Florian Jörg 11-923-299
Balz Aschwanden 05-052-691
Raoul Grossenbacher 13-124-532

Review Checklist Team1

Design

- Violation of MVC pattern
    - Functionality of get*.jsp belongs to the Controller and not to the View (get*.jsp in template folder are ok).
    - Connection between specific Controllers and Views is not obvious in respect to their names (e.g. which controller belongs to publicprofile.jsp).
- Usage of helper objects between view and model
    - Forms are good.
    - Pictureuploader should be a service.
    - get*.jsp good. But as mentioned before it should be implemented in the Controller, fed to the jsp via ModelAndView and displayed there by using the c taglib (e.g. <c:if … > … </c:if>).
- Rich OO domain model
    - Classes are structured in a useful way.
- Clear responsibilities
    - Only a few Controllers with multiple responsibilities each (e.g. AcountController).
    - Could be made clearer by creating more controller classes with less responsibilities.
- Sound invariants
    - No invariants found.
- Overall code organization & reuse, e.g. views
    - Reuse generally good. But why are there so many jsp files (One improvement proposal: see Encapsulation)?

Coding style

- Consistency
    - Well done.
- Intention-revealing names
    - In general very good. As mentioned before, controller names could be improved.
- Do not repeat yourself
    - Code is reused reasonably.
- Exception, testing null values
    - Null value testing is used frequently.
    - No Exceptions. But they are not needed because invalid input is handled otherwise.

Florian Jörg 11-923-299
Balz Aschwanden 05-052-691
Raoul Grossenbacher 13-124-532

- Encapsulation
    - Java: very good. All members are private and only accessible through getter and setter methods.
    - Jsp: Use tags instead of always including header and footer files. The readability would improve and so would the file structure.
- Assertion, contracts, invariant checks
    - None used.
- Utility methods
    - Not existing. Not necessary.

Documentation

- Understandable
    - Very good
- Intention-revealing
    - Very good (Names are descriptive too).
- Describe responsibilities
    - Generally good, every class has its responsibilities documented.
- Match a consistent domain vocabulary
    - Good (except for controller names). Especially names within classes are consistent.

Test

- Clear and distinct test cases
    - Seeders are ok, existing tests also. Names are descriptive.
- Number/coverage of test cases
    - Only geodataservice is tested. -> bad coverage.
- Easy to understand the case that is tested
    - Very good
- Well-crafted set of test data
    - Data in Seeders is well chosen.
- Readability
    - Very good. Good names & Good overview.

Florian Jörg 11-923-299
Balz Aschwanden 05-052-691
Raoul Grossenbacher 13-124-532

Class review: AccountController

**Is there some logic that should be moved to another class ? If so, why ?**
The responsibilities of the class are well explained by its method names:
- loginPage()
- signupPage()
- signupResultPage(..)
- editProfilePage(..)
- editProfileResultPage(..)
- publicProfile(..)

The functions handling signup and profile editing fit thematically into the controller, as well as the function for displaying the profile. It is unclear, however, if the login function should be part of this controller. And if so, the logout function should be included into this controller too.
Another point that could be improved is the naming of the controller. Why is it called "AccountController" if all the functions talk about the "Profile"?

**Does the class have too many responsibilities?**
The code structure is very clear and the responsibilities of the single methods are clearly separated. Login and logout could be handled by another class, but otherwise, the responsibilities are fine at the moment. If the complexity of user handling increases, different controllers for profile editing and displaying, as well as user registration could be an option.