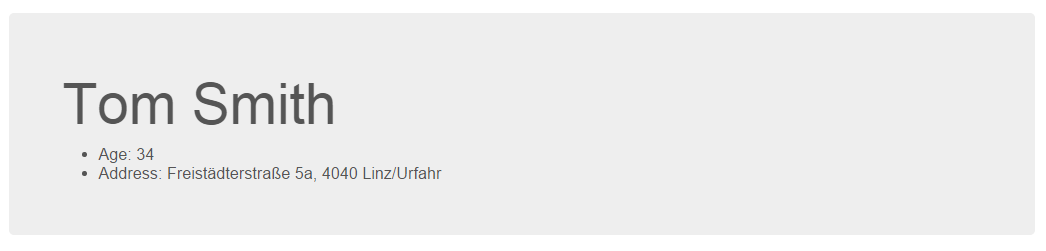
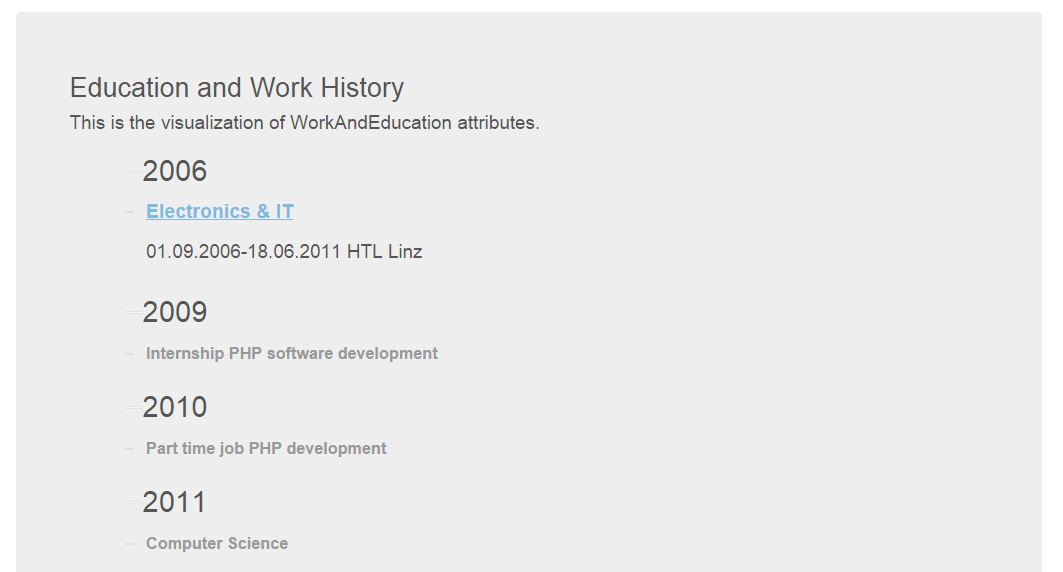
# User documentation

## User profile

On this site information about one special user (selected in the user overview accessed by the menu item *Users*) is displayed. The first things to see, are the name, the age and the location of the user. At this stage of development this is not done dynamically. Since this fields are mandatory when creating a new user this is no problem for the given data, but if there are more attributes given for one user, they are not displayed. This could be a realized by displaying all PersonalAttributes (see model for description). ****

### Timeline

The first graph is the *timeline*. When a user is created WorkAndEducationAttributes can be added. Every WorkAndEducationAttribute represents a occupation/internship/etc. or education (school/university) and consists of a title, a description, a start and end date and a organization (school, university, company, ...). All WorkAndEducationAttributes are then associated with the year they started (the end date is then shown within the item itself) and displayed as follows:



The timeline is interactive and can be closed and elapsed.

### Tag / Bubble / Zoom cloud

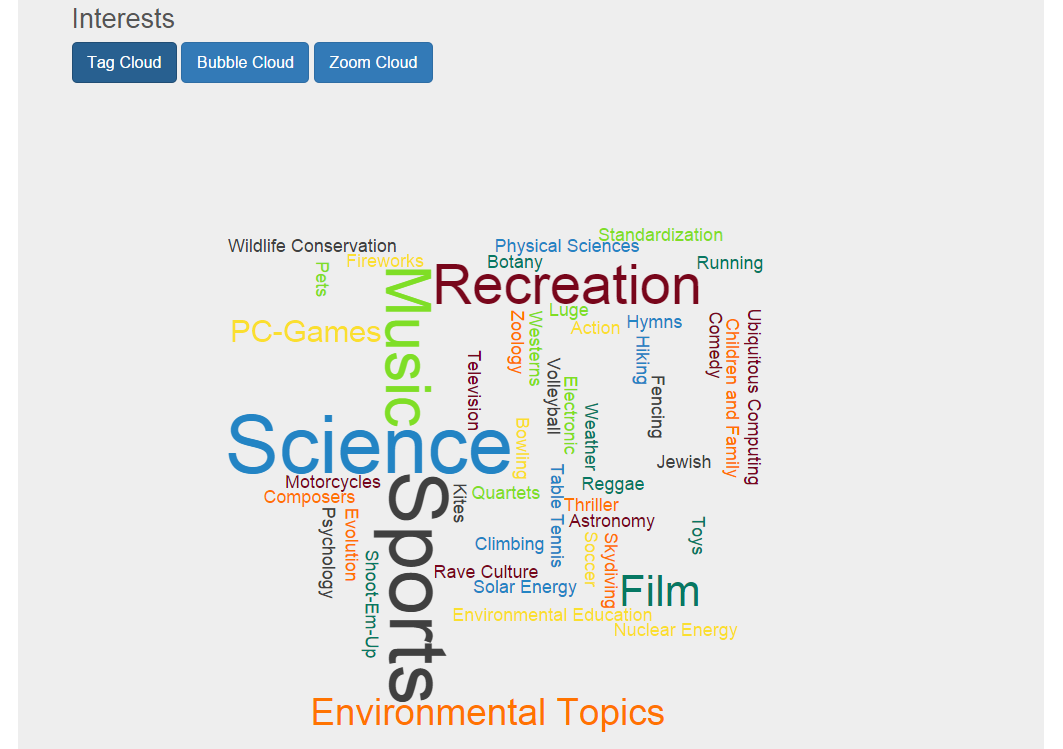
Since a user can have CompetenceAndInterestAttributes (for ease of reading from now on just called competencies) which are interests, non interests and competencies these are the next parts of the profile. Each of this attribute can be displayed in three different ways. First there is the *tag cloud*, then the *bubble cloud* and third a interactive bubble cloud, from now on called *zoom cloud*. The tag cloud and the zoom cloud are the same for all three types of attributes, the bubble cloud has an additional time slider for CompetenceAttributes to visualize the evolution of competencies. There is a collection of defined colors used, if more colors than the pre defined ones are needed random colors are generated.

