RESOURCE´S RELATIONSHIPS IN THE DESIGN OF COLLABORATIVE WEB APPLICATIONS

Enrique Barra Arias
ebarra@dit.upm.es

Antonio Mendo Hernández
amendo@dit.upm.es

David Prieto Ruiz
dprieto@dit.upm.es

Antonio Tapiador del Dujo
atapiador@dit.upm.es

Juan Quemada Vives
jquemada@dit.upm.es

ETSI de Telecomunicación - Universidad Politécnica de Madrid
Avenida Complutense, 30 - 28040 - Madrid (Spain). Phone: +34 91 549 57 00

ABSTRACT

At the moment of designing a web application, we usually run into the problem of how to deal with logical connections among resources. These connections have important implications in the operations that we take on a certain resource and its representation, as we could verify in the design of the collaborative web application that we have developed, the Virtual Conference Centre. For those reasons, in this paper we analyze the relationships among resources, especially focused on collaborative web applications, and we propose some solutions and good practices for the difficulties that we have encountered.

KEYWORDS: REST, resource design, collaborative web applications, relationships.

1. INTRODUCTION

Web based collaboration tools are becoming one of the main ways for group and distance collaboration. Due to the easy accessibility to the Internet (e.g. via Wi-Fi) and the popularization of web browsers, users have a nearly ubiquitous means of collaboration.

Every collaboration tool needs to manage resources and the relationships among them. As it is defined in RESTful Web Services book [1] "*A resource is anything that's important enough (in your web application) to be referenced as a thing in itself...*" and *"... it has to have at least one URI"*. Also it is important to define relationship, understood as *"a general term covering the specific types of logical connections found on class and object diagrams"*.

The Representational State Transfer (REST) is an architectural style for hypermedia distributed systems based in these resources and relationships, introduced by Roy Fielding in 2000 [2]. It is defined by a set of constraints, and it is inspired by the features and principles that made World Wide Web successful. In fact, The Web is the best instance of a REST architecture style.

In this paper we analyze, from the REST point of view, the relationships among resources as a key part of web collaboration tools architecture. We will see how these logical connections affect the representation of a resource or a collection of resources, how these cases arose in the design of the collaborative web application that we have developed, the *Virtual Conference Centre (VCC) [3]*, and we will propose some good practices and solutions to solve the obstacles we encountered.

2. DESIGN OF RESOURCE´S RELATIONSHIPS IN COLLABORATIVE TOOLS

Web resources are related among them, they share connections and references. According to the UML [4] instance level relationships there are four kinds of relationships: aggregation, external links, composition and association.

In next sections we will discuss some good practices to be taken into account when designing your web based collaborative applications, and we will show how implement them by using real examples taken from the *VCC*. We believe it will help developers design better applications, avoiding well known problems and promoting readability.

The *VCC* is a Web application developed by the *Universidad Politécnica de Madrid (UPM)* within *GLOBAL* project, a research project supported by the European Commission's seventh framework programme. It offers a complete environment for web collaboration; besides other classic tools like versioned document repository or dialog forums, the *VCC* offers users a service for creating, sharing, recording and publishing distributed events, which may vary from small meetings to big conferences. The *VCC* and specially the "*VCC* events" help project members to get closer and to collaborate as if they were office partners.

2.1 URI Design

URIs are one of the key building blocks of REST architectures like the *WWW*. They are used to reference resources and they are part of a resource's representation; in fact, every resource must be referenced by at least one URI. That way we can link one resource to another (inside or outside our application).

Although it is not mandatory, URIs should be human friendly and meaningful. Resources in web collaboration tools are usually linked in a big variety of media like emails, web pages and also in meetings and presentations. Providing meaningful URIs facilitate collaboration. Besides, resources are better indexed by web crawlers when their URIs are descriptive. Google provides webmasters [5] with advices in this direction.

