Building Web Applications with Protégé

Csongor Nyulas and Tania Tudorache

Stanford Center for Biomedical Informatics Research
Stanford University School of Medicine, Stanford, CA, 94305
csongor.nyulas, tudorache <at> stanford.edu

Abstract

As we are witnessing a steady increase of semantic content on the Web, there is a (maybe not so obvious) increase in the number of web-applications that are used to produce or consume that content. Protégé offers a great toolset to deal with almost every aspect of authoring and handling ontology based semantic content. Creating easy to use applications that present the power of Protégé in a lightweight and familiar manner, accessible through a web browser has an essential role in spreading the semantic technology. There are a number of tools and frameworks that can help the application developer to rapidly and easily develop such applications. In this paper we will show how we have used the Google Web Toolkit and the Protégé API to build a couple of web applications that provide means to a wide spectrum of users to create, edit and browse biositemaps.

Introduction

With the evolution of Web 2.0 [1] we are witnessing an ever increasing number of web applications that offer lightweight versions of what would have been desktop applications in the past. A few randomly selected well-known examples we could mention here are Google Docs[1], Google Calendar[2], Yahoo! Games[3], Zimbra Collaboration Suite[4], WebProtege[5], etc. Looking at arguments about the benefits and usability of web-applications compared to desktop applications, there is an observable trend showing that there are more and more web-applications that replace desktop applications, especially when: (i) the tasks to be solved are relatively simple, (ii) involve access to multiple resources from the web, (iii) require easy accessibility or (iv) involve user collaboration.

In this paper, we will describe our experiences, and some tips and tricks we have identified while developing a web application for creating biositemaps. The biositemaps initiative[6] was started by the Resourceome Working Group of the NIH Roadmap Initiative for National Centers of Biomedical Computing (NCBC)[7] and its goal is to enable technologies to address (i) locating, (ii) querying, (iii) composing or combining, and (iv) mining biomedical resources on the Internet. The solution proposed is inspired by Google's sitemap protocol but enhances it with the advantages of RDF over XML. The idea behind biositemaps is that all institutions with an interest in biomedical research can publish a biositemap.rdf file on their Internet site. Each biositemap.rdf file is simply a list of controlled metadata about resources (software tools, databases, material resources) that the publishing organization uses or believes are important to biomedical research. The key enabling technologies are the Biositemap Information Model (IM) - a list of metadata fields about a resource (like 'resource_name', 'description', 'contact_person', 'resource_type', etc.), and the Biomedical Resource Ontology (BRO) - a controlled

---

1 http://docs.google.com/
2 http://calendar.google.com/
3 http://games.yahoo.com/
4 http://www.zimbra.com/products/
5 http://protegewiki.stanford.edu/index.php/WebProtege
6 http://www.biositemaps.org/
7 www.ncbcs.org/

terminology to be used as the value of selected fields (like 'resource_type') that is used to improve the sensitivity and specificity of web searches.

Biositemap RDF files can be created, edited and viewed in several ways: with tools ranging from common text editors to specialized and powerful software applications like Protégé. Biositemap files can even be created and read programmatically by using the Biositemap Java API. Although this broad selection of tools is more then enough for tech-savvy people, the content of biositemaps is often created by persons with deep domain knowledge, but less profound technical skills. For this reason, we decided to create an editor for biositemap files, which is easily accessible, very simple to use, and ensures the correctness of the resulting biositemap files. In this paper, we present a simple web application, called Biositemap Editor, which we have built with the use of the Google Web Toolkit[8], running Protégé on the server side to create valid biositemap RDF files.

The Google Web Toolkit (GWT)

Google Web Toolkit (GWT) enables the web application developer to write AJAX front-end in the Java programming language. GWT then cross-compiles the Java code into optimized JavaScript that automatically works across all major browsers. During development, the developer can iterate quickly in the same "edit–refresh–view" cycle, which is possible when writing JavaScript code, with the added benefit of being able to debug and step through the Java code line by line. When the application is ready to be deployed, GWT compiles the Java source code into optimized, standalone JavaScript files.

Without a detailed explanation we present a short list of the major advantages that GWT brings to the software developer:

- The changes made in the Java files can be seen immediately in the web browser, without re-compiling.
- Developers can step through live AJAX code with the Java debugger.
- GWT compiles and deploys strongly optimized, cross-browser JavaScript.
- In addition to supporting an open ended set of transfer protocols, GWT also offers a simple, but efficient and sophisticated client-server communication with GWT RPC.
- Supports efficient application localization, and optimized JavaScript script downloads based on user profile.
- UI component can be easily reused across projects.
- Allows the usage of other JavaScript libraries and native JavaScript code.
- Easily supports the browser back button and history.
- Works with the Java Development Tool of choice, including testing with JUnit.

System Design and Implementation

The structure of the Biositemap Editor closely follows the recommendations found in online tutorials and some books about GWT [3]. The application has a simple client-server architecture, using the highly efficient and sophisticated GWT RPC protocol for communication.

Starting a GWT project is very simple. The GWT distribution comes with a couple of simple scripts that can be used to create a skeleton GWT project that has a client side, a server side. Google also provides an Eclipse plug-in[9] that speeds up considerably the GWT application development: The plug-in downloads the latest version of GWT, it creates skeleton project in Eclipse, and allows easy deployment of the application in a servlet container, such as Tomcat, or even directly in Google App Engine.

---

8 http://code.google.com/webtoolkit/
9 http://code.google.com/eclipse/

The Client-Server Communication using the RPC

When using GWT RPC protocol, the developer has to focus on two main issues: to define (i) the proper data structures describing the information sent back and forth between the client and the server, and (ii) the signature of the RPC calls that the client can make and the server has to serve.