Due to the fact that sub competencies (competence, level, example, ..) can be the same (see languages in screenshot 3.1) and their title can get really long and therefore not very speaking, the top level (language itself, category of interest/non interest) are shown in all graphs if one sub competence is reached. This does not imply that the top level competence is acquired, but is necessary for providing a pleasant graphical representation of competencies as there would be no meaning behind a single displayed sub competence. This is only applied for the top level. All other competencies are shown if one of the following rules apply:

* user has acquired competence itself
* competence itself has several levels and the user has gained one of them
* competence is a level of a parent competence and user has already gained the parent competence
* competence is a level of a parent competence and user may has gained higher level

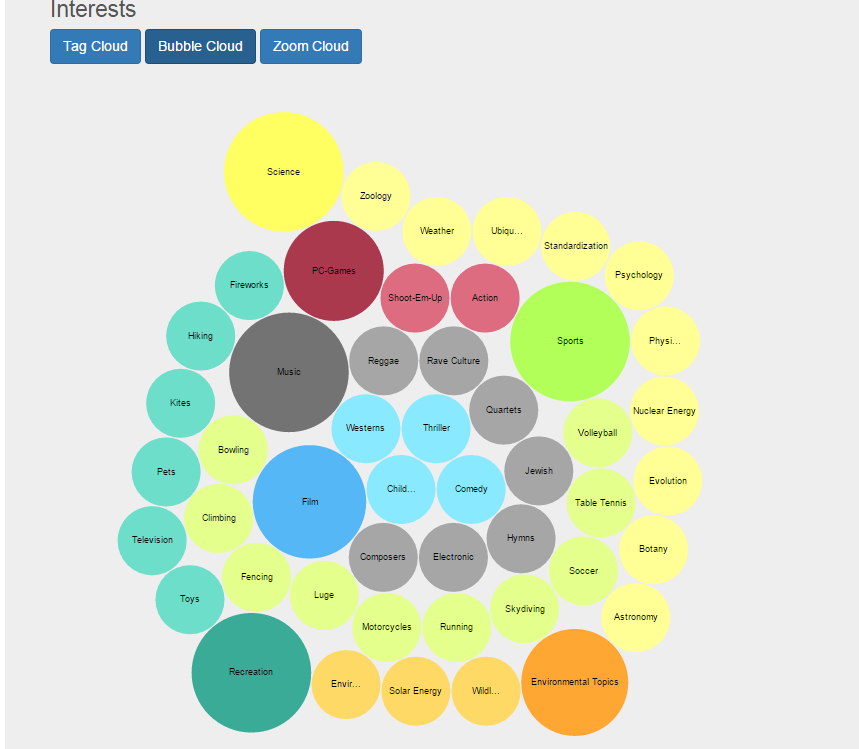
#### Tag Cloud

All gained competencies (described in 1.a.ii) are displayed in the tag cloud. Font size depends on level (parent competence is always bigger than sub competence) and gained sub competencies (font is bigger the more sub competencies are gained). The colors are selected random from a given array of colors.

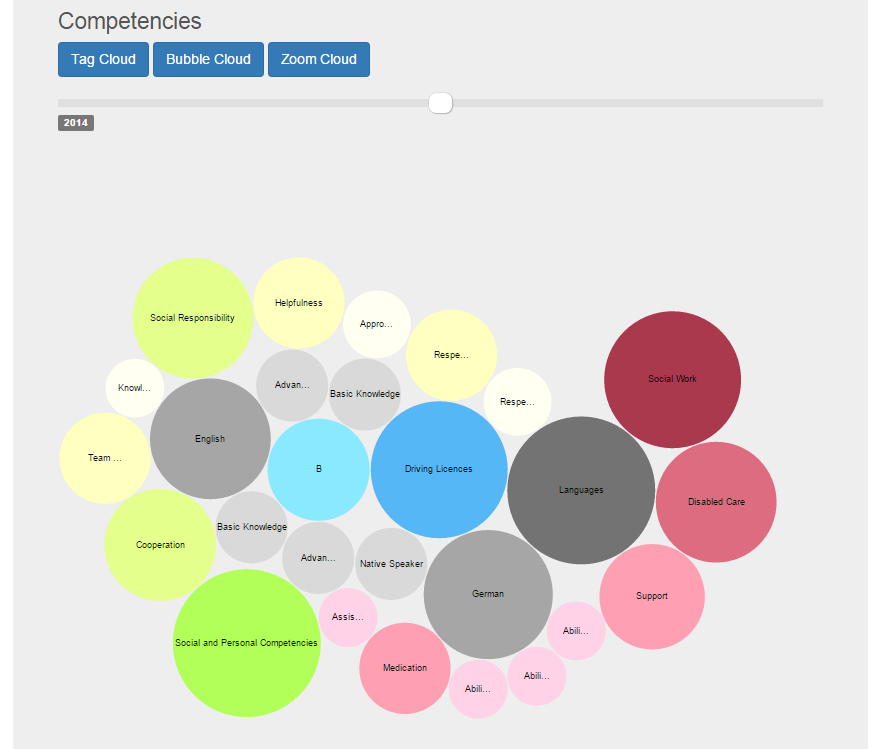
******

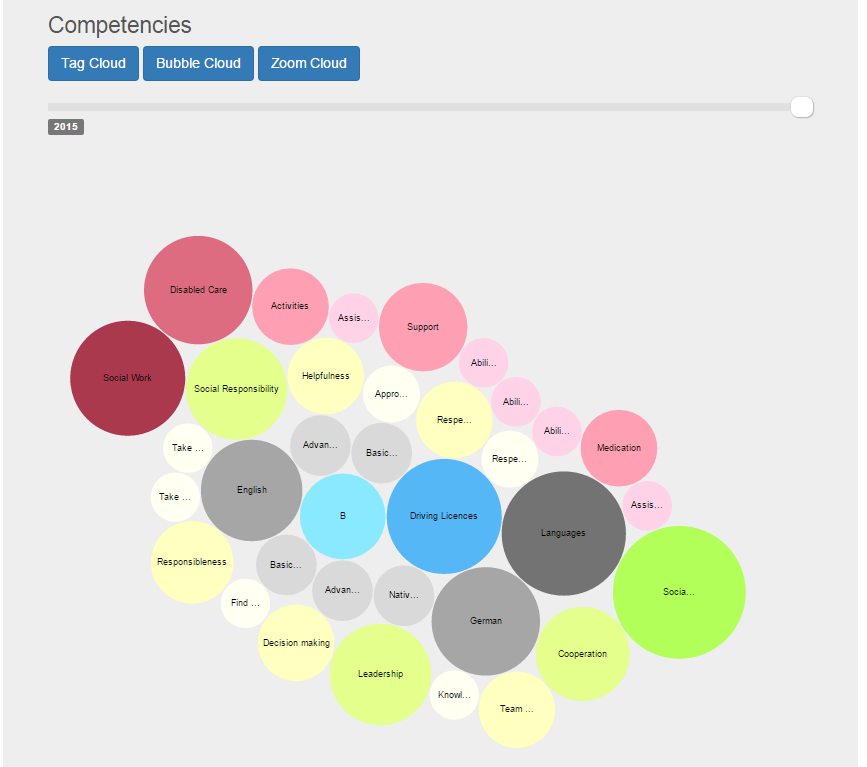
#### Bubble Cloud

All gained competencies (described in 1.a.ii)are shown in a bubble cloud. The first top level is displayed in the center and all sub competencies are drawn concentric. After the sub competencies of the first top level the next top level is shown and so on. The colors of parent and sub competencies are the same, but the deeper the level the brighter the color. So it can be seen at first sight which sub competencies belong to which parent competencies and vice versa. If the name is too long to fit in the bubble a short name is used (to read the whole title move the mouse over the bubble and a hover text containing the full title is shown). Currently this are just the first letters of the title but if the inLOC input would be extended by a short name it could be displayed.

