well done, corrections in red

Review for Team 5

Standard usage of form objects: too general, is that good or bad?
OO model: why not? possible example for improvement?

Design

- MVC pattern is mostly well-kept. *LoginController:76 checkLoginPageValidity* as a helper method should probably be placed in a service or helper object.
- Standard usage of form objects to handle input data.
- Objects are used where necessary. Not necessarily rich OO design.
- SampleService seems to fulfill same purpose as UserService at least partially. SampleService has too many responsibilities for one class. It has methods that are supposed to be in the UserService and CompetenceService.
  ProfilePicture and Pic seem to share responsibilities as well.
- There are no invariants at all.
- There is little to none code reuse. It's sensible to use a builder to create a new user object since its constructor is rather sizeable. It shouldn't be used to modify a user though.

Coding style

- Coding style is quite consistent (e.g. curly brackets on same line as method definition, correct indentation).
- SampleService is not an intention revealing name. show.jsp is too ambiguous. Else naming is okay.
- SampleService has methods that are the same as the ones in User and Competence services. Pic and ProfilePicture objects seem to be duplicates.
- Encapsulation is okay. There are no unencapsulated fields.
- There are no assertions, contracts or invariant checks
- There are some utility methods (e.g. findCompetence and findCompetenceLike in the SampleService). Since they are quite specific, they don't need to be in a separate utility class.

General comments to coding style

Classes should not print to console especially since you have the SLF4J logger.
Comments should not be used to disable code segments especially in release versions (e.g:

SampleServiceImpl.java:

```
//        User user = new User();
//        user.setFirstName(signupForm.getFirstName());
//        user.setEmail(signupForm.getEmail());
//        user.setLastName(signupForm.getLastName());
//        user.setPassword(signupForm.getPassword());
//        user.setEnableTutor(false);
//        user.setPic(profilePicture);
//        user.setAboutYou(null);
```

should be deleted instead of being put into comments)
In a release version should all implemented methods be finished or else remove them away (e.g: CompetenceServiceImpl.validateCompetence(...) or UserBuilder.modifyUser(...)).
Calling methods and save the return object in a new object, which is never used, is wasting memory (e.g: ProfileController.modifyUser(...) at line 112).

Documentation

- The documentation is well understandable. Sometimes it contains unnecessary information (e.g. difficult to implement) as it's not constructive and not really relevant.
- The documentation does well at revealing the intention of a method as it is written starting with a verb.
- The responsibilities are not very well defined as there seems to be no contracts to be kept with most methods (ties into missing preconditions)
- The vocabulary is domain specific and correct.

Test

- There are only two test cases and both contain the same code.
- There are only 2 tests classes. Both contain only one test case each. With only two test case it's obvious that the test coverage is low.

  Total coverage: 10.3%

  Model: 40.3%

  Pojos: 33.3%

  Service: 9.3%

  Rest: 0%
- testSaveUser() good, easy to understand

  testInvalidUserException(), unclear. according to the name this test case should test for an exception but there is no expected exception.
- No tests with null or boundary values
- The tests are readable and easy to understand

General comments to test

Wrong folder hierarchy of the test folders, they should be in the following relative path src/main/test/java/[1].

[1] https://maven.apache.org/guides/introduction/introduction-to-the-standard-directory-layout.html (16.11.2015)

Review for class (SampleServiceImpl)

The class name does not give any information about what this service is supposed to do.
The class comment is way too generic.

```
/**
 * Provides generic functionalities like searching and saving in the DB.
 *
 * @author ESE Team5
 *
 */
```

It would be a good idea to specify what can be searched for etc.
As for the responsibilities, it could be considered as a god class since it has logic that involves the UserDao, ProfilePictureDao and the CompetenceDao.These methods should be moved each to their own service which for some of them even exists already. Because as of now there is no separation of concerns at all.
I'd suggest to move all the logic concerning the user into the UserService and all the logic concerning competences into the CompetencesService.
The remaining code could be put into a new service called ProfilePictureService. In that way there would be no need for a SampleService.