse.sendRedirect(``../crear_todo.html'');
  }
```

Interfaz – Web

```
<!DOCTYPE html>
<html>
 <head>
  <title>Formulario Web para crearse un nuevo recurso</
      title></head>
<body>
  <form action="../nombreProyecto/rest/todos" method="POST
      ">
  <label for="id">ID</label>
  <input name="id" />
  <br/>
  <label for="resumen">Resumen</label>
  <input name="resumen" />
  <br/>
  Description:
  <TEXTAREA NAME="descripcion" COLS=40 ROWS=6></TEXTAREA>
  <br/>
  <input type="submit" value="Submit" />
  </form></body></html>
```