WebService Lab

Lab for WebService client and server demonstration.

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.

WebServices

- Check core.posts.dao.jaxrs package for client-side implementation.
- Check webapp.service package for server-side implementation

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. Run the application and check main page.

Client-side

First we finish the client-side of our application. The PostDaoJAXRS is not providing us with comments, we need to fix that.

1. Initially we need to define class which will represent the data we receive from the web-service. Take a look at the examples of **post** resource in the documentation.
2. Now take a look at the clas *dao.jaxrs.PostDTO*. See the attribute names and keys in **@JsonProperty** annotations? How are they connected to the documentation from the previous step?
3. Now take a look at the documentation of **comments** resource and modify *dao.jaxrs.CommentDTO* class accordingly.

Now we have prepared data model for consuming **comments** from the web service.

1. Take a look into the implementation of the `findAllPosts` method. The `findAllComments` will look very similar, the only differences will be **path** pointing to the *comments* resource and the fact that we will be working with **CommentDTO** class and **Comment** entity instead of **PostDTO** and **Post**.
2. Implement the `findAllComments` method in the same fashion the `findAllPosts` method is implemented.
3. Re-run the application and check comments are displayed now.

Server-side

The package *webapp.service* contains two implementations of the web-service: JAX-RS (Java standard for RESTful web services) and Spring MVC (Spring Framework module for RESTful web services). Both currently return list of posts:

- JAX-RS
- Spring

Implement the following resource in both: **/comments**. The examples contain all the annotations you need:

- **JAX-RS**: @Path, @GET
- **Spring**: @RequestMapping

You will need to implement own **CommentDTO** class for serialization.

You can test your implementations at the following URLs:

- JAX-RS
- Spring

Additional Notes

- This example takes advantage of DI: We have 3 instances of PostManager and PostDao, each displaying data from different source.
- UI is written in the Apache Wicket framework. Nice component-based framework for writing UIs.

License