Installation and User Guide

June 9, 2014

1 Requirements

- Internet connection
- The following tools must be installed (lower versions may work, but not recommended)
  - *Java SDK* (>= 7)
  - *Git* - to get the code - http://git-scm.com/downloads
  - *Maven* (>= 3) - for building - http://maven.apache.org/download.html
  - *Apache Tomcat* (>= 8) - for deploying the web interface - http://tomcat.apache.org/

2 Get the code

Using Git clone the repository at GitHub using the following command:
git clone https://github.com/btodorov88/springbank.git

3 Build the project

In order to build the project the following steps have to be executed using *CMD* or alternative:

1. Navigate to the SpringBankWeb in the cloned repository
2. Execute the *mvn clean package* command

4 Deploy

The build procedure creates deployable *war* file. It can be found at *SpringBankWeb\target\springbank-1.0.0-BUILD-SNAPSHOT*. In order to deploy the file, one has to copy it to the *webapps* folder of the Tomcat web server (for easier access the file should be renamed to *springbank.war*). Once deployed the web client can be accessed at *localhost:8080\springbank* (default Tomcat configuration).

5 Explore the application

When the application is started it redirects to a login page for authentication (Figure 1) . The test data defines a user *btodorov* with password *123*. Once successfully authenticated the user is redirected to the main page which presents all accounts (left table) and transactions (right table) associated with the authenticated user (Figure 2).

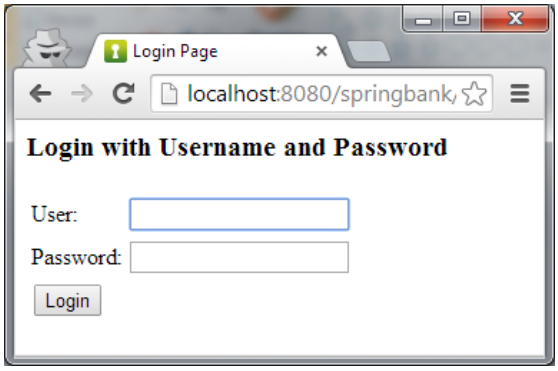

Figure 1: Initial login view

A new bank transaction can be made by clicking on the "Create transaction" button in the menu. Next the user is asked to fill in a form providing details about the transaction (Figure 3). When the form is submitted it is validated to make sure all necessary fields are populated and the "From account" has enough capital. In case of success the user is redirected to the main page where the newly created transaction can be seen.

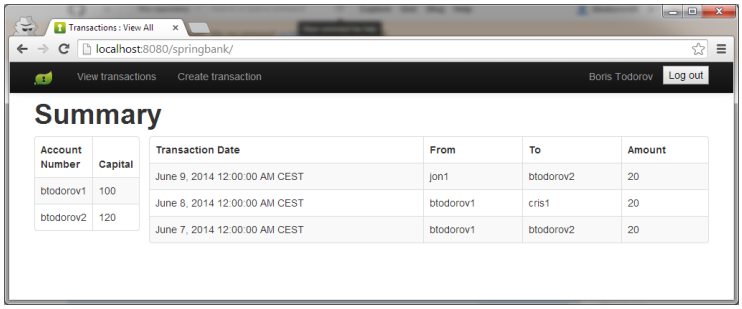

Figure 2: Main view which summarizes the accounts and transactions associated with the authenticate user

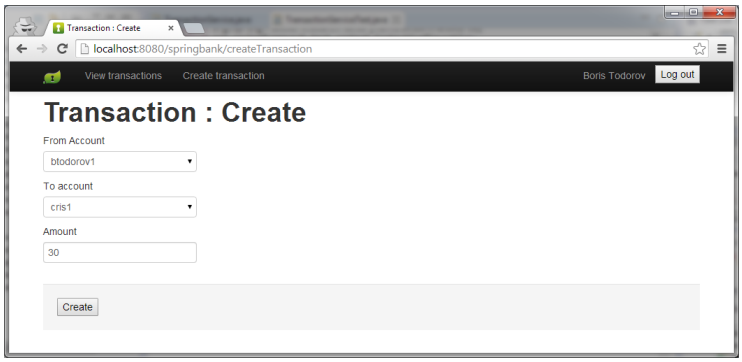

Figure 3: Component diagram illustrating the functional organization of the system

6 Take a look at the code

The cloned project can be easily imported into Eclipse. In order to do that go to *File\Import \Existing Projects into Workspace* and follow the instructions.

Next section briefly presents the internal organization of the system and identifies the high level components that it is built of.

6.1 Architecture

The architecture of the application is illustrated in Figure 4. It consists of the following components:

Figure 4: Component diagram illustrating the functional organization of the system

- *Database* - Responsible to store the accounts and transactions data. In the current implementation we use embedded H2 database for simplicity but it is just a matter of configuration to switch to another SQL database. On startup the database is populated with some test data.
- *Spring data repositories* - This component uses JPA2 and Hibernate to provide CRUD stile interaction with the database. The main application data objects (Person, Account, Transaction) are represented as JPA entities.
- *Business logic* - This component defines the Spring services which encapsulate the main business logic of the application.

- *Web UI* - This component is responsible to provide the Web interface of the application. It uses Spring MVC in combination with Thymeleaf and Apache Tiles to render the views presented to the users. The application also uses Spring Security to manage the access to its functionality. It uses the user information from the database (Person) for authentication.