Available Online at www.ijcsmc.com

International Journal of Computer Science and Mobile Computing

A Monthly Journal of Computer Science and Information Technology

ISSN 2320–088X
IMPACT FACTOR: 5.258

IJCSMC, Vol. 5, Issue. 6, June 2016, pg.398 – 400

Visual Representation of REST Resources

Rajendra Satpute[1], Prof. P. M. Chawan[2]

[1, 2] Department of Computer Engineering and IT, VJTI, Mumbai, India

[1] rajendra.satpute@gmail.com; [2] pmchawan@vjti.org.in

***Abstract*— Resource-oriented architecture is based on Resources, URI, representations and link between them. URI provided by REST are consumed by some applications, but it does not have pretty looks for end users to read directly. There is a way to access some URI and get the webpage directly as response without some presentation layer in between. This paper focuses on representation of URIs.**

***Keywords*— Representation of REST resources, REST resources, Viewable, JSTL and REST, Visual resources**

I. Introduction

Currently, Resource-oriented Architecture is being discussed and used in many applications. Applications communicate with each other using RESTful interfaces using JSON or XML as data format. If user needs to access resource directly and not through any application, some presentations is required. Here, some basic concepts of Resource-oriented Architecture, REST and JSTL are discussed in brief. Next, use of JSTL for representation of URIs is shown. The environment used for this implementation is Jersey and Eclipse J2EE IDE.

A. Resource-oriented Architecture

Resource-oriented Architecture is an architectural style which supports internetworking of resources. A resource can be accessed through URI assigned to it. Unlike Service Oriented Architecture, Resource Oriented Architecture is more concerned about resources than services. We can compare SOA vs. ROA as Operations vs. Data, or Verb vs. Noun.

Resource-oriented architecture is based in four concepts, resources, names for the resources (URI), their representations and link between the resources. And it is also based on four properties, Addressability, Statelessness, Connectedness and uniform interface.

Resource Oriented Architecture uses RESTful interfaces for design and development of softwares. Different software frameworks or tools like Jersey, RESTlet, Django, REST Easy are used which provide features of ROA.

B. REST

Representational State Transfer is a Stateless, Cacheable, Client-Server architectural style. RESTful applications communicate over HTTP. A path is assigned to each REST resource and it is in the form of http://example.com/webapp/info. REST uses GET, PUT, POST, DELETE methods for data transmission. Different content or media types are used. It supports custom media types as per the requirements. Popular among existing ones are JSON, XML for data transmission. Form parameters are used to capture data from form submission and path parameters are used to capture data from URI query.

While UI development, we can use JavaScripts with REST for real time data transfer. But, there is also another way to provide UI for URI using Viewable and JSTL.

C. JSTL

JSTL stands for JavaServer Pages Standard Tag Library. It is a collection of extended useful JSP tags used for common tasks including data processing, database access, conditions, loops. JSTL helps to reduce use of Java Code inside JSP pages. The use of different tag sets helps to make code maintainable.

II. REPRESENTATION OF URI

To begin with, the setup involves Eclipse J2EE IDE, Maven with jersey-quickstart-webapp archetype. The major dependency in pom.xml file includes jersey-mvc-jsp. This library contains Viewable class that is used to return a viewable JSP as a response for URI.

In web.xml, init-param is added with param-name jersey.config.server.provider.classnames and param-value org.glassfish.jersey.server.mvc.jsp.JspMvcFeature to tell that this application is using JSP MVC Feature of Jersey. Also we need to give the base path for JSPs by adding param-name jersey.config.server.mvc.templateBasePath.jsp with param-value /WEB-INF/jsp. Here /WEB-INF/jsp is used because WEB-INF is not a public directory and hence a user can not access JSPs directly. These JSPs will be accessed only by Java Classes. One can change the path as per need, but JSPs inside WEB-INF folder is recommended.

