Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich
Institute of Information Systems
Prof. Moira C. Norrie
Matthias Geel
Michael Nebeling

Web Engineering
Spring Term 2012

Exercise 8

In the lecture, you have been introduced to web APIs, the REST-style architecture and also to several techniques that can be used to adapt web pages on the client side (case studies of crowdsourced adaptation). In this exercise, you are going to explore some of the technologies involved in order to develop a Firefox Add-On that not only performs client-side manipulation of web sites but also communicates with a REST-based web server to extend its functionality. You are asked to present your applications as well as to report on your experiences from developing it in the exercise session on **31 May 2012**. The exercise is divided into two parts and both parts need to be completed in order to gain the highest grade.

Part One

The goal of the first part is to familiarize yourself with the tools required to develop a Firefox Add-On. In addition, you are also asked to design and implement a very basic REST-based web service that your Firefox Add-On can use to store some data persistently on a server.

Exercise 8.1 *Getting Started*

Before you can start developing Firefox Add-Ons, you need to understand how the Firefox Add-On SDK works and what the fundamental abstractions are. The official Add-On SDK documentation can be found here: `https://addons.mozilla.org/en-US/developers/docs/sdk/latest/`

However, since the amount of information might be overwhelming, we advise you to read at least the following articles in the specified order:

1. Setting up the environment:
   `https://addons.mozilla.org/en-US/developers/docs/sdk/latest/dev-guide/tutorials/installation.html`
2. How to use the cfx tool:
   `https://addons.mozilla.org/en-US/developers/docs/sdk/latest/dev-guide/tutorials/getting-started-with-cfx.html`
3. Working with events:
   `https://addons.mozilla.org/en-US/developers/docs/sdk/latest/dev-guide/guides/events.html`
4. Two types of scripts:
   `https://addons.mozilla.org/en-US/developers/docs/sdk/latest/dev-guide/guides/two-types-of-scripts.html`

Note: Although the documentation states that you need Python 2.5 or 2.6, it also works with the latest Python version 2.7.3. On Microsoft Windows, please make sure that you include the Python root directory in the system environment variable PATH.

Exercise 8.2 *Create your first Firefox Add-On*

In order to get a grasp of the possibilites of a Firefox Add-On, we ask you to follow Mozilla's official Annotator tutorial which you can find here: `https://addons.mozilla.org/en-US/developers/docs/sdk/latest/dev-guide/tutorials/annotator/index.html`

Although its source code is part of the Add-On SDK distribution, we strongly recommend to go through the steps manually as you may need to use some of the techniques in part two for your own Add-On.

Exercise 8.3 *Extend the Annotator Example*

Now it is your turn. As stated in the tutorial, the Annotator example lacks the ability to delete annotations and has also some potential to be styled more appealing. In this step of the exercise, you are asked to extend the example Add-On with a functionality to delete all the annotations of the current web page and also to modify the appearance of the panels. These modifications can be minor but one should easily be able to tell the difference between the original styling and your version.

Exercise 8.4 *Create your first REST API with Spark*

Spark is a "micro web framework for quickly creating web applications in Java with minimal effort". If you already have a favorite web framework, you can skip this step and continue directly with exercise 8.5. Spark is extremely easy to start with. Go to `http://www.sparkjava.com/index.html` and download the Spark libraries. Make sure that you also download the dependencies. Create a new Java Project and add all the downloaded .jar files as depenendencies to the classpath. Follow the *Getting Started* example at `http://www.sparkjava.com/readme.html` to complete your first REST API!

Exercise 8.5 *Put the pieces together*

The final goal of part one is to substitute the local storage of the Annotator Add-On with a remote storage service accessed by a RESTful API. In order to achieve this goal, you need to do the following two sub-tasks:

- Create a REST-based web service that allows you to retrieve, add and delete annotations.
- Replace all references to the local storage in the Annotator Add-On with calls to the web service, using the *request* module provided by the Add-On API.

Important: You might want to use a database (e.g. MySQL or SQLite) on the server side to store the annotations persistently. However, for the purpose of this exercise, you can also store them in-memory if you prefer. Due to some limitations of the *request* module, you cannot perform *DELETE* HTTP requests. Even though it violates the REST principles, you can use a *POST* request to implement the delete operation in this exercise.

Part Two

Based on the lessons learned in the first part, you are now supposed to create your own Firefox Add-On that is based on the ideas presented in part one, but you should implement a different application.

Exercise 8.6 *Implement one of the Scenarios*

Similar to the RIA frameworks exercise, you may choose from two scenarios:

- Social Media Integration: The goal of this Add-On is to allow the user to share an item (i.e. image, text snippet, link) on any web site with either Twitter or Facebook. In addition, any comments made by the user's friends should be retrieved (and displayed) whenever the user visits the original web page again. Example: Bob shares a quote from a CNN article with his followers on Twitter. Several of his followers comment on that tweet. Whenever Bob visits that CNN article again, the Add-On should make those comments directly accessible at the appropriate position within the page.
- In-place manipulation: The goal of this Add-On is to perform some non-trivial computations on the server-side (as part of your web service) on items selected by the user and to replace those items with the result of the computation. In addition, your Add-On should be able to persist those replacements and to apply them again when the user retrieves the web page anew.
  Examples of non-trivial computations are:
  - Replace an image with a sharpened version or a version optimized for colorblind people (basically, any image operation on the server-side is acceptable)
  - Identify the names of places in a text snippet and link them to Google Maps
  - Beautify or perform syntax-highlighting for code-snippets

In both scenarios, you will need a RESTful web service to store some application data and to perform some of the computations on the server side. You may use any third-party library on the server-side to help with your computation/transformation or you may combine several third-party services to provide a unique transformation.

Workmode and Grading. Your solutions for both parts will need to be presented during the exercise session on **31 May 2012**. In the first part, you will be asked to clearly indicate which parts of the Annotator example you have modified to implement the delete functionality and the connection with the REST web service. In the second part, you are required to present a working application that fulfils the goals defined in either scenario. You do not need to hand in the source code, but the assistants may ask questions about how you have implemented some of the features and which Add-On SDK and Web APIs were used for your application.

For the grading we will use an A, B, C, D grading scheme similar to previous exercises:

A You have successfully completed part one **and** the final Firefox Add-On from Exercise 8.6 supports all the features described in the corresponding scenario.

B You have successfully completed part one **and** the final Firefox Add-On from Exercise 8.6 supports only parts of the features described in the corresponding scenario.

C You have successfully completed part one.

D Neither part one nor part two has been completed or you failed to present the solution in the exercise session.