Aspect-oriented Refactoring of a J2EE Framework for Security and Validation Concerns

CS 586

Aspect-Oriented Software Development Project

Group Members :

Başak Çakar, Elif Demirli, Şadiye Kaptanoğlu

Bilkent University, Dept. of Computer Engineering

Outline

- Introduction
- Problem Statement
- Requirements Analysis
- Identifying Aspects
- Spring AOP
- Aspect-Oriented Program in Spring AOP
- JBoss Implementation
- Conclusions & Future Work

Introduction

- A software system – asset of modules
- A lot of concerns can not be implemented by module hierarchy
- Concerns crosscut the module hierarchy
  - Scattering
  - Tangling
- Decreased extensibility and maintainability, increased complexity

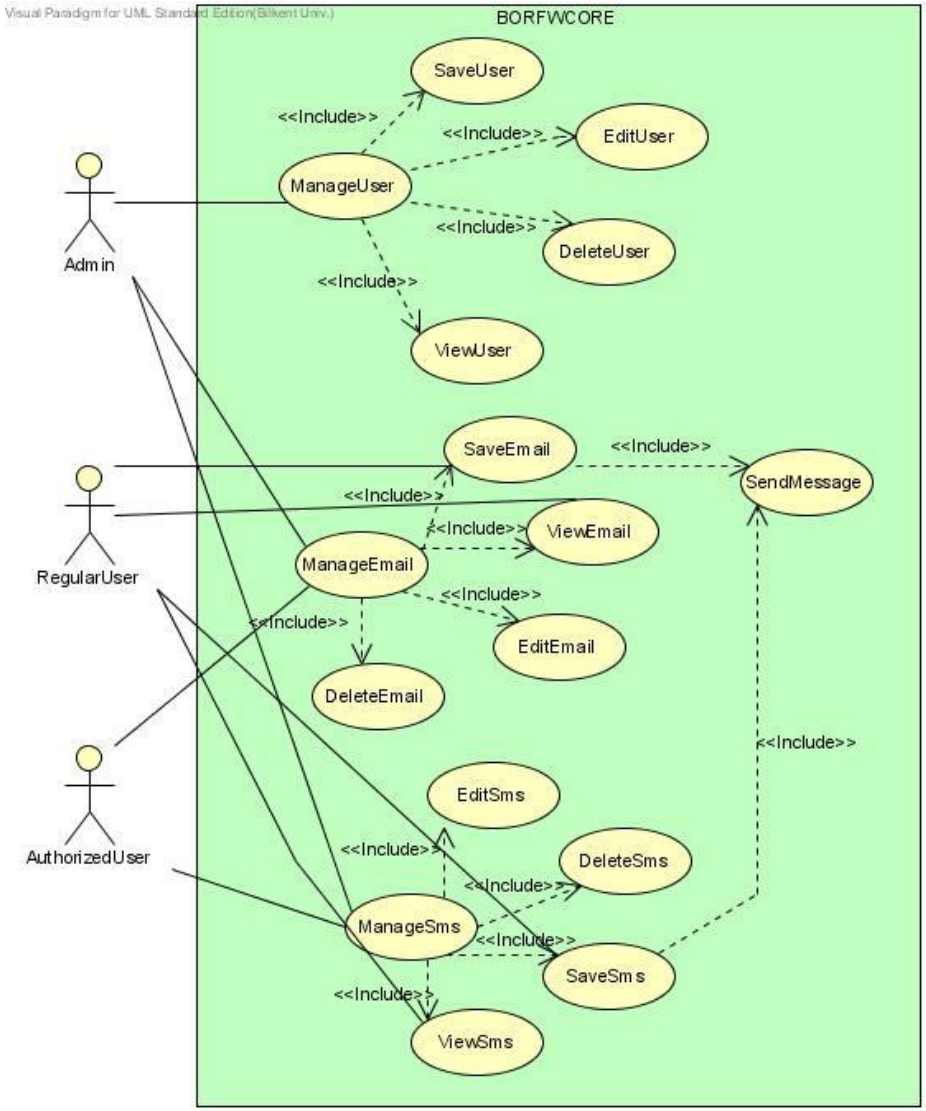
Bor Yazılım

- Bor Soft is a software innovation company focusing on mobile and Web 2.0 software solutions
- Bor Soft has a robust mobile application framework, and a suite of mobile applications
  - Provides customer specific mobile software solutions
- Bor Soft develops Web 2.0 applications and customer-centered Web solutions

