WPI Suite TNG

Advisor: Prof. Gary F. Pollice

Core
Michael Della Donna
Brian Gaffey
Ryan Hamer
Tyler Wack

Exemplar
Christopher Casola
Andrew Hurle
Jennifer Page
Problem

- A project for software engineering students
- Lack of tools for software engineering students
Previous Solution

- Mainly written by students
- Lacked comprehensive unit tests
- Performance issues
- Lack of Security
- No Client-Server architecture
Goals

- Well-defined module framework
- Configurable
- Client-server
- Excellent documentation
- Exemplar module(s) with documentation
- Open source availability
Approach

Core Team
*Creates Framework*

Exemplar Team
*Leverages Framework*

Feedback / Requirements

Designs
Core
Goals

Flexible
Avoid Complexity
Comprehensive Documentation
Robust and Maintainable
• **Client-Server architecture**
  – Client agnostic

• **Module System**

• **HTTP REST API**
  – Using JSON for communication
Avoid Complexity

- Abstraction
  - Network Communication
  - Persistent Storage
  - Login Security
Network Communication
Avoid Complexity

Persistent Storage Interface

```java
<<Interface>>
Data
+save(aModel : T) : boolean
+retrieve(anObjectQueried : Class, aFieldName : String, theGivenValue : Object) : List<>
+delete(aTNG : T) : T
+update(anObjectToBeModified : Class, fieldName : String, uniqueID : Object, changeField : String, changeValue : Object) : void
+retrieval(aSample : T) : List<T>
+deleteAll(aSample : T) : List<T>
```

DataStore

- myself

1
Avoid Complexity

Login Security

WPILoginServlet

+ WPILoginServlet()
# doPut(request : HttpServletRequest, response : HttpServletResponse) : void
# doPost(request : HttpServletRequest, response : HttpServletResponse) : void
# doGet(req : HttpServletRequest, resp : HttpServletResponse) : void

Authenticator

+ Authenticator(type : String)
+ logout(sessionToken : String) : void
+ login(postString : String) : Session
# parsePost(post : String) : String[]

BasicAuth

PasswordCryptographer

<< Interface >>

+ generateHash(password : String) : String
+ validate(password : String, hash : String) : boolean

Sha256Password
Well-documented components
- Apache Tomcat
- Db4o
- Google GSON

As well as maintaining our own
Robustness

• Creating a module only requires 7 methods
• Descriptive Error Handling
• Server side logging
Architecture Diagram
• 6,192 lines of code
• 52% code coverage
• 37.3% comment ratio
• 40 Classes
Looking Forward

• Dynamic Module Loading

• Increased Test Coverage

• Stronger Web Admin Console
Exemplar Module

Defect Tracker
Goals

- Extensible desktop client
- Example module for students to learn from
- Simple network communication
- Good documentation and design
Design Decisions

- Defect tracking
- Java and Swing
- Dynamic module loading
- Modules provide tabs and toolbars
- Cross-platform look & feel
- Modeless editing
- Network library
Janeway Client for WPI Suite

Login

Username: alice
Password: 
Project: myProject
Server URL: http://localhost:8080/WPISuite/API

Connect
Code Base Metrics

- 6,836 SLOC
- 17.6% Comment Ratio
- 150 classes
- 28.1% Testing Coverage
  - Server-side code: 93%
  - Network library: 70%
Creating an EntityManager

To allow our PostBoard module to save message in the core we must provide an entity manager for PostBoardMessages. The first step is to create a class that implements the EntityManager interface. Each EntityManager is responsible for all requests involving one Model class. The core automatically forwards all requests to the EntityManager. An EntityManager implements the generic `EntityManager<T>` interface.

We will start by creating a new class with the following signature:

```java
public class PostBoardEntityManager implements EntityManager<PostBoardMessage>
```

As shown, you must provide the type of the object that this EntityManager will be storing (PostBoardMessage). Eclipse should also automatically generate method signatures for all of the methods required by the EntityManager interface.

You need to provide a simple constructor for the entity manager that takes as an argument a reference to the database so that it can be saved in a field.

```java
public PostBoardEntityManager(Data db) {
    this.db = db;
}
```

Next, we need to implement the `makeEntity` method. This method is responsible for receiving a message in JSON form, parsing it into an actual PostBoardMessage, and then saving it in the database. It must also return the PostBoardMessage that was saved back to the client. It is called whenever a PUT request is received to /postboard/postboardmessage.

```java
public PostBoardMessage makeEntity($session s, String content)
throws BadRequestException, ConflictException, WPISuiteException {
    // Parse the message from JSON
    final PostBoardMessage newMessage = PostBoardMessage.fromJson(content);
    // Save the message in the database if possible, otherwise throw an exception
    // We want the message to be associated with the project the user logged in to
    if (s.db.save(newMessage, s.getProject())) {
        throw new WPISuiteException();
    }
    // Return the newly created message (this gets passed back to the client)
    return newMessage;
}
```

