Middleware Labs: OpenEJB

Petr Tůma Vojtěch Horký Antonín Steinhauser
Vladimír Matěna

May 15, 2018

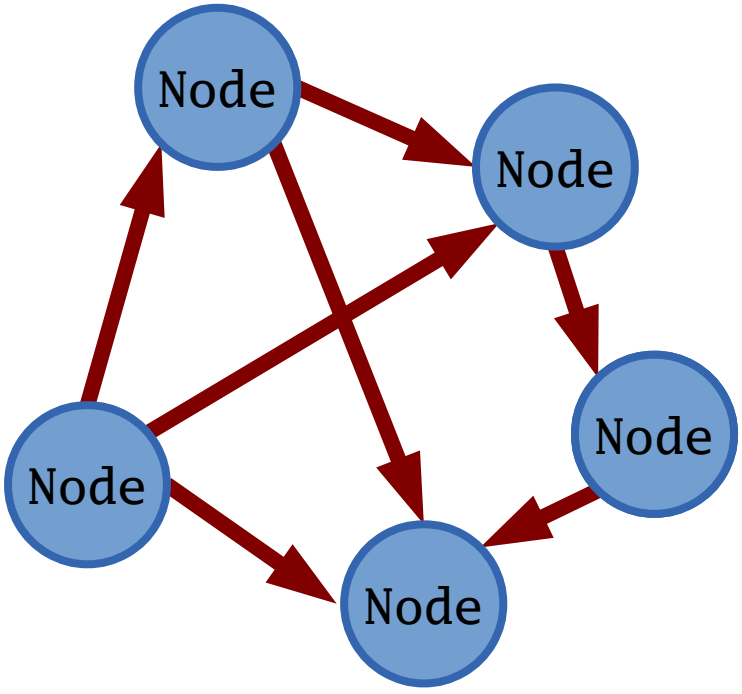
Enterprise JavaBeans 3.0

- `http://d3s.mff.cuni.cz/teaching/middleware/files/OpenEJB-4.7.4.tar.gz`
  - Uncompress into `~/OpenEJB`
- `http://d3s.mff.cuni.cz/teaching/middleware/files/as5.zip`
  - `Example/` - EJB demo, including `README` and few helper scripts
  - `Input-Files/` - local implementation of the task

Notes

- EJB server must be running(`run-server`)
  - Different port when already used
    - Server: `~/OpenEJB/conf/ejbd.properties`
      `port = XYZ` (XYZ>1024)
    - Similarly `admin.properties` (for `stop-server`)
    - Client:
      `props.put(Context.PROVIDER_URL, "ejbd://127.0.0.1:XYZ");`
- Server part deployed with `run-deploy`
  - Needed after each re-compilation!
  - Persistent data stored in `~/OpenEJB/data`
    - In case of rather bigger changes it is safer to `stop-server` and delete the (`hsqldb`) data

Task - compute distances in the graph




```
public interface Searcher {
  public int addNode();
  public void connectNodes
    (int nFrom, int nTo);
  public int getDistance
    (int nFrom, int nTo);
}
```

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

1. Local implementation

- Class `Node`
- Interface `Searcher` and class `SearcherImpl`
- Launchable class `Main` (`java Main`)
- Measure the speed on the random graph

2. Searcher as a stateless session bean

- Searcher as a remote business interface
  - Use appropriate annotation
- Annotation of class `SearcherImpl`
- Compilation and deployment
  - See scripts in `Example`
  - Output also contains JNDI name of the bean - `Jndi(name=<ClassName>Remote)`
- Client - class `Main`
  - JNDI context creation – see `ExampleClient`
  - `Searcher` instance retrieved by JNDI `lookup`

3. Node as an entity bean

- See `Movie` and `Director` in `Example`
- Annotation of class `Node`
- Getter/setter for `id` with appropriate annotation
- Neighbour nodes as relations among entities
  - Getter/setter with appropriate annotation of the relation

4. Persistence of Node objects

- Update the class `SearcherImpl`
  - See `ExampleEntityBeans`
  - Replace hashmap `nodeMap` with EJB equivalents
- Annotated `EntityManager`
  - `unitName` - corresponds to persistence.xml
  - Method `persist()` for persistence of created `Node`
  - Method `find()` for finding `Node` by `id`
- The deployed JAR must contain file `META-INF/persistence.xml` - see `Example`
  - Set persistence-unit name and class correctly

5. Verify persistence

- Stop the server after creating the graph, start it before searching through it
- Where to get the node `id` for the second launch?
  - Try not to assume anything about automatic `id` assignment to `Node`
  - Remember which `id` was returned during creation
  - Optimal: `add` method to `Searcher` that selects a random `id` from existing nodes

6. Multiple graphs

- Clients with different client id operate on separate graphs
- Change the definition of SearcherImpl to keep track of the client id
- Do not pass the client id as an argument to every method

Implementation

- Reuse available code
  - Algorithm implementation of the local variant
  - Scripts and code from the example
    - Do not add packages etc.
- Use Eclipse, NetBeans etc., if you like
- Report issues, ask questions when unclear
  - Mailing list...

Submission

- Part of the solution is also documentation of the chosen approach
  - See point 4 in the task description, where you can choose among different approaches
- By e-mail (deadline is on the web)
- Make sure it works in the lab downstairs
- The submission shall be easy to start