

# REStful Web Services

# Introduction

- ❖ The REStful Web Services are highly popular when implementing the Microservices architecture.
- ❖ The Representational State Transfer (REST) web services set a new style fro SOA systems.
- ❖ The term Representational State Transfer was introduced in 2000 by Roy Fielding in his doctoral dissertation.

# The REStful Services Architecture

- ❖ REST stands for Representational State Transfer. It is a web service architecture that focuses on the system resources.
- ❖ Each resource is identified by a URI (Unified Resource Identifier).
- ❖ Accessing the resources is done using HTTP. The server reply is the representation of the resource we try to access. This representation is usually a JSON document.

# The REStful Services Architecture

- ❖ Web Services that use the REST architecture are called RESTful Services.
- ❖ Most RESTful services are being used via a simple HTTP GET request for which they reply with a simple JSON document.

# Java API for REStful Web Services

- ❖ JSR 311 specifies Java API for REStful web services. It aims at allowing us to develop REStful web services in a standard way.

# Java API for REStful Web Services

The screenshot shows a web browser window displaying the Java Community Process (JCP) website. The URL in the address bar is <http://jcp.org/en/jsr/summary?id=311>. The page features the Java logo and the text "Java Community Process" and "Community Development of Java Technology Specifications". There are navigation links for "Press Room" and "Get Java Here", and a search box for "Search JSRs".

The main content area has tabs for "JSR", "Update", and "Expert Group". Below these tabs are links for "Summary", "Proposal", and "Detail (Summary & Proposal)". The page title is "JSRs: Java Specification Requests" and the specific request is "JSR 311: JAX-RS: The Java™ API for RESTful Web Services".

On the left side, there is a sidebar with a search box and several navigation links: "JSRs by Platform", "JSRs by Technology", "JSRs by Stage", "JSRs by Committee", and "List of All JSRs". Below the sidebar is a "My JCP" section with a "User ID" field (containing "username"), a "Password" field, and links for "Register for Site" and "Having problems logging in?".

The main content area contains a table with the following data:

Stage	Access	Start	Finish
Maintenance Release	<a href="#">Download page</a>	23 Nov, 2009	
Maintenance Draft Review 2	<a href="#">Download page</a>	02 Oct, 2009	01 Nov, 2009
Maintenance Draft Review	<a href="#">Download page</a>	27 Jan, 2009	03 Mar, 2009
Final Release	<a href="#">Download page</a>	10 Oct, 2008	
Final Approval Ballot	<a href="#">View results</a>	09 Sep, 2008	22 Sep, 2008
Proposed Final Draft	<a href="#">Download page</a>	15 Aug, 2008	
Public Review Ballot	<a href="#">View results</a>	27 May, 2008	02 Jun, 2008
Public Review	<a href="#">Download page</a>	02 May, 2008	02 Jun, 2008

The bottom of the screenshot shows the Windows taskbar with icons for Internet Explorer, Firefox, and other applications. The system tray on the right indicates the time is 10:00 AM on 6/13/2011 and the battery level is at 99%.

# Jersey Project

- ❖ Jersey is the reference implementation for JSR 311. We can easily integrate Jersey with Tomcat in order to develop a REStful web service.

# Jersey Project

http://jersey.java.net/

Aquarium Core Webtier WS/XML Tools

Login | Register | Join | Help

GlassFish » Jersey

Start Download Communicate Learn Contribute

Jersey is the open source, production quality, **JAX-RS (JSR 311)** Reference Implementation for building RESTful Web services. But, it is also more than the Reference Implementation. Jersey provides an **API** so that developers may extend Jersey to suit their needs. The **governance policy** is the same as the one used in the **GlassFish** project. We also use the same two licenses - **CDDL 1.1 and GPL 2 with CPE** - so, you can pick which one suites your needs better.

Have a question on JAX-RS or Jersey? then send email to the Jersey users list: [users@jersey.java.net](mailto:users@jersey.java.net).