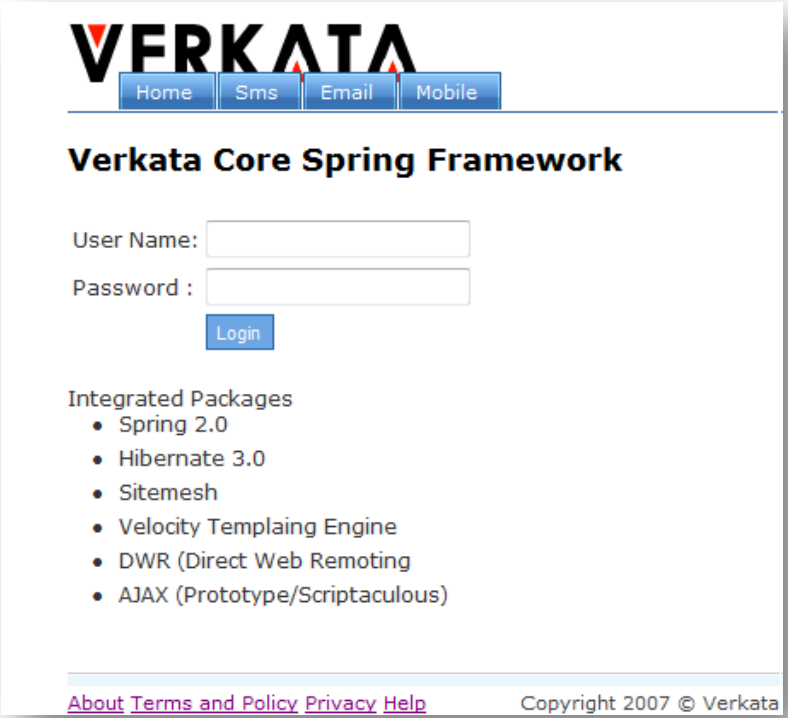
BORFWCORE

- A mobile application development framework
- Uses Struts, Spring, JSF and Hibernate libraries
- Various mobile applications had been developed by Bor Yazilim using this framework
- We use a small example application to show aspect implementation

Example Application



Example Application



Required Extensions

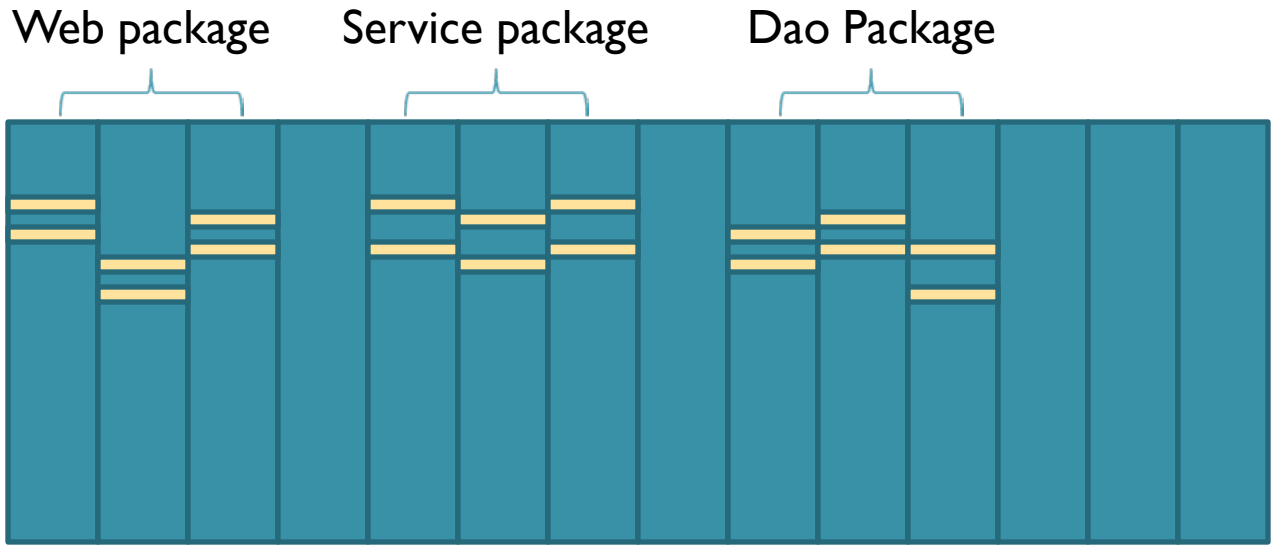
- Validation
  - Check user input format
- Authentication (Security)
  - Check whether user is logged in
- Authorization (Security)
  - Check whether user is authorized
- Access Control (Security)
  - Check whether user has rights to access data