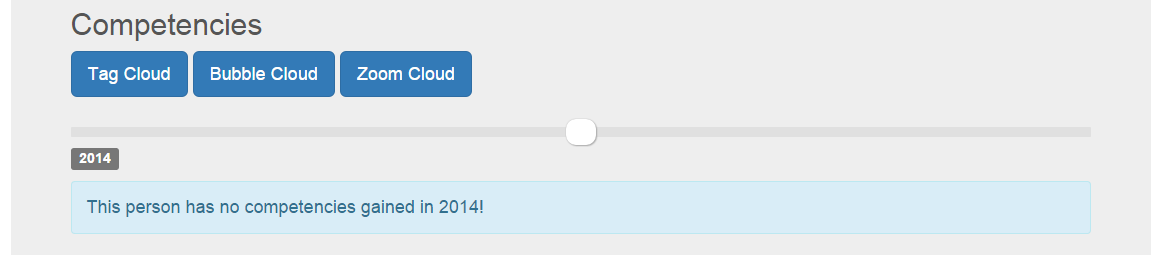


Evolution of CompetenceAttribute:



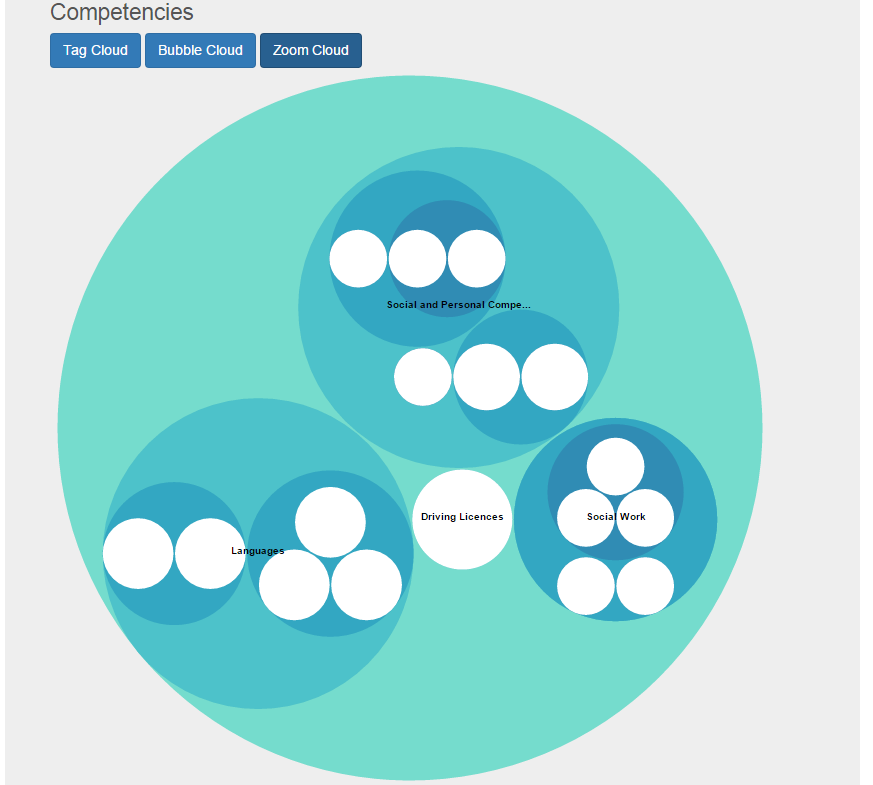


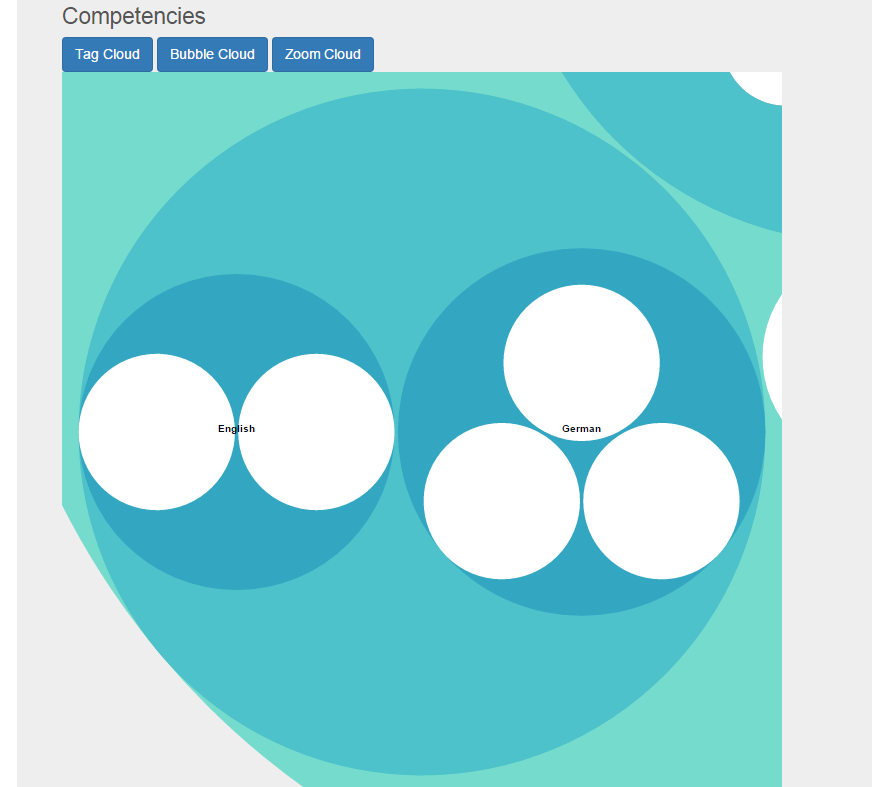
As this is just a prototype the time entries for the slider (2013/2014/2015) are hard coded at the moment. Another improvement would be to consider the expiration date of the competencies as well, currently there is only the date when the competency is gained considered for displaying. Nevertheless these would not require major changes in the code.

If there are no competencies acquired in a special time slot a notification is shown. ******

#### Zoom Cloud

All gained competencies (described in 1.a.ii)are shown in a interactive bubble cloud. There is one bubble for every top level and in every bubble all acquired sub competencies are shown. By clicking on the bubble the graph can be zoomed in and out (by clicking on a outer bubble). Again there are short names displayed and a hover text containing the full title is provided.