We have defined a couple of data structures in the Biositemap Editor. For communicating information about the content of the a biositemap, we used a simple data structure that stores a property-values map, and a more complex data structure that represents the full description of a resource, aggregating all the necessary information, including multiple property-values data structures. Since the values of some of the properties need to be classes selected from the BRO (Biomedical Resource Ontology) we have defined a simple recursive data structure that is suitable of representing a class hierarchy tree. Since we have to deal also with file operations (open/create/save biositemaps or ontologies), we have defined a simple "file-handling" data structure. Finally, in order to make the configuration of the editor easily configurable, we have also defined a couple of data structures that enable us to describe the configuration of the editor including its entire form layout. The only requirement for defining data structures in GWT is to make the Java class representing that data structure implement the IsSerializable interface and ensure that all the class fields are Java serializable.

Defining the RPC calls in GWT is as simple as creating an interface that extends the RemoteService interface and adding the desired methods that take and return serializable classes (especially the ones that define our data structures), as well as the corresponding "Async" interface. The requirement on the method signatures in the "Async" interface are described in the various documentation sources. In the case of the Biositemap Editor we have declared the BiositemapService interface, which is implemented on the server side, and the corresponding BiositemapServiceAsync that declares the methods that can be called from the client. We declared methods for creating, opening, saving, closing a biositemaps, for getting BRO class hierarchy, for retrieving, updating, removing biositemap elements, etc.

In our experience with developing the Biositemaps Editor and also WebProtege, to enhance the performance of your application, it is best to optimize the information transferred between the server and client. For example, one simple rule is not to transfer to the client information that it does not need. We recommend optimizing the RPC data structures and leave out anything that is not absolutely needed by the client. Another simple rule is to try to minimize the calls to the server. For example, if your web client has to display a list or a tree of classes, you would need to send the class identifier (e.g., the class name), but also the browser text of that class. In this case, the data structure could include both the class identifier and the browser text, so that a second call to the server (to get the browser text) is avoided.

The Server

In the case of the Biositemap Editor, we followed the same design principle that we recommend to all the web applications that use Protégé to handle ontologies, namely that the server side executes all the ontology related operations and returns only relatively simple data structures that the web client can easily present to the user. Since the Biositemap Editor server can serve multiple clients in parallel, on the server side we manage a pool of Protégé OWLModel-s, one for each active client that has opened a biositemap. Once, on the client's request, the server has created or opened a biositemap successfully, it will generate a unique client ID used as a key in a map of OWLModels. The server then returns this ID to the client, and it will be used in any further communication between the client and the server related to that biositemap.

The server can also cache frequently asked information, or information that are common to all or multiple clients. For example in the case of the Biositemap Editor we cache the different class trees extracted from the BRO and we serve the same information to all the clients that ask for them.

In the case of the Biositemap Editor we have implemented an effective timeout mechanism that notifies inactive clients, and after auto-saving the biositemap in the process of editing frees up the JVM memory.

The Client

To develop the client side of your application, a developer has multiple options, ranging from using the GWT widgets to using 3rd party widget libraries, or even embed directly native JavaScript into the code. In our experience, the GWT widgets are very well supported, and provide consistent look-and-feel and behavior across different web browsers. However, the widgets provide basic functionality needed in a UI (text fields, combo-boxes, trees, grids, etc.), but more advanced capabilities (e.g., drag-n-drop) are not provided and other libraries have to be used.

In developing the Biositemap Editor and WebProtege, we have used the gwt-ext library[10] that provides rich widgets, such as grid with sort, customizable trees, drag-n-drop, etc. However, due to some licensing issues with the underlying JavaScript library (ext JS), the gwt-ext library might not be evolved in the future. A very promising replacement for it is the SmartGWT[11] library that provides not only a powerful widget library, but also ties in these widgets with your server-side for data management.

Discussion and Summary

In this paper we have presented our experience with the Google Web Toolkit to build a relatively simple web application that exposes the capability of Protégé to create valid biositemap RDF files, in an easy-to-use manner within a web browser. We called this application Biositemap Editor and is available at http://biositemaps.bioontology.org/editor/.

GWT had simplified our effort tremendously. It enables software developers without any knowledge of JavaScript to write both the server side and the client side of their web application in Java. GWT takes care of the subtleties of the client-server communication and it also compiles the client side to highly efficient JavaScript.

The presentation of the Biositemap Editor is easily configurable with a server side configuration file. Reusing most of the Biositemap Editor's code we have also created a Biositemap Browser that can be used to present in a read only mode the content of any biositemap RDF file, or the results returned by the official biositemap search tool available at: http://portal.ncibi.org/biositemaps/search/basic. The source code of the Biositemap Editor and Browser is open source and can be accessed at: https://bmir-gforge.stanford.edu/gf/project/biositemaps/scmsvn/?path=BiositemapEditor.

References

[1] O'Reilly, Tim: "What Is Web 2.0?" O'Reilly Network (September 30, 2005). Accessible online at: http://www.oreillynet.com/pub/a/oreilly/tim/news/2005/09/30/what-is-web-20.html (last retrieved in June 2009).

[2] Berners-Lee, Tim; Hendler, James and Lassila, Ora. "The Semantic Web". Scientific American Magazine (May 17, 2001). Accessible online at: http://www.scientificamerican.com/article.cfm?id=the-semantic-web (last retrieved on in June 2009).

[3] Hanson, Robert and Tacy, Adam. GWT in Action – Easy Ajax with the Google Web Toolkit. Manning Publications Company (2007).

---

[10] http://www.gwt-ext.com/

[11] http://code.google.com/p/smartgwt/