Fifth assignment Enterprise Java Beans 3.0 principles

Send the finished task by e-mail to your teaching assistant. Deadlines for the submission are on the web page of the course: http://d3s.mff.cuni.cz/teaching/middleware/.

The assignment uses a simple algorithm to measure distance between two nodes in a graph of objects, to mimic working with entities with relationships. Understanding the assignment requires no special knowledge.

Prerequisites

As a technology for client/server architecture and data persistence, the assignment uses the Enterprise Java Beans standard, version 3.0. OpenEJB version 4.7.4 was chosen as a concrete implementation of the standard. The following knowledge is needed for the implementation:

- Defining remote business interfaces (annotating the interface with the `javax.ejb.Remote` annotation).
- Implementing remotely accessible Stateless Session Bean (annotating the class with `javax.ejb.Stateless`, implementing remote business interface).
- Implementing Entity Beans with persistent attributes (annotating the class with `javax.persistence.Entity`) and specifying the mapping to a relational database via `persistence.xml`.
- Managing persistent Entity Beans via EntityManager injected using the `@PersistenceContect` annotation into the Stateless Session Bean.
- Defining relations between Entity Bean classes (annotations `@OneToMany`, `@ManyToMany` etc.).
- Lanuching an OpenEJB server, deployment of EJB Beans. Connecting client and server via Naming Service and RMI.

A simple example of remote use of the EJB Beans variants will be provided.

Assignment details

http://d3s.mff.cuni.cz/teaching/middleware/files/as5.zip

The assignment consists of a simple algorithm to measure distance between two nodes in a graph. Graph nodes are represented by objects of the class `Node`.

```
public Node
{
    private int id;
    public Collection<Node> getNeighbors ();
    public void addNeighbor (Node oNeighbor);
}
```

The `id` attribute is an unique number identifying the node. The `addNeighbor()` method adds a node to the set of neighbors of the node and is used when a graph is being created. The `getNeighbors()` method returns a set of all neighbors of the node and is used when measuring the distance. The measuring itself is provided by the `Searcher` interface. The interface also provides methods for creating and connecting nodes, so that the client does not work directly with the `Node` objects. The `addNode()` method thus returns the id of the newly created node, the `getDistance()` and `connectNodes()` methods expect the id's of nodes to work with.

```
public interface Searcher
{
    public static final int DISTANCE_INFINITE = -1;
    public int getDistance (int nodeFrom, int nodeTo);
    public int addNode();
    public void connectNodes(int nodeFrom, int nodeTo);
```

```
}
```

Your tasks are:

1. Examine the provided local implementation of the assignment.
2. Implement the `Searcher` as a Stateless Remote Bean and the `Node` as an Entity Bean with container-managed persistence, and the `id` attribute as a primary key. Represent the set of neighbors as a suitable relationship between entities.

   Modify the client to create nodes and measure distances using a `Searcher` deployed on the EJB server.
3. Verify that the `Node` entities are persistent.

   Add a command-line option to the client that will cause it to just run searches on the persisted graph without creating new nodes.
4. Extend the Searcher interface and change the implementation to offer working with persistent nodes to multiple clients simultaneously, so that each client has his own graph.

   The id of the client will be provided as a command line argument.

   Consider what is the best way of keeping track of client id (clients should not send their id on each request to the searcher) and how to change the attributes of the Node objects to allow storing independent graphs of multiple clients.