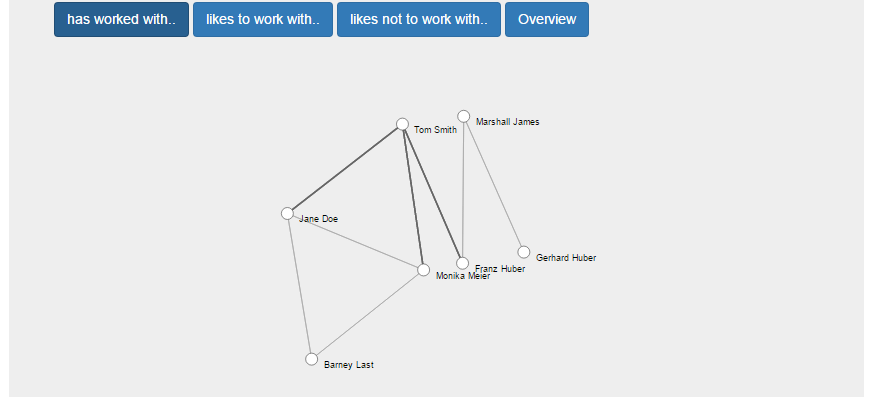


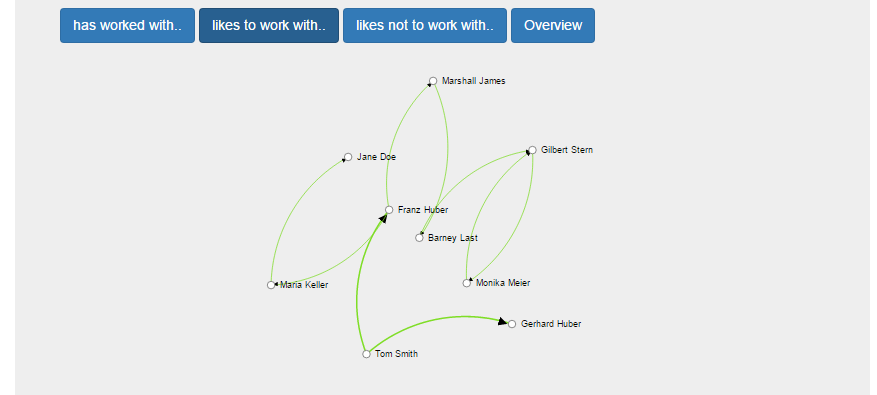


### Connection networks

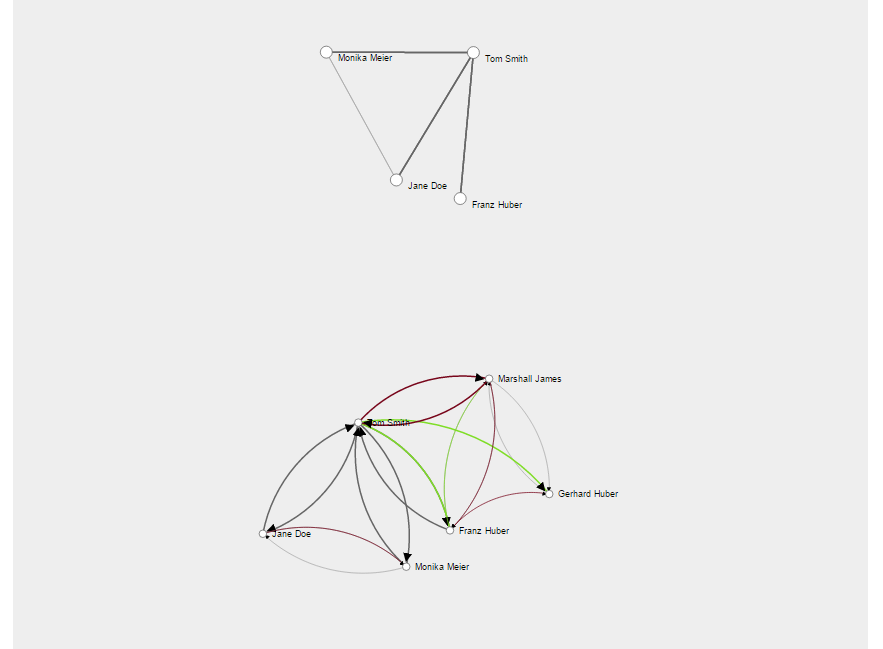
A user can have different connection to other users. First there is the simple "has worked with"-connection indicating with which users the user has already worked in the past. Then there are the "likes to work with"-connection and "likes not to work with"-connection representing users the user likes or does not like to work with. There are four graphs to display the connections, one for every type of connection and one overview.

The "has worked with"-graph is bidirectional and all the other are unidirectional.





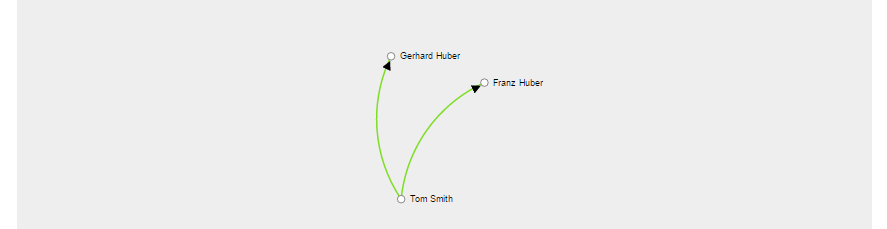
If there is more than one type of connection between to users ("has worked with" and "likes to work with" f.e.) the "likes to work with"-connection is the one that is shown in the overview. Therefore the "has worked with"-connections are shown as directed line in the overview as otherwise information could be lost in the overview.



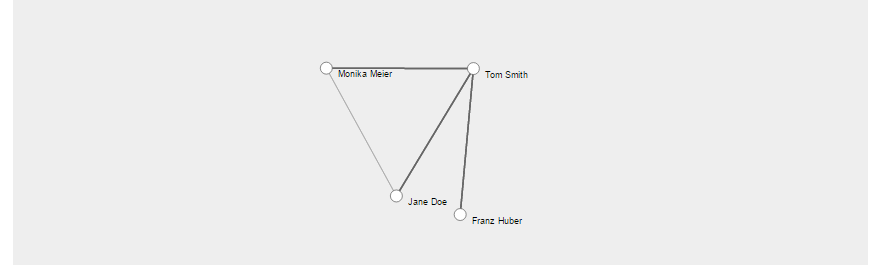
See the example above: Tom Smith and Franz Huber have already worked together, therefore there is one connection in the "has worked with"-graph. But in addition Tom Smith likes to work with Franz Huber. Due to this you can see the "likes to work with" connection between Tom Smith and Franz Huber as well as the "has worked with" connection from Franz Huber to Tom Smith.

In case there is a "likes to work with" and a "likes not to work with"-connection between two users in the SAME direction (would be senseless to like and not to like to work with another person) the "likes not to work with"-connection is displayed.

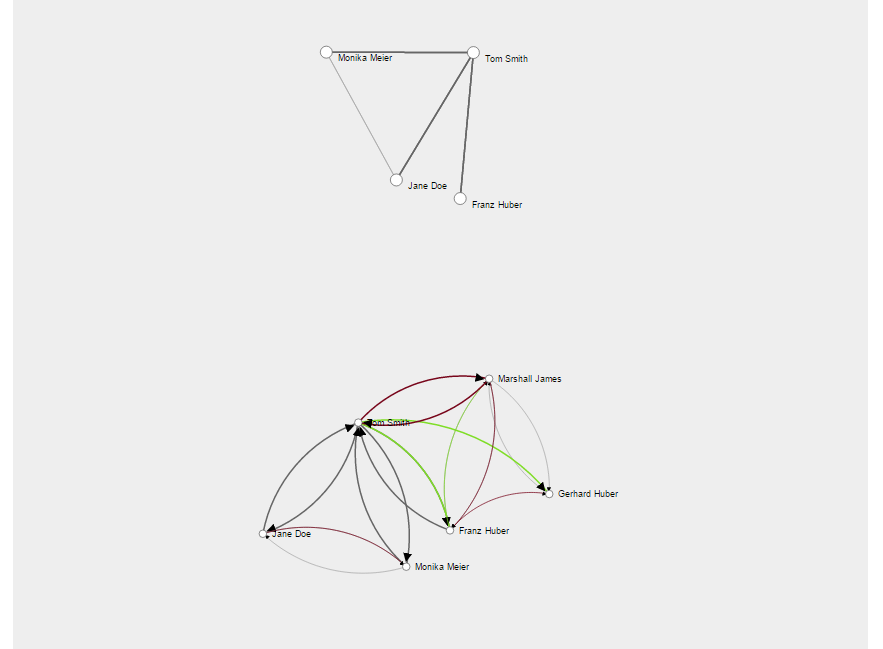
The networks can be displayed with different depth of recursion from now on called level. This level can be adjusted by a slider. Level 1 shows the user itself with his direct connections.



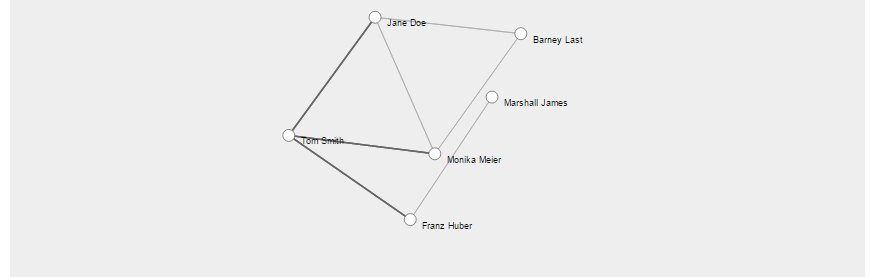
Since the whole available information should be displayed in the graph also connection within the displayed connection are shown. This is only a minor modification of the code, but this was the intentional way to do it. If two users are shown in the graph their connection should also be displayed (even this is not really information related to the user). In the screenshot below you can see an example:

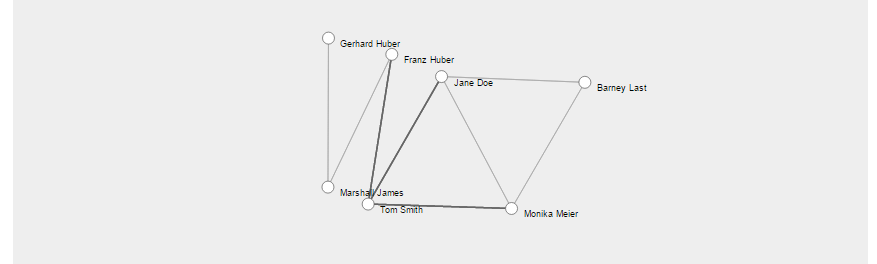


The overview is in level 1 already quite informative, as all available connection to the direct connected users of all three types of connections as well as the connections amongst these direct connected users are shown.