Let's walk through the code above step-by-step. The first parameter of makeEntity is a Session object containing information about the user making the request. The second parameter is a String containing the body of the HTTP request (in this case the body contains a JSON-encoded PostBoardMessage). The first line of code in the method uses the static method PostBoardMessage.fromJson to convert the JSON content into a new PostBoardMessage. The if block attempts to save the PostBoardMessage in the database, throwing an exception if an error occurs. Importantly, the message is associated with the project that the user is currently logged in to. Finally, the new message is returned. Any
Future Goals

- More testing, especially of the GUI
- HTTPS support, including certificates
- Conflict resolution
- Dashboard screen
- Filters on defect search tab
Results

- Studied D Term Software Engineering
  - Teams developed a Requirements Management module
  - Three MQP team members were coaches
761 commits
16,705 Physical SLOC

3,287 more commits
42,039 more SLOC

A Term

D Term

5 weeks later

Team 1

Team 2

Team 3

Team 4

Team 5
### Create Requirement

**Name:**
- My first epic

**Description:**
- With a description.

**Type:** Epic
- Priority: Medium
**Status:** New
- Iteration: Backlog
**Estimate:**
- Actual:
**Release Number:** 1

---

### Log

**Chris Casola on 04/09/13 11:13 PM**
- Priority BLANK to MEDIUM
- Estimate 0.0 to 3.0

**Chris Casola on 04/09/13 11:12 PM**
- Created Requirement
### Janeway - WPI Suite Desktop Client

#### Requirements Management
- **Create Requirement**
  - Lookup by ID
  - Search Requirements
- **Create Iteration**
- **Navigation**
- **Show Reports**
- **View Release Numbers**
- **View Iteration**
- **View Permission**

#### All Requirements

<table>
<thead>
<tr>
<th>ID</th>
<th>Name</th>
<th>Description</th>
<th>Iteration</th>
<th>Status</th>
<th>Priority</th>
<th>Estimate</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Need to be able to view pie charts</td>
<td>need those pie charts</td>
<td>Backlog</td>
<td>NEW</td>
<td>LOW</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>Have to be able to assign users to ...</td>
<td>kfhndljhsfidshfds</td>
<td>Backlog</td>
<td>NEW</td>
<td>HIGH</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>Typing on keyboard</td>
<td>As a user, I should be able to type on my keyboard</td>
<td>Backlog</td>
<td>NEW</td>
<td>LOW</td>
<td>0</td>
</tr>
</tbody>
</table>
# Project requirements

<table>
<thead>
<tr>
<th>id</th>
<th>Name</th>
<th>Description</th>
<th>Iteration</th>
<th>Status</th>
<th>Priority</th>
<th>Estimate</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>cerlyth</td>
<td>gfgdfghfgdfghfgdfhwe</td>
<td>NONE</td>
<td>NEW</td>
<td>LOW</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>hillwv</td>
<td>hghhwe</td>
<td>NONE</td>
<td>OPEN</td>
<td>NONE</td>
<td>650</td>
</tr>
<tr>
<td>3</td>
<td>gh</td>
<td>This is a well thought out description. It holds text. Make sure to do this this this that hahahaha</td>
<td>4wtdgd4</td>
<td>IN_PROGRESS</td>
<td>NONE</td>
<td>6667</td>
</tr>
<tr>
<td>4</td>
<td>safg</td>
<td>sfsdfsdf</td>
<td>4wtdgd4</td>
<td>IN_PROGRESS</td>
<td>NONE</td>
<td>454</td>
</tr>
<tr>
<td>5</td>
<td>h</td>
<td>ui</td>
<td>NONE</td>
<td>NEW</td>
<td>NONE</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>fg</td>
<td>ng</td>
<td>NONE</td>
<td>NEW</td>
<td>NONE</td>
<td>0</td>
</tr>
</tbody>
</table>

## Requirement - fg

- **ID:** 5  
- **Name:** fg  
- **Description:**  
  
- **Iteration:** NONE  
- **Status:** NEW  
- **Priority:** NONE  
- **Estimate:** 0  
- **Actual effort:** 0  
- **Date of creation:** Apr 12, 2013 3:27:41 PM  
- **Date of last modification:** Apr 12, 2013 3:27:41 PM  
- **Creator name:** josh  
- **Notes:**  

---

**WPI**

[Logo]
Survey Results

**Usefulness of Defect Tracker**
- Not useful: 0%
- Somewhat useful: 45%
- Very useful: 55%

**Confidence in Team’s Ability to Deliver**
- Not confident: 5%
- Confident: 71%
- Very confident: 24%
Survey Results

Ease of Setting Up Development Environment
- 60% Difficult
- 24% Moderate
- 16% Easy

How Helpful was the Documentation
- 71% Helpful
- 26% Very helpful
- 3% Unhelpful
Overall Future Goals

- Virtual machine for development, and/or a bootstrap script
- Pagination
- Database migration
- Email support
- Consistent code style and CodePro audit rules
Repo: github.com/fracture91/wpi-suite-tng

- Governance policy established
- Alumni have expressed interest in contributing
- Will be presented to education community, other schools
- Pull requests welcome!
github.com/fracture91/wpi-suite-tng

Questions?