

CIVTAT: A Web Application Based on REST And XForms

Sebastià Vila-Marta
Eulàlia Formentí-Famadas
Ramon Navarro-Bosch
Maria Soler-Climent

Free Software Chair
Technical University of Catalonia
sebas@lsi.upc.edu

July 2006

Abstract

Most of web applications built today are founded on application servers. Those applications maintain the user interaction state in the server. This fact poses some important scalability issues for this applications. In the project CIVTAT, we are developing a web application to manage the inhabitants data of a municipality. The architecture of CIVTAT concentrates the interaction state in the client. Thus very light servers are obtained. The application is heavily founded on REST paradigm and it makes an intensive use of W3C standards.

1 Introduction

Today there are many old information system applications that are being migrated to web applications. In this paper we name *web applications* to those client-server applications which use a generic web user agent as a client. Additionally, we use *desktop applications* to refer to any other applications. Web applications are becoming pervasive because of its advantages. The most important of them is the lack of ad-hoc clients to be maintained.

In this architectures the server has a prominent role. Most of web applications are built on application servers. Application servers are complex mostly because they should deal with interaction state while providing efficiency and scalability. This complexity poses some important problems: application servers are complex to manage and they require a lot of resources.

In this paper we describe our experience in designing and implementing a web application following an opposite path. Our objective is to obtain

an application that concentrates as much state information as possible in the client side. This leads to simple servers which are very scalable. The application closely follows the REST architectural principles and it is tightly adhered to W3C standards.

The project, which is named CIVTAT [1], has two main objectives: to encourage a joint development of an application and to design a distributed application based on thin servers. This article focuses only on the technical aspects of the project.

The paper outline is as follows. In Section 2 we show the most salient drawbacks of applications servers. These problems are mainly related to state managing complexities. Then, in Section 3 we describe the prominent characteristics of CIVTAT application and the project goals. Section 4 is committed to the principles and techniques that are being used in CIVTAT project. The paper ends with some conclusions in Section 5.

2 Application server drawbacks

In an interactive application there is information that represents the interaction state. In desktop applications this is part of the application data. When an application follows a client-server architecture the user interaction state should be stored in the client, the server or in both ends.

Web architecture is based on a stateless concept. Web servers don't store interaction state. Every time an user agent sends an HTTP request to a server it is served independently of previous ones.

However, web applications have evolved to mimic the same user interaction patterns used by desktop applications. To reproduce those patterns interaction state should be managed. User agents are not well suited to manage this kind of interaction state and therefore it should be stored in the server.

A server uses to attend to several clients. Then, a server manages as many interaction state buckets as clients served. Every connection must be associated to the corresponding interaction state. This is usually achieved by storing a cookie in the user agent that uniquely identifies the bucket in the server. Application servers are specialized applications devoted to efficiently manage this. See for instance JOnAS, [8], which is based on J2EE architecture.

Despite the advances of application server technology, the following drawbacks remain:

- Memory space and computing power required by application servers grows linearly according to the number of clients. This a consequence of the state being stored in the server.
- Application servers are difficult applications to distribute.

- Application servers are complex applications.

3 CIVTAT: the application

In Catalonia town councils benefit of a high degree of management independence. This has lead to a very heterogeneous set of corporative information systems. The CIVTAT project wants to help in this situation by promoting the joint development of an important part of the information system: the list of inhabitants. This project is a joint effort of the Free Software Chair of the Technical University of Catalonia and the Manresa town council.

To achieve the project goals, the following requirements for the application were established:

- The application must use only free software and must be itself free software.
- The application should closely follow the existing standards.
- The application should be a web application. That is, all the clients should be web user agents.
- The application should have a low maintenance cost.
- The application should be as portable as possible to allow for distinct towns with distinct information systems to share most of it.

In Section 4, we describe the main principles and techniques used during the application design to achieve these goals.

Currently we are developing a prototype to provide a proof of concept for the designed architecture. This prototype can be reached from our web page <http://civtat.lafarga.cpl.upc.edu>. Following, three aspects of the prototype are described: the end user experience, the interface exhibited by the server to the clients and the server internals.

3.1 End user experience

Many current web applications are trying to emulate user interface practices of desktop applications. This is frequently explained by the principles of human interface design discipline. It is also well known that many of these fine grained interaction patterns require a precise control of interaction state. Web architecture, as conceived in [4], do not precisely fits in this paradigm. However, the number of web users increases every day. This shows in some sense that the user experience when surfing the web is good enough despite of being coarsely grained.

In this project we offer to the user an application that shows no difference with the web. The user can navigate by the web of the inhabitants data in

the same way that he navigates by the other pages of the world wide web. The same user actions remain valid: bookmarking a page, sending by email some page URL or printing a page. They have a coherent meaning which covers all the application. In the Figure 1 there is a screenshot of one these pages.



Figure 1: A page corresponding to an inhabitant data.

The application have also some specific tools. For instance there are services used to search specific web pages like the one corresponding to a given inhabitant. These services are shown to the user through forms.

3.2 Server interface

The server shows all the inhabitants data model through its interface. This is organized through a carefully designed URL space that uniquely identifies every data instance. When issued a GET operation on one of these URL's, an XML description of the instance data is returned by the server. This XML object embeds other URL's pointing to related information. This is done using XLink.