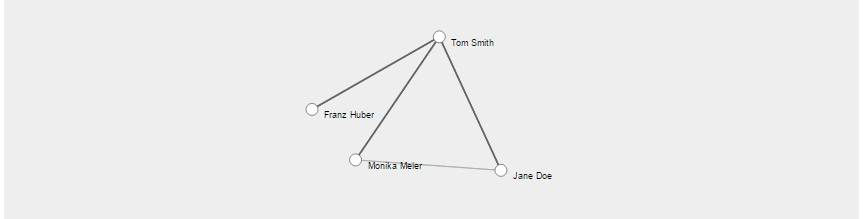
When increasing the level the direct connection from the outer most users are added to the graphs. In the example below you can see level 2 and level 3:



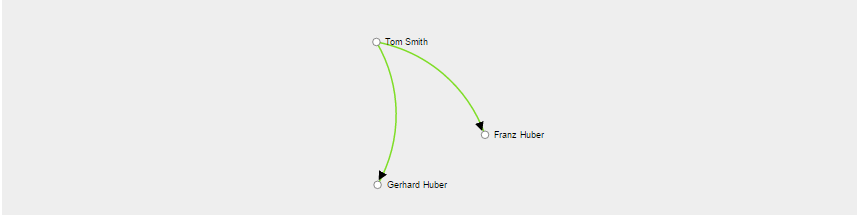


At the highest level all connection in the context of the current user are shown. The overview is the most informative graph, there can be more connections than in the other graphs. The following example shows why this can happen (and is mostly the case).

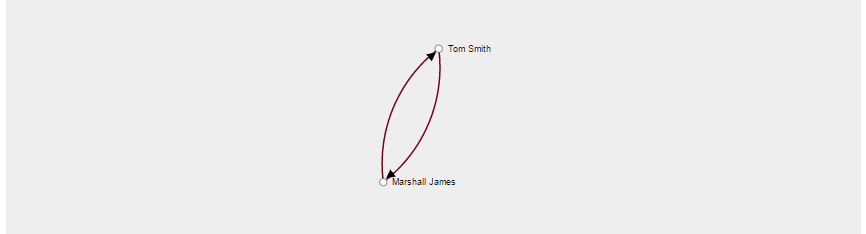
"has worked with":

****

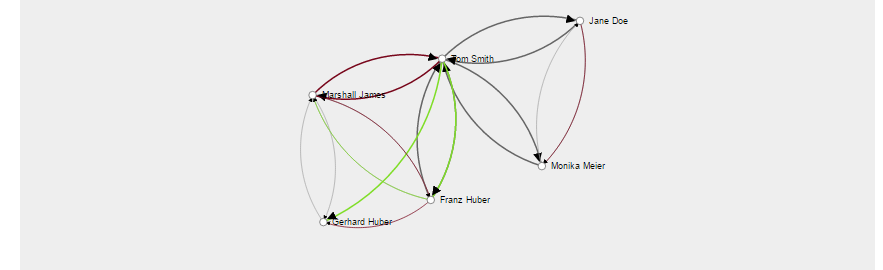
"likes to work with":



"likes not to work with":



"overview":



Example: Franz Huber is contained in level 1 in the "has worked with" graph and Marshall James is contained in level 1 in the "likes not to work with" graph. In the overview you can see also the connections between those two users.

## Task profile

A task is represented by a title, a description and a location. As well as in the user profile this is not displayed dynamically at the moment. A task can require competencies or interests which are displayed like in the user profile at the moment. In future work this has to be adapted. A task can have special constructs of required competencies (e.g.: Competence A OR (Competence B AND Competence C) are required for this task) since this is not in the scope of this work neither in the model nor in the implementation the task is visualized similar to the user profile and the required competencies and interests are just seen as a set. To read detailed explanation to the different graphs see 1.a.ii.

# Installation instruction

For the test installation on localhost a MySQL database (XAMPP), JBoss, Hibernate and for development Eclipse Java EE IDE were used . In the following section the installation process is explained in detail.

## Download

XAMPP (Version 1.8.2 used) - https://www.apachefriends.org/de/download.html

JBoss (Version 7.1.1.Final used) - http://jbossas.jboss.org/downloads/  
For the test environment the downloaded file was unzipped and moved to *C://ProgramData/JBOSS/*. Make sure that there are no blanks in the path to the server, because this caused a problem while starting the server. Please note that JBoss 7.1.1.Final is not compatible with Java 1.8!

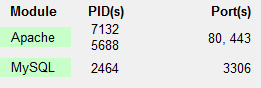
MySQL-JAVA connector (Version 5.1.36 used) - http://dev.mysql.com/downloads/file.php?id=457912

Eclipse Eclipse Java EE IDE for Web Developers (Version Indigo Service Release 2) - http://www.eclipse.org/downloads/packages/eclipse-ide-java-ee-developers/indigosr2  
Make sure that used Java version is NOT 1.8 since the JBoss server will NOT be working on Java 1.8.

Additional used libraries are contained in *CrAc/WebContent/WEB-INF/lib*.

## MySQL

Install XAMPP and start the control panel. Find the used ports in the picture bellow.



Open *http://localhost/phpMyAdmin*. Create a new database and make sure that collation is set to *utf8\_general\_ci*. The URL to the database is needed for configuring the JBoss server.

## JBOSS

The following instructions are valid for MySQL database and has to be adjusted if any other database is chosen but this is not in the scope of this work.

Create the directory *JBOSS/jboss-as-7.1.1.Final/modules/com/mysql/main/* if not existing and add the following files:

* module.xml   
  <?xml version="1.0" encoding="UTF-8"?>  
   <module xmlns="urn:jboss:module:1.0" name="com.mysql">  
   <resources>  
   <resource-root path="mysql-connector-java-5.1.XX-bin.jar"/>  
   </resources>  
   <dependencies>  
   <module name="javax.api"/>  
   </dependencies>  
   </module>
* mysql-connector-java-5.1.XX-bin.jar (unzip MySQL-Java-Connector, it's enough to move the executable jar-file to the main folder)

The file *JBOSS/jboss-as-7.1.1.Final/standalone/configuration/standalone.xml* has to be modified. Detailed information can be found in 2.a.i.

### standalone.xml

In section <subsystem xmlns="urn:jboss:domain:logging:1.1"></subsystem> add:

<logger category="org.jboss.as.jpa">  
 <level name="WARN"/>  
</logger>  
<logger category="org.hibernate.SQL">  
 <level name="WARN"/>  
</logger>  
<logger category="org.hibernate">  
 <level name="WARN"/>  
</logger>  
<logger category="org.infinispan">  
 <level name="WARN"/>  
</logger>

If further information about the Hibernate and SQL actions are wanted in the server log file change WARN to TRACE. But this cause **major performance slow down** when starting the server.

