Spring-Lab

Spring Lab example for introduction to DI, IoC, WebServices, Unit tests and mocks.

This example has been created for a Internet Application Programming (KIV/PIA) labs at the Department of Computer Science, University of West Bohemia in Pilsen, Czech Republic.

License

Project Structure

Application parts are wired together using Spring framework (see `applicationContext.xml` and `applicationContext-test.xml` in `main/resources` and `/test/resources` folders).

- `core.posts.domain` contains domain objects (inner representation of data
- `core.posts.manage` contains manager classes for the data (application logic)
- `core.posts.dao` contains client interface for data retrieval
- `core.posts.dao.jaxrs` contains Data Transfer Objects used for (De)serializaiton of the data retreived via web service. While in this example they have the same structure as domain objects, in the real world they can represent only a subset of a domain class or even have a completely different structure
- `webapp.components` contains custom wicket components - components should contain main part of display logic
- `webapp.pages` contains wicket pages - pages should be used for parameter parsing and basic layout (dont make large pages, decompose into components instead)

Tests

The example contains two implementations of the same test class. One uses own stub implementation of PostDao, the other uses Mockito framework to mock the PostDao instead.

Commands

- To run the project, exec `mvn jetty:run`
- To build the project and run tests, exec `mvn install`

WS API

Mock service and WS API description is located at Apiary.io

Apiary.io is a tool for web service API specification using Markdown syntax. Except for pretty documentation, you get a mock server responding with provided examples and client code generator in various programming languages.

Your API blueprint can be automatically linked to the project repository at GitHub.

Tasks

1. Get familiar with the project
   - Spring configuration, application context
   - Unit tests - stubs, mocks
   - Logger configuration - slf4j dependencies in pom.xml, log4j.properties, logger instance in PostDaoJAXRS
   - JAX-RS configuration - jaxrs and jackson dependencies in pom.xml, PostDaoJAXRS implementation
2. Try to start the application and go to `/posts` page.
   - If fails... :)
3. Define PostManager bean and try again.
4. Finish the functionality - comments are not displayed for their quotes at the moment.
   - Implement remaining methods of the provided PostDao interface in the PostDaoJAXRS class.
   - implement missing part of DefaultPostManager so that comments get joined to their posts.
5. Run tests to ensure your implementation is correct.
6. Ask questions