RESTful, Resource-Oriented Architecture with OHH4RIA: a Model-Driven Approach

Author1 and Author2 and Author3 and Author4

Research Group, University of ?, City, Country
author1@domain.com, author2@domain.com, author3@domain.com, author4@domain.com
http://www.website.com

Abstract. dfdfdfdfd

Key words: rest, restful, resource-oriented, ROA, OOH4RIA, model-driven, MDD

1 Evaluation

This section evaluates the work by analyzing its applicability on web wide service available on the Internet. In particular, we analyze and discuss how Google Maps Web services could be annotated if using our framework. Particularly we will model the functionalities implemented by the GMap2, the most important class within the Maps API responsible for managing and creating maps. Among many available methods we chose four of them just for the sake of illustration:

- addMapType(type:GMapType) – Adds a new map type to the map. The class GMapType provides some predefined map types in the API.
- getCurrentMapType() – returns the currently selected map type.
- removeMapType(type:GMapType) – Removes the map type from the map.
- getMapTypes() – Returns the array of map types, i.e. GMapType[] registered with the currently used map.

In order to reproduce the GMap2 behavior through REST services, we need to define GMapType as a resource and map the GMap2 RPC methods into REST services. Table 1 shows the GMap2 RPC methods followed by their respective REST URL paths. Note that each interface has a path assigned to it. This path can include PathParams (i.e. URI template parameters or a path segments containing the template parameters) enclosed by the path-parameter-name. The software architect can edit the path. On the other hand, operations can also define a path which is appended to the interface path. Operations inside an interface are grouped depending on the mapping to the corresponding HTTP method, that is, inside an interface we can have as many operations groups as HTTP methods. Of course, the software developer can include new operations in a group, delete operations from a group, move an operation from one group to other, and so on.
Table 1. GMap2 Methods represented as REST Services

<table>
<thead>
<tr>
<th>Nr.</th>
<th>Native Method</th>
<th>Rest Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>addMapType(type:GMapType)</td>
<td>gmap2/gtype/maptype</td>
</tr>
<tr>
<td>2</td>
<td>getCurrentMapType()</td>
<td>gmap2/currentmaptype</td>
</tr>
<tr>
<td>3</td>
<td>removeMapType(type:GMapType)</td>
<td>gmap2/gtype/removemaptype</td>
</tr>
<tr>
<td>4</td>
<td>getMapTypes()</td>
<td>gmap2/maptypes</td>
</tr>
</tbody>
</table>

2 Discussion and Conclusion

In this paper we extend OOH4RIA, an approach that proposes a model-driven development process for RIAs, to support RESTful web service generation at server side. To this, we introduce JAX-RS annotations into the RIA Component Model. The applicability of the solution is shown on Amazon Simple Storage Service (Amazon S3) that is used as the running example as seen in Section xx. The work requires further evaluation from an end user point of view and this will possible as long as the toll is completed implemented.

3 Acknowledgment

The research leading to these results is part of the project "KiWi - Knowledge in a Wiki" and is partly financed by the European Community’s Seventh Framework Program (FP7/2007-2013) under grant agreement No. 211932.

References