In section   
<subsystem xmlns="urn:jboss:domain:datasources:1.0">  
 <datasources></datasources>  
</subsystem> add:

<datasource jndi-name="java:/MySQLDS" pool-name="MySQLDS" enabled="true" use-java-context="true">  
 <connection-url>jdbc:mysql://localhost:3306/databasename</connection-url>  
 <driver>mysqlDriver</driver>  
 <security>  
 <user-name>username</user-name>  
 <password>pw</password>  
 </security>  
</datasource>

JDNI-Name (and pool-name) are important, because they are used later for configuring the hibernate.xml, should not be changed to ease installation.

As mentioned above the used test environment was: MySQL database (databasename was *db\_bakk\_ee*) running on localhost port 3306 (XAMPP), with no user configuration. Therefore the used username was *root* and the section <password></password> has been deleted.

In section   
<subsystem xmlns="urn:jboss:domain:datasources:1.0">  
 <drivers></drivers>  
</subsystem> add:

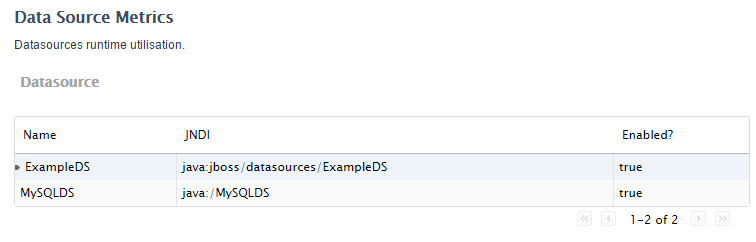
<driver name="mysqlDriver" module="com.mysql">  
 <driver-class>com.mysql.jdbc.Driver</driver-class>  
 <xa-datasource-class>com.mysql.jdbc.Driver</xa-datasource-class>  
</driver>

Since the section <datasource><driver></driver></datasource> refers to this driver, the name should either not be changed, or changed in both sections.

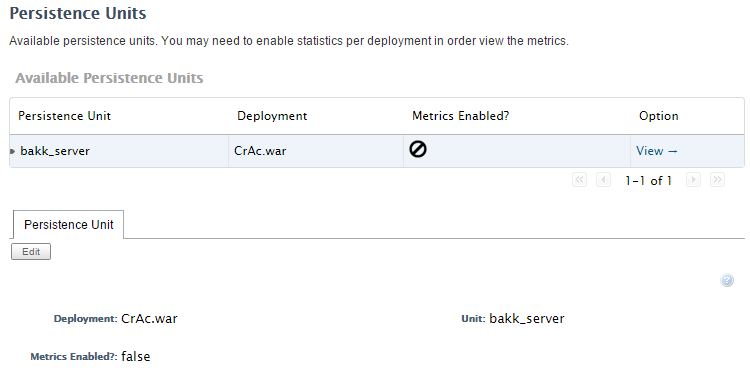
### JBoss administration dashboard

Once the server is started (explanation in 2.d), a administration dashboard can be opened in browser (*http://127.0.0.1:9990*). To access the console a user must be added. Run *JBOSS/jboss-as-7.1.1.Final/bin/add-user* to create a new user. Add a new Management User (a), Realm can be left empty. Enter username and password, if admin is chosen as username there will be an additional confirmation, JBOSS informs that this is a username that is easy to guess.

The created user can be logged in to the dashboard and under *Subsystem Metrics/Datasources* the right configuration of the MySQL datasource can be checked.



*Subsystem Metrics/JPA* gives details about the JPA persistence units (configuration of Hibernate in 2.c).



If the metrics are enabled (can be done with Edit -> enable) transactions, sessions and database queries are counted and can be looked up.

## Hibernate

The JPA configuration file (*CrAc/src/META-INF/persistence.xml*) can be taken without modification, if MySQL database is used and the JDNI-name in *standalone.xml* has not been changed.

<?xml version="1.0" encoding="UTF-8"?>  
<persistence version="2.0" xmlns="http://java.sun.com/xml/ns/persistence" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://java.sun.com/xml/ns/persistence http://java.sun.com/xml/ns/persistence/persistence\_2\_0.xsd">  
 <persistence-unit name="bakk\_server" transaction-type="JTA">  
 <jta-data-source>java:/MySQLDS</jta-data-source>  
 <properties>  
 <property name="hibernate.dialect" value="org.hibernate.dialect.MySQLDialect"/>  
 <property name="hibernate.show\_sql" value="true" />  
 <property name="hibernate.format\_sql" value="true" />  
  
 <property name="hibernate.hbm2ddl.auto" value="create"/>  
 </properties>  
 </persistence-unit>  
</persistence>

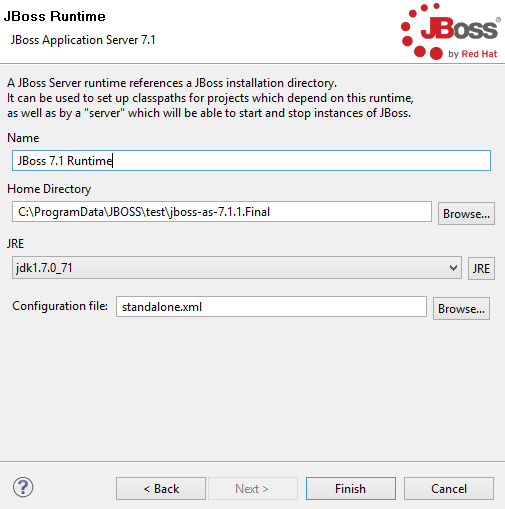
For using another database the hibernate.dialect and the jta-data-source have to be adapted. But the installation was not tested in any other environment.

## Start - Eclipse einrichten

Start IDE and import existing project into workspace.

The next step is to configure the build path to add all dependencies.

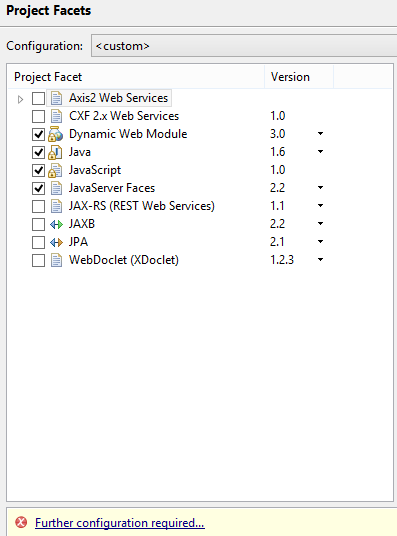
Create a new Targeted Runtime: Open Project->Properties->Target Resources and create a new resource. Select JBoss 7.1 Runtime (if this is not available Eclipse provides a link "*Download additional server adapters*", download the JBoss adapter (JBossAS Tools Server Adapter) and continue) and click *Next*.

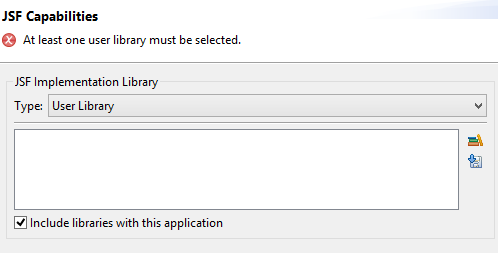


For configuring the server in Eclipse the home directory (*JBOSS/jboss-as-7.1.1.Final*) has to be modified in addition make sure that the used JRE matches the projects Java version (NOT 1.8) and the configuration file is *standalone.xml*. Finish and Apply.

