Università degli Studi di Trento

Web Architectures

AngularJS front end on top of an EJB layer

Gianvito Taneburgo

December 31, 2016

1 Introduction

The following report describes the procedure that has been followed to develop a client-server application allowing users to keep an online notepad. By connecting to a web page, users can create, read, modify and delete notes in a handy way. The application architecture is similar to the one described in the 7th assignment, *Application front end on top of an EJB layer*: CRUD operations on notes are exposed by the back end via an EJB layer running on Wildfly and performed via JPA; an AngularJS front end interacts with a RESTful application running on Apache Tomcat connected to the EJB.
The Implementation paragraph contains details on the Apache Tomcat application, profoundly different from the one of the previous assignment. On the contrary, the Wildfly back end is almost unchanged and will not be extensively described.
The deployment procedure is completely similar to the one reported in the *Deployment* paragraph inside *Application front end on top of an EJB layer* and will not be copied here.
Java EE specifications have been modified over the years and the process to write and deploy client-server applications relying on EJBs has changed as well. For the remainder of this report the following versions of specifications, applications and frameworks will be considered:

- Java Platform, Standard Edition: *8*
- Java Platform, Enterprise Edition: *7*
- Enterprise JavaBean: *3.2*

- AngularJS: *1.2.32*
- Java Persistence API: *2.1*
- Wildfly: *10.1.0.Final*
- Apache Tomcat: *8.0.39*
- Hibernate: *5.2.3.Final*
- Apache Derby: *10.12.1.1*
- Apache Maven: *3.3.9*

Further details can be found inside Maven *pom.xml* files.

2 Implementation

The Tomcat and Wildfly applications share, as usual, some classes. For this application they are the EJB remote interface *NotepadBeanI* and the note transfer object *NoteTO*. The first one exposes CRUD operations that will be executed on the database, the second is solely used to exchange data between the two sides of the architecture. Further details about persistence via JPA and remote connection to EJBs are present in the *Application front end on top of an EJB layer* report.
The Wildfly application also contains a stateless EJB implementing the interface (*NotepadBean*), the entity model (*Note*) to be persisted and two utility classes to interact with the database and convert objects (*EntityDAO* and *TOFactory*).
The Tomcat application is made up of two parts, one related to the notepad front end developed in AngularJS and one related to the back end. The latter one, together with the classes previously described, contains a *RemoteProxy* responsible of connecting to the Wildfly EJB using the remote interface, and another class exposing REST-like operations for the front end (*NoteResource*). The REST layer is automatically generated with Jersey annotations on *NoteResource*'s methods, whose implementations only invoke the EJB operations.
The simple front end consists of HTML, CSS and JavaScript files. The result is a single page application in AngularJS, using *ngRoute* to switch between HTML views and JavaScript controllers. Besides AngularJS JavaScript libraries and a CSS file, in fact, the web folder contains the home page, the home controller and one HTML files and JavaScript controller for each view. The *ngRoute* module switches between views according to URLs: the final result is indistinguishable from a typical navigation across different pages. Each view controller provides the implementation of the functions invoked by the event triggered during the user interaction with the application. An AngularJS service wired to the application is globally available and performs HTTP requests to the RESTful layer described above. Response data are consumed by view-specific controllers and displayed to the user exploiting AngularJS two-way-binding.
The result is a simple yet useful application developed with few expressive lines of code involving different technologies, libraries and frameworks.