Validation Aspect

- Web applications collect data from users
- Validation is in every class and method which takes user input
- This results in scattering

```
if( sms.getFromNumber().length == 0 ||
       sms.getToNumber().length == 0 )
  {...}
Pattern p = Pattern.compile("\\d{11}");
Matcher m = p.matcher(sms.getToNumber());
Boolean matchFound1 = m.matches();
m = p.matcher(sms.getFromNumber());
Boolean matchFound2 = m.matches();
if( !matchFound1 || !matchFound2 ) {...}
```

Validation Aspect



Authentication Aspect

- Authentication - verify that someone is who they claim they are
- In the example application:
  - User cannot perform any system activity without logging in
- Check this before all method calls
- This causes scattering code

Authorization Aspect

- Authorization - finding out if a user is permitted to do something
- In the application:
  - Authentication is the precondition of authorization
- Check whether user is authorized to perform specific operations before calling these methods
  - Edit, delete, etc.

Access Control Aspect

- Access control - authenticated users access only what they should, nothing more
- In our example application
  - Only admin can see all saved messages and users in the system
  - Other authenticated users should see their own sms messages and email messages
- Check whether user is admin before each related method execution

Spring AOP

- A framework based approach
- Implemented in pure Java, no new language
- Suitable for use in a J2EE web container or application server
- Supports only method execution join points
- The aim is not to provide the most complete AOP implementation but rather to provide a close integration between AOP implementation and Spring IoC

Aspect-Oriented Program in Spring AOP

- We used schema support of Spring AOP
- An aspect is simply a regular Java object defined as a bean in the Spring application context
- The pointcut and advice information are captured in XML

DEMO

JBoss Implementation

- JBoss AOP is a 100% pure java aspect-oriented framework
  - Aspects and advice code blocks are written in regular Java
- Pointcut definitions are put into XML file
- Weaving is dynamic and performed at run time
- Can define method execution, constructor execution, field (get, set), and method call types of pointcuts

Discussions

- An aspect-oriented approach helped us to resolve the crosscutting concerns in the framework
- Development of applications - easier
- Application development effort - reduces dramatically
- Spring AOP is quite easy for implementing aspects on a Spring framework
- When compared to AspectJ, expressiveness of Spring AOP is low

Conclusions & Future Work

- Framework: BORFWCORE
  - Helps to reduce the application development effort
  - Results in code scattering and tangling for some concerns
- Aspect-oriented implementation of crosscutting concerns makes valuable contribution
  - Eliminating code scattering and tangling
  - Increasing the degree of reusability of the framework
  - Reducing application development effort
- Future work: consider a larger case study

Acknowledgements

- We would like to thank to Özer Aydemir in Bor Yaz ılımfor providing BORFWCORE framework to us and explaining the problems they face without an aspect-oriented implementation.
- We also thank to Asst. Prof. Dr. Bedir Tekinerdoğan for consulting us during our project and organizing the 4th Turkish Aspect Oriented Software Development Workshop.

Thanks for Listening

Questions & Comments are Welcome