

REST BASED WEB SERVICES - A Configuration File Based Technique

Ms. Sharon Coelho

Vidyavardhini's college of Engineering & Technology, Mumbai University,

Abstract

Web services can be seen as an outgrowth of work on distributed information systems. Web services attempt to solve problems, such as service coordination and service composition that distributed information systems have addressed in various ways (but not always successfully) in the past. This paper highlights the Core Working Techniques and challenges of Web Services and examines what new techniques and/or benefits REST based Web services can bring to distributed information systems.

1. INTRODUCTION

REST stands for Representational State Transfer. The term was introduced by Roy Fielding in his doctoral dissertation. REST is an architectural style for designing networked applications. It is an alternate to using complex mechanisms like COBRA, RPC, and SOAP to connect between client and server.[1]

REST defines a set of architectural principles by which you can design Web services that focus on system's resources, including how resource states are addressed and transferred over HTTP by a wide range of clients written in different languages. If measured by the number of Web services that use it, REST has emerged in the last few years alone as a predominant Web service design model. In fact, REST has had such a large impact on the Web that it has mostly displaced SOAP- and WSDL-based interface design because of its considerably simpler style of use.[3]

2. PROBLEM DEFINATION

In a service-oriented application, the developers of the system are divided into varied locations and have different ways of implementing the design of an application. As the platforms are varied and the technology used is different, rendering the information to the client becomes a key problem for the service provider which needs to be addressed carefully. REST is a resource-oriented design which till date is the most effective solution to the problem.

The Key Principles of REST are as follows:

1. Everything in a system is a resource. Application Servers, Modules inside the application, Web Server, Hardware Equipment such as printer, copier etc. are all resource

2. URL is the unique identifier of each resource

There are many issues for rendering the resource representation.

i. There are many kinds of representation for one resource; and different requests may ask for different representations.

ii. We may consider writing the representations manually but it makes the work load extremely huge. Some resources may need multiple representations, hence it is hard to figure out how many representations do we need to fulfill the requirements of end user

iii. To use a predefined manual representation, the client may have to edit the code as per its requirement which makes use of the resource very challenging

iv. It is not optimum to write the resource representation manually; This paper will highlight the automated Mechanism used for transfer of representation of the resources.

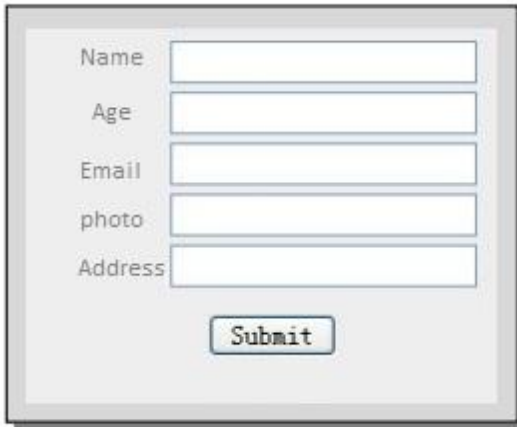
3. WHAT IS REPRESENTATION

As mentioned earlier, every module inside the application, every class inside the module, everything to be precise is a resource. We use a class definition as a resource and discuss different solutions with their advantages and disadvantages

```
public class Student
{
private String name;
private int age;
private String email;
private String address;
private string password;
private String photo;
}
```

Figure 1. Resource Class Student

There can be many representations of this class. The general ones used are Plain Text - HTML, Form Based and Table format representations which are shown in the figure.



(a)



Name : Sharon Coelho
Age : 26
Email : coelho_10@rediffmail.com
Photo : c:\My Documents\sharon.jpg
Address: 14, Clair Street, Hounslow, UK. TW3 1SE

(b)



| | |
|---------|---|
| Name | Sharon Coelho |
| Age | 26 |
| Email | coelho_10@rediffmail.com |
| Photo | c:\My Documents\sharon.jpg |
| Address | 14, Clair Street, Hounslow, UK. TW3 1SE |

(c)

Figure 2. HTML representation (a) Form Format, (b) Text Format and (c) Table Format

It is very important to highlight the fact that these simple representations have some notable deficiencies. Although these are the methods of representations, It should not be the way of designing the application. The deficiencies of these representations are:

--It cannot control the layout of the representation. For example if the student wants the name in red, the client developer cannot alter its representation as it is fixed.

