

RESTful webservices

vinod@javatrainer.org

OVERVIEW

- REST is a architectural style for an application which is highly based on web-standards and the HTTP protocol.
- REST was first described by Roy Fielding in 2000.
- In a REST based architecture everything is a resource.
- A resource is accessed via a common interface based on the HTTP standard methods.

OVERVIEW

- Every resource should support the HTTP common operations.
- Resources are identified by global ID's (which are typically URIs).

OVERVIEW

- In an REST architecture you typically have a REST server which provides access to the resources and a REST client which accesses and modify the REST resources.

OVERVIEW

- REST allows that resources have different representations, e.g. text, xml, etc.
- The rest client can ask for specific representation via the HTTP protocol (Content Negotiation).

HTTP METHODS

- GET
 - Defines a reading access of the resource without side-effects.
 - The resource is never changed via a GET request.
- PUT
 - Creates a new resource or updates an existing resource.
- DELETE
 - Removes the resources.
- POST
 - Updates an existing resource.

RESTFUL WEBSERVICES

- A RESTful webservice is based on the HTTP methods and the concept of REST.
- It typically defines the base URI for the services, the MIME-types it supports (XML, Text, JSON, user-defined, ..) and the set of operations (POST, GET, PUT, DELETE) which are supported.
- JAX-RS supports the creation of XML and JSON via JAXB.

JAVA, REST AND JERSEY

- Java defines standard REST support via JAX-RS (The Java API for RESTful Web Services) in JSR 311 .
- Jersey is the reference implementation for this specification.
- Jersey contains basically the core server and the core client.
- The core client provides a library to communicate with the server.

JAVA, REST AND JERSEY

- JAX-RS uses annotations to define the REST relevance of classes.
 - @Path, @GET, @POST, @PUT, @DELETE, @Produces, @Consumes, @Context

EXAMPLE : RESOURCE

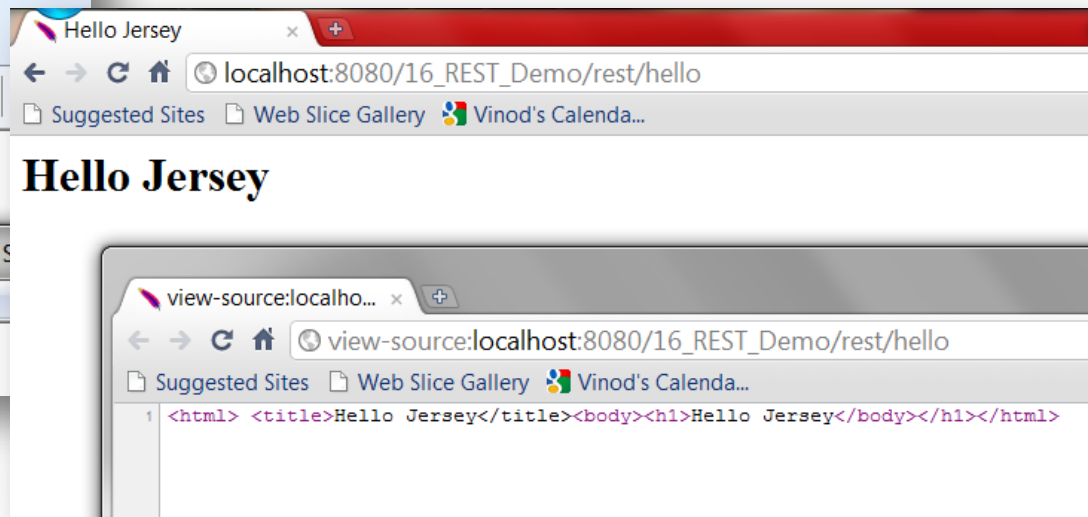
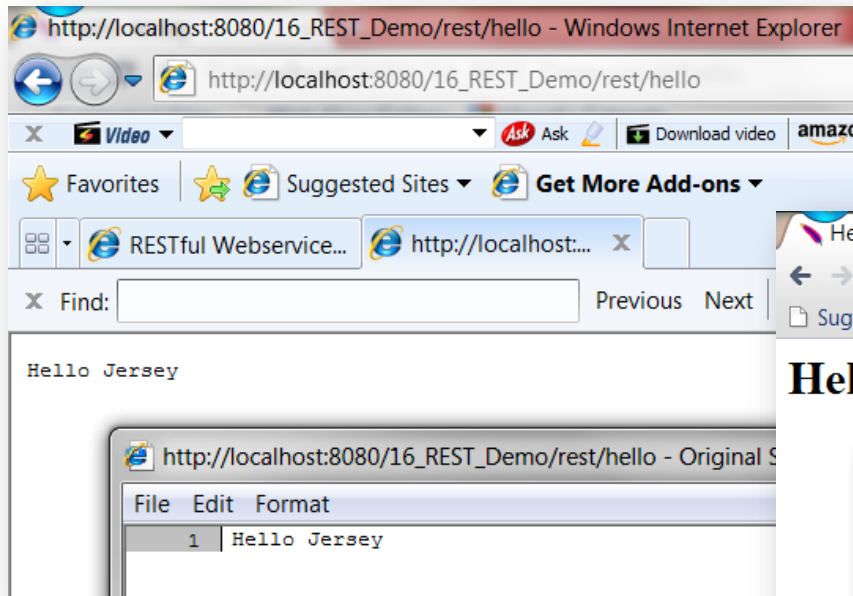
```
@Path("/hello")
public class Hello {

    @GET @Produces(MediaType.TEXT_PLAIN)
    public String sayPlainTextHello() {
        return "Hello Jersey";
    }

    @GET @Produces(MediaType.TEXT_XML)
    public String sayXMLHello() {
        return "<?xml version='1.0'?'><hello> Hello Jersey</hello>";
    }

    @GET @Produces(MediaType.TEXT_HTML)
    public String sayHtmlHello() {
        return "<html><title>Hello Jersey</title><body><h1>"
            + "Hello Jersey</h1></body></html> ";
    }
}
```

BROWSER OUTPUT



EXAMPLE: CLIENT

```
public class Client1 {
    public static void main(String[] args) {
        String path = "http://localhost:8080/16_REST_Demo/rest/hello";
        Client client = new Client();
        WebResource res = client.resource(path);

        String resp = res.accept(MediaType.TEXT_HTML).get(String.class);
        System.out.println(resp);

        resp = res.accept(MediaType.TEXT_XML).get(String.class);
        System.out.println(resp);

        resp = res.accept(MediaType.TEXT_PLAIN).get(String.class);
        System.out.println(resp);
    }
}
```

JAXB: OBJECT TO {JSON} AND <XML/>

```
@XmlRootElement
public class Todo {
    private String summary;
    private String description;

    public String getSummary() {
        return summary;
    }

    public void setSummary(String summary) {
        this.summary = summary;
    }

    public String getDescription() {
        return description;
    }

    public void setDescription(String description) {
        this.description = description;
    }
}
```

```
@GET
@Produces({ MediaType.APPLICATION_XML, MediaType.APPLICATION_JSON })
public Todo getXMLorJSON() {
    Todo todo = new Todo();
    todo.setSummary("This is my first todo");
    todo.setDescription("This is my first todo");
    return todo;
}
```

```
<terminated> Client2 [Java Application] C:\Program Files\Java\jdk1.6.0_21\bin\javaw.exe (Nov 19, 2010 5:54:55 AM)
{"summary":"This is my first todo","description":"This is my first todo"}
-----
<?xml version="1.0" encoding="UTF-8" standalone="yes"?><todo><description>
```