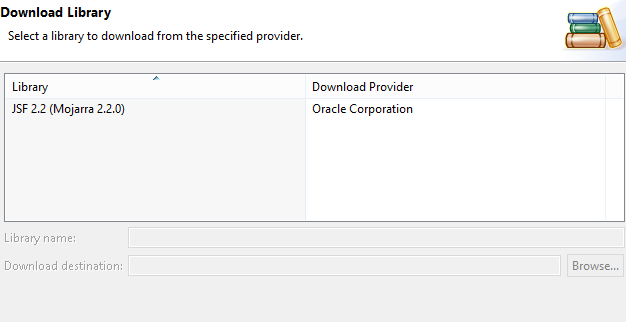
Add the runtime to your project build path if this is not done by the IDE automatically under Project->Properties->Build Path -> Add library -> Server Runtime and select the created Runtime.

Enable JavaServer Faces in Project->Properties->Project Facets.



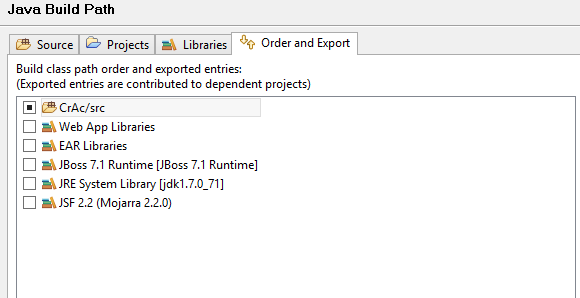
Click on "Further configuration required" and select "User Libraries":  
 

Use the download symbol and download JSF 2.2 (Mojarra 2.2.0).

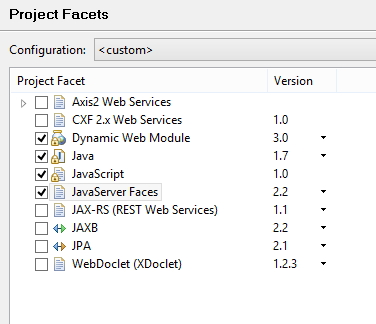


After adding JSF 2.2 apply the changes in the Project Facets.

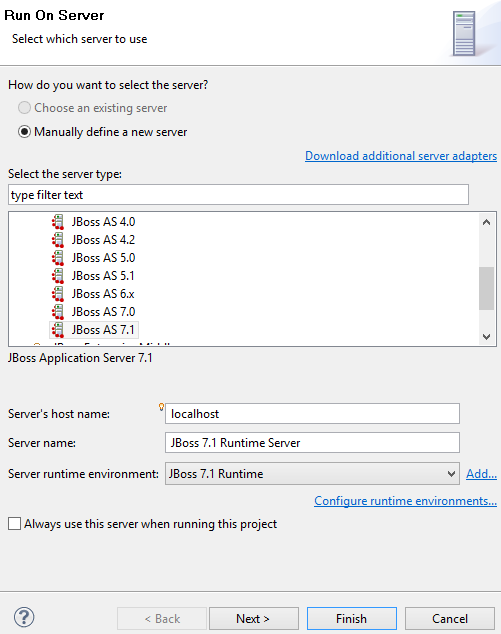
Finally the build path should look like this:



Make sure that the Java version used in the build path and used for compiling the project matches the Java version in the project facets. In this case it has to be changed to 1.7:



To start the server right click on the project, select run on server and configure the server:

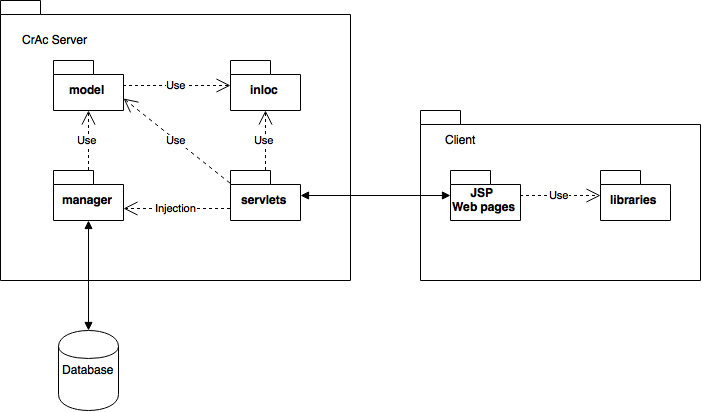


Select "Finish" to start deployment.

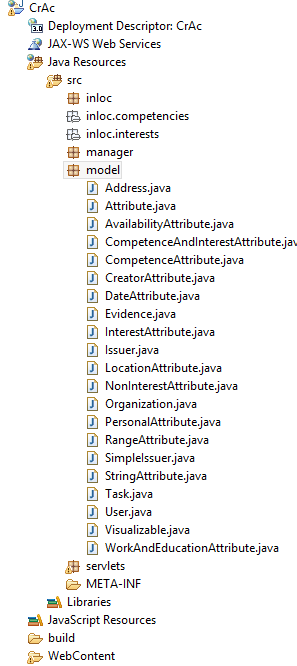
See the logging on the console and after the log entry "*Deployed CrAc.war*" the application is available on *http://localhost:8080/CrAc* (test environment).

In JBOSS/jboss-as-7.1.1.Final/standalone/deployments/CrAc.war you can find the deployed project.

# Architecture



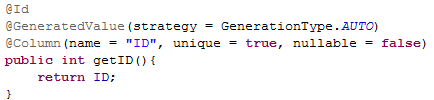
## CrAc Server



### Model

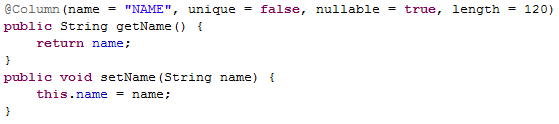
The model is the exact implementation of the model described in xx (TODO Verweis auf model). As we are using Java Perstistence API (Hibernate) the Java classes are annotated as entities for further information on entity annotation see [MuWe12] 2.2.2.

Each entity needs an primary key, annotated like this:

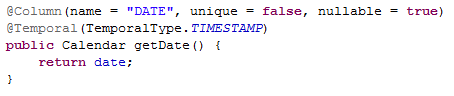


Be careful with replacing standard equals() and hashCode() methods, especially within inheritance relations, as Hibernate is using this two methods to check for equality in a special way.

For all POJO [MuWe12] 2.2.1 getter and setter methods must exist and getters has to be annotated as well. The options of the annotations depend on the type of the POJO. A String for example is annotated like the following:



If java.util.Calendar fields are used they must be annotated like this [MuWe12] p 39:



All methods that are not getters and setters from class fields and should be invisible for JPA and have to be annotated with @Transient.

Inheritance method JOINED is used for all inheritance relations. The methods differ in representation in the database. If method is JOINED there is a separate table for every entity class joined by the same ID [MuWe12] 5.2:

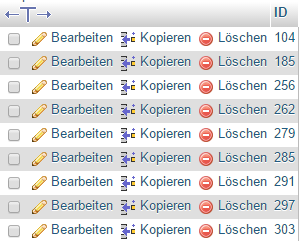
Table issuer:



Table task (extends issuer):

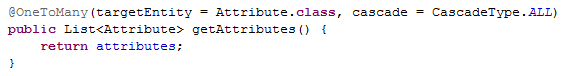


Table user (extends issuer):

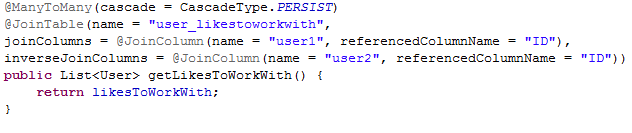


There are six different standard relations between entities: 1:1, 1:n, n:m each unidirectional and bidirectional [MuWe12] 4.1.

