h_da
HOCHSCHULE DARMSTADT
UNIVERSITY OF APPLIED SCIENCES

Darmstadt University of Applied Sciences: Java Lab

Java Lab 2008/11/03
Use cases

Prof. Dr. Bernhard Humm

Version: 1.0
Date: 2007/10/08

Darmstadt University of Applied Sciences
Department of Computer Science
Schöfferstr. 8b
64295 Darmstadt
Germany

1 Preparation

1. Open my homepage www.fbi.h-da.de/~b.humm and all relevant documents

2. Update the project from the Subversion server

In this lab session, you will implement the uses cases of the library system. I have already implemented the use case interfaces with their specifications in Javadoc. Organize in your team as to who is implementing which use case classes.

2 Use case interfaces

Prepare your use case interface for the application server:

1. Annotate the use case interface (e.g., `de.h-da.library.datamanagement.usecase.Search`) with `@Local` (from package `javax.ejb`). This will tell the application server that the use case can be found and accessed locally.

2. Add an Interface `<Use case>Remote`, e.g., `de.h-da.library.datamanagement.usecase.SearchRemote` and provide the same methods as the original use case interface (Tip: use *Copy / Paste* of classes). Annotate it with `@Remote` (from package `javax.ejb`). This will tell the application server that the use case can be found and accessed remotely.

3 Use case classes

Implement the use case classes:

1. Implement the use case class in package `de.h-da.library.<component>.usecase.impl`, e.g., `de.h-da.library.datamanagement.usecase.impl.SearchImpl`. The class must implement the use case interface and the remote interface, e.g., `Search` and `SearchRemote` (from package `de.h-da.library.datamanagement.usecase)`.

2. Annotate the use case class with `@Stateless` (from package `javax.ejb`). This will tell the application server to treat the use case as a stateless session bean. Provide a mapped name for JNDI (y group only).

3. For all entity managers and other use cases you need for the implementation of your use case: implement a private instance variable (typed by local interface) and annotate it with `@EJB` (from package `javax.ejb`), e.g., `@EJB BookManager bookManager;`

This will implicitly invoke the name service and provide you with an instance of the implementing class, e.g., `BookManagerImpl`.

4. Implement all use case methods as specified in the use case interface (Javadoc).

5. Tip: See `de.h-da.library.usecase.impl.UseCase1` for an example.

4 Test

Implement a JUnit Test case `<UseCase>Test`:

1. Create the JUnit class.
   Tip: use the *New / File, Folder / JUnit / Test for existing class* wizard..

2. Implement the `setUp` method to instantiate proxies via name server lookup.

3. Define a private instance variable to store a use case object. Type it with the remote use case interface, e.g., `SearchRemote`

4. Implement all test methods.

5. Deploy your application (y group only) and run the tests.

6. Tip: See `de.h_da.library.component1.usecase.impl.UseCase1ImplTest` for an example.

5 Optional Tasks

Extend the uses cases with more sophisticated functionality.