Download NOW Jersey 1.7

Download NOW Jersey 1.0.3.1

Jersey User Guide

- Client, Server/Grizzly, WADL & JSON API
- Test Framework API
- Spring integration API
- Guice integration API
- MIME multipart API
- Client & Apache HTTP client API
- Atom & Apache Abdera API
- Simple HTTP server API
- OAuth support Signature API Server API Client API

Friends and Relatives

Wiki

Please join us at the **Community Wiki for Project Jersey** at

A java.net project

10:02 AM 6/13/2011



# Download Jersey

- ❖ You can easily download Jersey latest version browsing at <http://download.java.net/maven/2/com/sun/jersey/jersey-archive/1.7/jersey-archive-1.7.zip>
- ❖ In order to deploy a simple Jersey based RESTful web service we will need the following files:

`jersey-core.jar`

`jersey-server.jar`

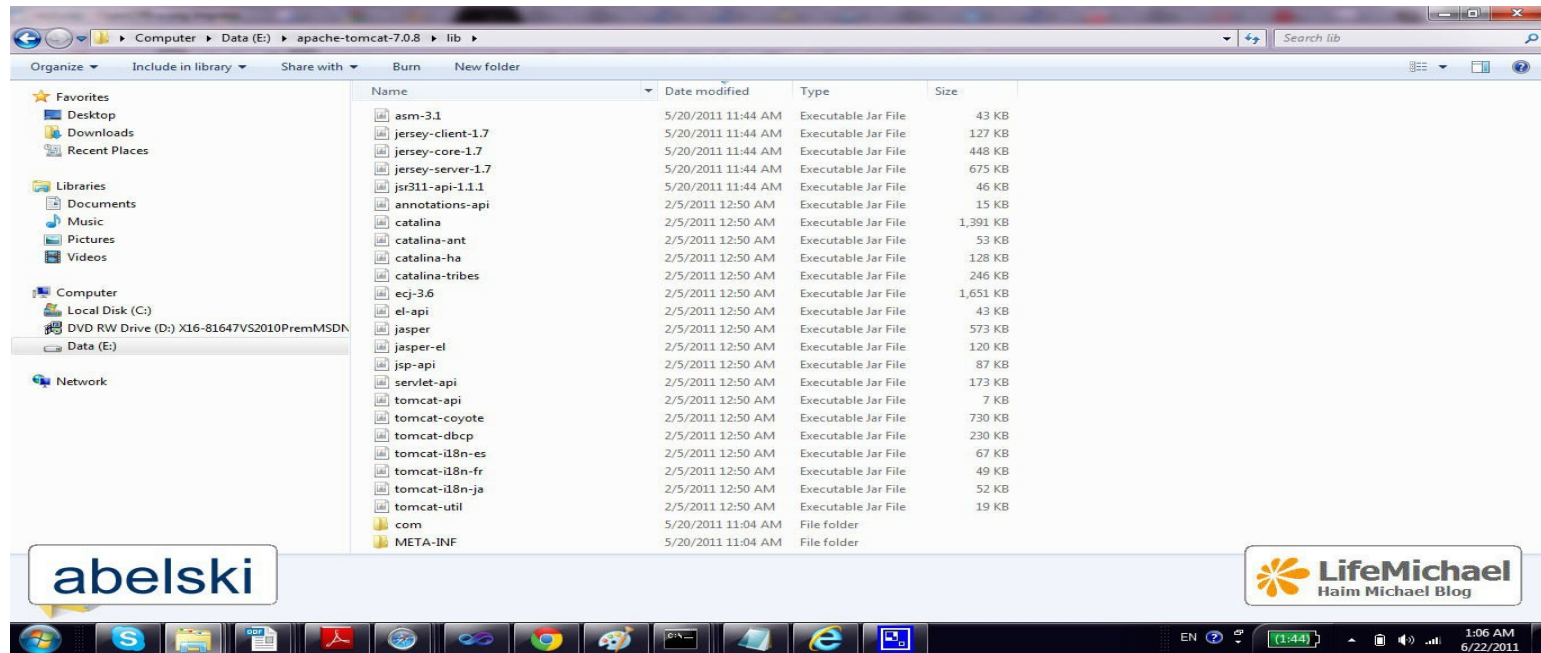
`jsr311-api.jar`

`asm.jar`

# Install Jersey

❖ We should first copy these jar files into the `lib` folder of our Tomcat installation.