Find in the example below a unidirectional 1:n relation:



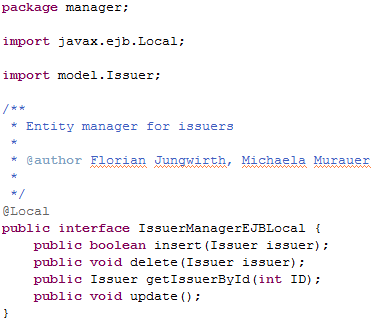
as well as a n:m relation:



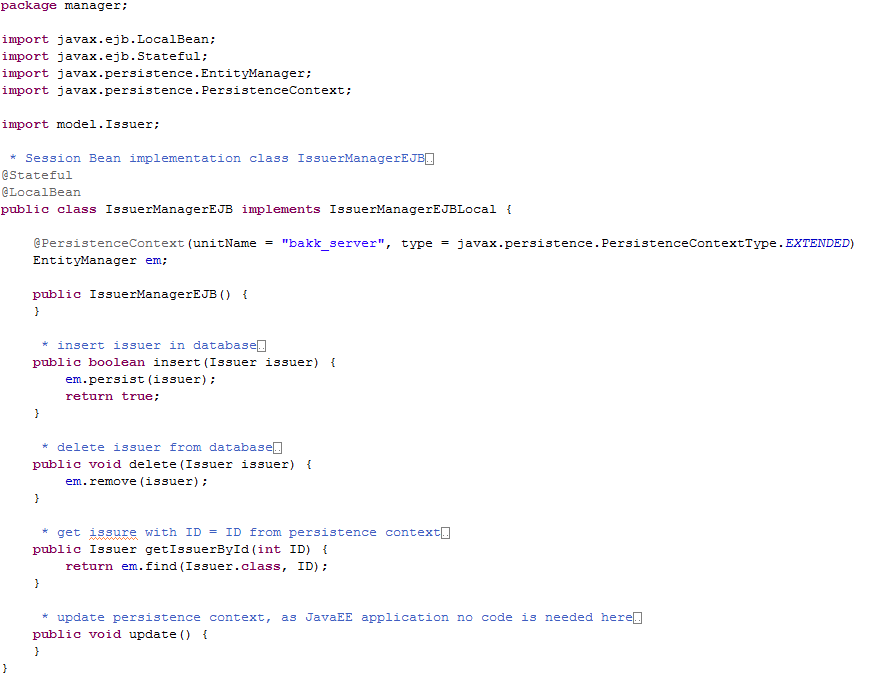
In the examples above different cascade types are used. If CascadeType.ALL is used, every action applied on the actual entity is also applied to all related entities. In the first case an attribute cannot exist without being attached to a user or a task, therefore it will be deleted if the user/task is deleted. In the second example users are related to other users who should not be deleted when the current user is, due to that CascadeType.ALL would be the wrong type here.

### Manager

The manager contains all local beans, to communicate with the database. A manager can exist for each entity and for each manager a interface and a actual implementation is needed. See below a example for a manager bean:

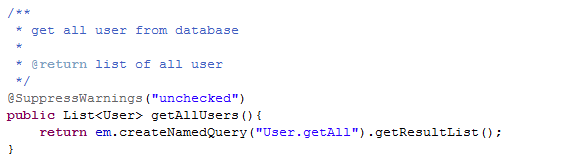


The implementation IssuerManagerEJB looks like the following:



To load, modify and delete entities within the persistence context it must be known that an entity can have different states: transient, managed, detached and removed. The Entity Manager manages all entities within the persistence context and can be used in different modes. The easiest way is to use the extended persistence context JPA 2 provides since the developer does not have to deal with the different states of entities. Use PersistenceContextType.Extended and annotate the EJB with @Stateful as the context should exists for more than one user interaction. Whenever the Entity Manager is accessed the EJB Container commits the transaction and all dirty (modified) entities are synchronized with the database. Therefore the update method can be left empty. For further information read [MuWe12] 3.3.4. The only disadvantage for this method is, that the persistence context can get really big. But as this is not bothering us at this stage of development the extended persistence context seemed to be the right choice.

For keeping the database consistent only the issuer manager would be needed, as this is the highest possible instance in our model. Nevertheless there are also user/task/attribute and evidence manager who provides additionally methods to get all entities of one specific class, this is done by JPA Queries [MuWe12] 7 and is shown in the example below:



To make sure that no detached entities are accessed while committing a transaction updating should be only done with the issuer manager.

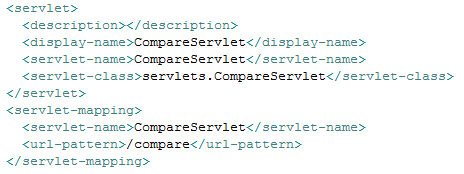
### Servlet

Servlets are used to communicate with the Java Server Pages and with the managers. Each servlet must be annotated like the following:



Different scopes can be used for the annotation of the servlet, find more information in [MuWe12] 6.4.

Additionally the servlet must be registered and mapped in Project/WebContent/WEB-INF/web.xml:



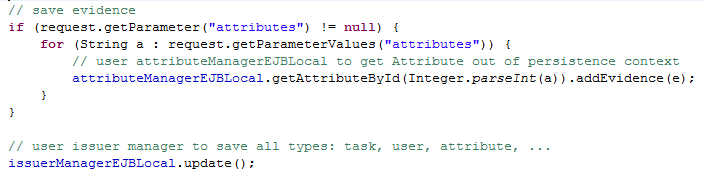
To access the manager within the servlet a annotated instance of a EJBLocal is needed:



After the declaration the manager can be used without further initialization:



Once the entities are loaded in the persistence context they can be modified. After modification they have to be synchronized with the database again. In the following example (EvidenceServlet.java) user attributes are altered and evidences are added. To commit the transaction the update method from the issuer manager is called:



### InLoc

## Client

Every Java Server Page communicates with a servlet. Different JavaScript libraries are used for graphs and design:

Bootstrap (http://getbootstrap.com/) - general design and navigation bars

Timeliner (https://www.technotarek.com/timeliner/timeliner.html) - interactive timeline, events sorted by year used for timeline in user profile to draw WorkAndEducationAttributes

D3:

* Tag Cloud (http://jsfiddle.net/adiioo7/RUTpJ/light/) - tag cloud with JSON input for data used for tag cloud in user and task profile to visualize interests/ non interests/ competencies
* Bubble Chart (http://bl.ocks.org/mbostock/4063269) - non interactive bubble chart with JSON input for data used for Bubble Cloud in user and task profiles to visualize interests/non interests/competencies
* Zoomable Circle Packing (http://bl.ocks.org/mbostock/7607535) - interactive zoomable bubble cloud with JSON input for data used for Zoom Cloud in user and task profiles to visualize interests/ non interests/ competencies
* Force-Directed Graph with Mouseover(http://bl.ocks.org/mbostock/2706022) - entries (gained from JSON) connected by non directed lines used for "has worked with" graph in user profiles
* Directional Force Layout Diagram with varying link opacity(http://bl.ocks.org/d3noob/5155181) - entries (gained from JSON) connected by directed and different opacity lines used for "likes to work with"/ "likes not to work with"/ "overview" graph in user profiles

### Technologien

XAMPP version 1.8.2

Java 1.7 (WARNING: Java 1.8 is not supported by JBoss 7.1.1)

JBoss 7.1.1

JPA 2.0: Hibernate

JavaScript

JQuery

Beschreibungen und Downloadlinks in der Installationsanleitung..keine Ahnung was ich da jetzt noch schreiben soll ..

# Work balance

Florian Jungwirth:

* Model (concept)
* InLOC (concept)
* InLOC (implementation)
* create task / user (implementation)
* compare task/user (implementation)
* InLOC (create files)

Michaela Murauer:

* Model (concept)
* InLOC (concept)
* Model (implementation)
* Manager (implementation)
* user / task profile (implementation)
* InLOC (create files)

[MuWe12]  
Bernd Müller, Harald Wehr: Java Persistence API 2 - Hibernate, EclipseLink, OpenGPA und Erweiterungen, Carl Hanser Fachbuchverlag, München, 2012