Composition relationships, such as the container-content one, are usually reflected on URIs. The hierarchical order provides a natural way to nest resources in the URI. They have this appearance: */container/contents*. If we use an example from the *VCC* like */spaces/global/events/monthly_meeting*, we realize there is a composition relationship between the event and the space (notice that in the *VCC*, group collaboration is organized into spaces). This URI clearly shows where you are going if you click on it, and so, it will influence on the user's decision to follow it and, finally, collaborate.

Sometimes, when contents have a lot of nesting, we end up using too long URIs: e.g. */spaces/global/events/meeting/attachments/session-1.odp*. They provoke visual noise and they lack persistency, as relationships among resources can change. That is why we do not need to always keep all the containers in the path, e.g. when the content's identifier is unique per application scope. We can decouple some of the resources and make the URI shorter and more persistent. We need to find a compromise between URIs being descriptive and being persistent.

2.2 Resource Representations

Another key point in a RESTful design is the representation of a resource. Typically resource representations are divided in two groups: **collections** and **elements**.

On one hand, forums, blogs or photo albums are just a few examples of collections that appear in most of the web collaboration tools. In a generic case, the relationship between the collection and its container can be found in the URI: */spaces/global/attachments* or */spaces/global/events/monthly_meeting/attachments*. In both URIs a collection of attachments is shown. The first one is the representation of the space document repository, containing all the attachments that belong to the "*global*" space; and the second one only returns those that belong to the event called "*monthly meeting*" of the "*global*" space. Furthermore, the first

collection will be shown, in the HTML representation, inside the context of the space (surrounded with information and links related to the space), while the second one will be shown inside the event context.

On the other hand, element representations show the state of a single resource, like a picture, a comment in a blog or a video. Usually the relationships of a resource with other resources are included at the resource representation, and means of communication are provided through links.

These relationships have also several consequences in resource manipulation operations. More specifically, we will see how WRITE operations get affected in these scenarios, and how we applied it to the *VCC*.

There are three ways of creating resources that are related to some other resource:

1. Providing a reference to the container in the representation sent to the server: used in the cases that those relations were optional, e.g. posts that are related to events.
2. Using a URI that includes the container: Using POST requests to a URI that contains the space in the path is our preferred way for the creation of resources that need a space to exist.
3. Including the contents in the container representation when manipulating the container: we send nested resource representations in some cases, such as creating posts with several attachments. The web development framework tool that we have used, Ruby on Rails [6], provides some facilities like the one called "nested resources" which, along with the use of JavaScript to include several file inputs in the HTML form, make this method very convenient.

Finally, update and delete methods are independent of container relations, beyond the representation sent to the server when updating resources. These methods do not need descriptive URIs, neither the full URL with all the nested resources.

3. CONCLUSIONS

REST gives us some useful guidelines that can be used in the design of collaborative web applications. It is focused on resources and their representations, and we have learnt that the key for a successful design is the resource identification and definition.

In this paper, in order to face the design and implementation of the *VCC*, we have classified the relationships present in collaborative web applications' resources, according to the UML instance level relationships. Taking into account the compromise between URI persistency and URI description, we have analyzed how and when these relationships should be reflected on URIs. Clear and meaningful URIs facilitate collaboration, so it is important bringing the most relevant relations into URIs. We have also studied how relations affect the resource's representation, documenting three different ways of establishing relationships. Finally, we have supported our study and conclusions with a real web service design.

4. REFERENCES

[1] Leonard Richardson, Sam Ruby. May 2007. *RESTful Web Services*. O'Reilly.

[2] Fielding, Roy Thomas. 2000. Architectural Styles and the Design of Network-based Software Architectures. Doctoral dissertation. University of California, Irvine.

[3] The Virtual Conference Centre. http://www.globalplaza.org

[4] Scott W. Ambler. 2004. *The Object Primer: Agile Model-Driven Development with UML 2.0*. Cambridge University Press, New York, USA.

[5] URL structure, Webmaster Tools Help, http://www.google.com/support/webmasters/bin/answer.py?hl=en&answer=76329

[6] Sam Ruby et al, March 2009. *Agile Web Development with Rails: Third Edition.* Pragmatic Bookshelf.