Once this setup is done, we can directly return a Viewable response for a resource as shown in Fig. 1 -

```java
import javax.ws.rs.GET;
import javax.ws.rs.Path;
import javax.ws.rs.Produces;
import javax.ws.rs.core.MediaType;

import org.glassfish.jersey.server.mvc.Viewable;

@Path("/")
public class Index {

    @GET
    @Produces(MediaType.TEXT_HTML)
    public Viewable index() {
        return new Viewable("/index");
    }

}
```

Fig. 1 Simple Viewable response without parameters

We can give a JSP file’s name as parameter to Viewable. We need not include .jsp extension to the file as shown. This will display provided JSP in browser. Note that org.glassfish.jersey.server.mvc.Viewable import is essential. We can also pass parameter to this JSP as shown in Fig. 2 -

```java
@GET
@Produces(MediaType.TEXT_HTML)
public Viewable index() {
    String username = "temp";
    Posts posts = new Requester().fetchAllPosts(username);
    Map<String, Object> map = new HashMap<String, Object>();
    map.put("posts", posts);
    map.put("name", username);
    List<Post> list = new ArrayList<Post>();
    list = posts.getPostList();
    map.put("postlist", list);
    return new Viewable("/index",map);
}
```

Fig. 2 Viewable response with parameters

This map passed as parameter can be accessed in JSP using JSTL and JSP-Expression Language as Shown in Fig. 3. Passed parameter map can be accessed as ${it} in JSP-EL. This way we can use URIs to represent data using JSP-EL and JSTL.

```
<table style=" width:80%; overflow: scroll; table-layout:fixed;
        margin-left:10% !important; margin-right:10% !important;">
    <tr>
        <td width="150" style="vertical-align:top;">
            <div class="w3-card-4" style="width:150 px; overflow: auto; padding:8px;">
                Name : <c:out value="${it.name}"></c:out>
            </div>
        </td>
        <td width="10"> </td>
        <td style="display:inline-block; vertical-align:top;">
            <c:forEach items="${it.postlist}" var="post">
            <div class="w3-card-2" style="width:100%; overflow:auto; padding:8px;">
                <c:out value="${post.title} "></c:out><br />
                <c:out value="${post.shortContent} "></c:out><br />
                <c:out value="${post.content} "></c:out><br />
            </div>
            <br />
            </c:forEach>
        </td>
    </tr>
</table>
```

Fig. 3 Using parameters passed through Viewable in JSTL

III. ADVANTAGES

Using JSTL with REST, we can take advantage of path parameters in URIs of REST. Instead of static links, users can pass their parameters through path to access particular content.

We can give different responses for the same path with different methods.

IV. CONCLUSIONS

JSTL with REST represents URI in form of HTML pages. This is advantageous when we need to access particular non-static path. This can be used either as a presentation layer for the application or just as a admin display interface for a particular layer. Through this admin display interface, we can configure changes for the particular layer without need of direct access to layer or need of any other layer to access the layer.

REFERENCES

[1] Dominique Guinard, Vlad Trifa, Friedemann Mattern, Erik Wilde, *From the Internet of Things to the Web of Things: Resource-oriented Architecture and Best Practices*, Architecting the Internet of Things, Springer, 2011.

[2] Lucchi, Roberto, Michel Millot, and Christian Elfers, *Resource oriented architecture and REST.* Assessment of impact and advantages on INSPIRE, Ispra: European Communities , 2008.

[3] Fielding, Roy Thomas, *Architectural styles and the design of network-based software architectures*. Diss. University of California, Irvine, 2000.

[4] Battle, Robert, and Edward Benson. *Bridging the semantic Web and Web 2.0 with representational state transfer (REST).* Web Semantics: Science, Services and Agents on the World Wide Web 6.1: 61-69, 2008.

[5] Roy T. Fielding and Richard N. Taylor. *Principled design of the modern Web architecture*, In Proceedings of the 22nd international conference on Software engineering (ICSE '00). ACM, New York, NY, USA, 407-416, 2000.

[6] Heaton, Jeff, *JSTL: JSP Standard Tag Library Kick Start.* Sams Publishing, 2002.