--It cannot determine the visibility of the password or address field. For hierarchy of users, the exposure of information changes and hence the predefined representation does not allow the client developer to modify or alter its representation to suit the needs of the user.

Client developer cannot implement cascading of various classes and subclasses within an application. Hence, it is mandatory to programmatically (through scripts) provide multiple representation options to the client rather than a fixed one which would give client developer more operational functionality over a resource.

4. PROPOSED CONFIGURATION FILE TECHNIQUE

The solution proposed in this paper represents the idea of having a separate configuration file for each resource. In the example mentioned in section 3, resource class Student has multiple representations for different set of end users. Although it's very difficult to point out exactly how many end user representations are needed, but a general form of representation such as name in red, Age and address visibility, cascading of multiple representations etc. can be easily considered and implemented.

Hence we create 2 configuration files of the resource class Student which will be stored on the Server. The client will request for a specific representation and the configuration file will be transferred in the form of an XML file using Simple Object Access Protocol.

To demonstrate the above solution we implement two representations.

Representation 1

(color in RED, hide password, Age and Address)

```
<root className="Student" name="default">
  <configType>text</configType>
  <configIsPassword>>false</configIsPassword>
  <configShowPassword>>false
</configShowPassword>
  <configShowAddress>>false</configShowAddress>
  <configShowAge>>false</configShowAge>
  <field name="name">
    <color>Red</color>
  </field>
```

Representation 2

(show password, Age, Address,hide email)

```
<root className="Student" name="default">
  <configType>text</configType>
  <configShowPassword>>true
</configShowPassword>
  <configShowAddress>>true
</configShowAddress>
  <configShowAge>>true</configShowAge>
  <configShowemail>>false</configShowAge>
```

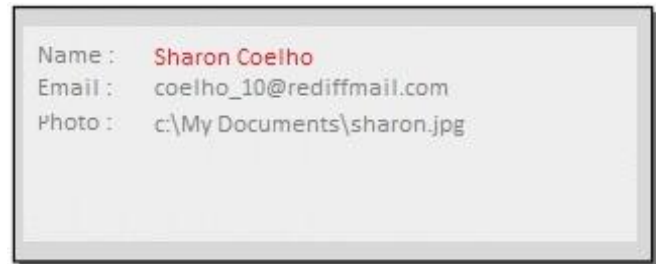


Figure 3. Representation 1 Result



Figure 4. Representation 2 Result

5. WORKING DETAILS

1. The above representation rules will have to be followed by all the client developers.
2. A DOM (Document Object Model) Tree structure is formed on the Server. The class becomes the root of the tree and subclasses and fields become the nodes of the tree.
3. SOAP (Simple object Access Protocol) is used to access the DOM tree. An object for each class is created by default which grants the access to the class fields.
4. The requested resource is transferred from server to client in the form of XML as it most widely acceptable mode of document transfer within machines.

5. A representation is Selected for the resource, depending upon the user request and the configuration file of the representation is accordingly transferred to the client via RMI/RPC or Java Beans.
6. The same resource can be accessed by multiple clients, each having different representations, depending upon the need of the end user.
7. Making changes to configuration file will allow the client developer to control layout, content and pattern of the representation of the resource.

6. CONCLUSION

Hence, Going by the protocols of the above proposed technique, We can not only solve the problem of load balancing of client requests in distributed platform but can also reduce the complexity of client developer by changing the state of representations of a resource form client to client.

7. REFERENCES

- [1] Fielding, R. T, "Architectural Styles and the Design of Network-based Software Architectures (PhD Thesis) ", UC Irvine, Information and Computer Science, 2000
- [2] Leonard Richardson, Sam Ruby and David Heinemeier Hansson, "RESTful Web Services", O'Reilly Media, Inc, 2007.
- [3] Daniel Szepielak, "REST-based Service Oriented Architecture for Dynamically Integrated Information Systems", PhD Symposium at ICSOC 2006. 2006.