# Install Jersey



06/22/11

© 2009 Haim Michael (Web Services, SOAP)

11

# Deployment

- ❖ We should define the Jersey servlet dispatcher in our `web.xml` file.

# Deployment

```
<servlet>
  <servlet-name>My Jersey REST Service</servlet-name>
  <servlet-class>
    com.sun.jersey.spi.container.servlet.ServletContainer
  </servlet-class>
  <init-param>
    <param-name>com.sun.jersey.config.property.packages
    </param-name>
    <param-value>
      com.abelski.samples
    </param-value>
  </init-param>
  <load-on-startup>1</load-on-startup>
</servlet>
<servlet-mapping>
  <servlet-name>My Jersey REST Service</servlet-name>
  <url-pattern>/rest/*</url-pattern>
</servlet-mapping>
```

this param defines the package where jersey should look for the classes we defined

we should add it into our web.xml configuration file

# Deployment

- ❖ We can develop the class that will be instantiated and function as a RESTful web service.

# Deployment

```
package com.abelski.samples;

import javax.ws.rs.Path;
import javax.ws.rs.GET;
import javax.ws.rs.Produces;
import javax.ws.rs.core.MediaType;
@Path("/hello")
public class HelloWorld
{
    @GET
    @Produces(MediaType.TEXT_PLAIN)
    public String sayHello()
    {
        return "Hello World";
    }
}
```



# Deployment





# Query String Parameters

- ❖ We can easily extract the value of a specific query string parameter by using the `@QueryParam` annotation.

# Query String Parameters

```
@Path("/crx")
public class CurrenciesExchangeRate
{
    @GET
    @Produces(MediaType.TEXT_PLAIN)
    public String getCurrencyExchangeRate(
        @DefaultValue("ILS") @QueryParam("currency")
        String currencyName)
    {
        double reply = 1;

        if(currencyName.equals("ILS")) reply = 1;
        if(currencyName.equals("USD")) reply = 3.4;
        if(currencyName.equals("EUR")) reply = 4.9;
        if(currencyName.equals("GBP")) reply = 6.5;
        if(currencyName.equals("CA")) reply = 3.9;
        return "<rate>" + reply + "</rate>";
    }
}
```

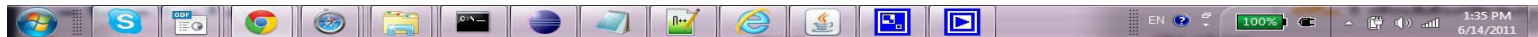
# Query String Parameters



A screenshot of a web browser window. The address bar shows the URL `localhost:8080/jerso/rest/crx?currency=CA`. Below the address bar, the response content is displayed as `<rate>3.9</rate>`.

abelski

 **LifeMichael**  
Haim Michael Blog



# Path Segments

- ❖ We can easily associate different methods with the various URL possibilities. In other words, we can specify different methods to be invoked in according to the URL path that was used for using the web service.

# Path Segments

```
@Path("/crx")
public class CurrenciesExchangeRate
{
    @GET
    @Produces(MediaType.TEXT_PLAIN)
    public String getCurrencyExchangeRate(
        @DefaultValue("ILS")
        @QueryParam("currency") String currencyName)
    {
        double reply = 1;
        if(currencyName.equals("ILS")) reply = 1;
        if(currencyName.equals("USD")) reply =
        3.4; if(currencyName.equals("CA")) reply =
        3.9; return "<rate>"+reply+"</rate>";
    }
    @GET
    @Path("/count")
    @Produces(MediaType.TEXT_PLAI)
    public String getCount() {
        return "<gogo>99</gogo>";
    }
}
```



# Path Segments



abelski

 LifeMichael  
Haim Michael Blog