This allows for a double use. Web user agents can make use of these URL's because they contain an embedded XML processing order. This allows to a client to render the XML data by means of XSLT and CSS. Then, everyone of the URL's of the model correspond to a web page from an user standpoint.

In addition, this very same interface can be used by any program through any standard HTTP access library to fetch information from the server. This is named a REST-compliant web service.

In some sense, the interface exhibited by the server defines a higher level interface than that offered by the information system database back-end. At the same cost this common interface lets:

- To an end user to query and navigate through the data using a web browser.
- To data processing programs to access information through a higher level interface.

3.3 Server internals

The CIVTAT application server is implemented on top of Apache. Static content like CSS, XSL, forms, etc. are served using the ordinary Apache resources. More interesting is how the data model is exhibited through web services. This is accomplished by using the `python_mod` module and some lightweight Python coded servlets. These servlets are put on a layer that interfaces the application to the backend database. This layer lets to port the application to distinct back-end organizations.

4 CIVTAT: technical principles

CIVTAT closely follows the principles sketched below:

- Representational state transfer.

In [7] and [6], Fielding describes the representational state transfer style of architecture, in short REST. The REST model is built after accurately analyzing how the web is organized. The principles stated in the model have been used to guide the web development. See for instance [4].

REST provides very useful tips to build web applications that exhibit a good integration to the web. One of the most salient conclusions express that the web is stateless and thus the only agent responsible for interaction state is the user agent. When an applications breaks this principle it loses many of the advantages of web.

- URI space design.

CIVTAT exhibits all the application data model through a well structured set of URI's. This URI space is designed according to [3] and [4].

- Trivial web services.

Although web services are usually associated to SOAP protocol, the concept of web service, as defined in [2], is not related to any specific protocol. Indeed web services can be implemented exclusively on top of HTTP following the REST principles. This is known as REST-compliant web services, [2].

In CIVTAT, all the data model is offered through REST-compliant web services.

- Computing in the client.

Nowadays most of personal computers have a lot of processing power available. This allowed web user agents to increase functionality. Currently, most of user agents have an embedded ECMAScript interpreter,

an XSL processor available, etc. CIVTAT try to take advantage of this by displacing most of computation to the user agent. This is achieved using a broad set of technologies from the W3C: XML, XSLT, XPointer, CSS and XHTML.

By using this scheme, we obtain several benefits:

- A clean separation between static and dynamic content. This is very important because every kind of content have distinct server configuration requirements.
- An increased cache effectivity. This is a consequence of the static vs. dynamic content separation.
- This scheme smoothly integrates with the REST-compliant web services. The result is a very economic architecture.

All those contribute to a thinner server software and hardware.

- XForms.

XForms, [5], plays an important role in CIVTAT. The main source of interaction state in a web application comes from implementing insert and update operations. These operations are carried through form processing. In CIVTAT, to reach the goal of having a stateless server, we are using XForms. This allows us to delegate most of complex form processing to the client. Then, only the user agent cares about form processing state. This scheme requires web user agents with XForms processing capabilities.

5 Conclusions

CIVTAT is a free software application devoted to manage the inhabitants list of a town council. In this paper we have explained the technical aspects of CIVTAT. The most outstanding characteristics are the intensive use of W3C standards and the design oriented to thin servers. To reach the thin servers infrastructure goal, CIVTAT is based on a stateless paradigm inspired in REST. In some sense CIVTAT points to what some authors name as Web 2.0.

After implementing a prototype, the results are promising. We obtained good throughput on light servers and beta testers declare a good user experience when using the application. However, the lack of mature free implementations of XForms processors places a barrier in the implementation of insert and modify functionalities. We expect that XForms implementations will evolve fast enough.

Acknowledgements

We thank to Montse Morera and Marc Costa by their support. This project is being partially founded by Manresa's town council.

References

- [1] Civtat project web page, 2006. <http://civtat.lafarga.cpl.upc.edu>.
- [2] David Booth et al. Web services architecture, 2004. <http://www.w3.org/TR/ws-arch>.
- [3] Ian Jackobs et al. Uris, addressability, and the use of http get and post, 2004. <http://www.w3.org/2001/tag/doc/whenToUseGet.html>.
- [4] Ian Jacobs et al. Architecture of the world wide web, volume one. <http://www.w3.org/TR/webarch/>, 2004.
- [5] John M. Boyer et al. Xforms 1.0 (second edition), 2006. <http://www.w3.org/TR/xforms>.
- [6] Roy T. Fielding. *Architectural Styles and the Design of Network-based Software Architectures*. PhD thesis, University of California, Irvine, 2000. http://www.ics.uci.edu/~fielding/pubs/dissertation/fielding_dissertation.pdf.
- [7] Roy T. Fielding and Richard N. Taylor. Principled design of the modern web architecture. In *ICSE '00: Proceedings of the 22nd international conference on Software engineering*, pages 407–416, New York, NY, USA, 2000. ACM Press.
- [8] ObjectWeb. Jonas application server site. <http://jonas.objectweb.org>.