WHA

Version 1.0

Developer's Guide

Table of Contents

1 Introduction

WHA is a simple web-based application to help HM Solution's employees record and maintain their monthly hours and help the compnay's managers to administer and audit official and financial information in this regard.

1.1 Problem Statement

An employee can create, modify and delete their working times of the current month. At the end of each month they will receive an email notification to finalise their timesheet. After that time, only administrators are able to modify the timesheets. In addition, administrators are able to open (or close) the timesheets for specific employee. Furthermore, administrators are able to manage basic data for application operation.

1.2 Features

There are two main roles in WHA system: “Employee” and “Administrator”. Users with role employee (ROLE_EMPLOYEE) are able to use following features:

1.2.1 Features for Employee

1. **Timesheet Month View** (/time/view/month)**: DailyTimesheet form and list**
2. **Timesheet Weekly View** (/time/view/weekly)**: DailyTimesheet Weekly form**
3. **Travel Month View** (/time/travel)**: DailyTravel form and list**
4. **Travel Weekly View** (/time/travel/weekly)**: DailyTravel weekly form**
5. **Expense Month View** (/time/expense)**: DailyExpense form and list**
6. **List All Timesheet** (/time/timesheet/daily)**: DailyTimesheet list and report**
7. **List All Travels** (/time/timesheet/travel)**: DailyTravel list and report**
8. **List All Expenses** (/time/timesheet/expense)**: DailyExpense list and report**
9. **Open Timesheets** (/time/timesheet/opentimesheets)**:** Open DailyTimesheets
10. **Profile** (/employee/profile)**:** View Profile and edit form to change password

Users with role admin (ROLE_ADMIN) are able to use the following features:

1.2.2 Features for Administrator

- **Create new User and Employee (/admin/signup): Sign up new employee**
- **List all Users (/admin/user): CRUD for User model**
- **List all Online Users (/admin/user/onlineUsers): List all online users**
- **List all Employees (/employee): CRUD for Employee model**
- **List all Timesheets (/admin/timesheet): CRUD for Timesheet model**
- **Search Timesheets (/admin/timesheet/search/form): Search Timesheets**
- **List all Projects (/project): CRUD for Project model**
- **List all Customers (/customer): CRUD for Customer model**
- **List all Technical Roles (/techrole): CRUD for Technical Roles model**
- **List all Roles (/role): CRUD for Role model**
- **List all Constants (/constants): CRUD for Constants model**
- **List all Biz Logs (/bizlog): List, view and delete for Biz Logs model**

1.3 Approach

1.3.1 MVC

MVC (Model-View-Controller) architecture implemented in WHA application by using Spring-MVC framework. Spring-MVC(version 3.x)[1] simplifies the MVC implementation by offering advanced features. For example @Controller and @RequestMapping annotations were helpful to simplify the controller layer. Furthermore Spring-MVC helped us to integrate the Hibenrate ORM and Java Server Pages (version 2) respectively as model and view layers.

1.3.2 Technology Overview

Spring framework[2] and SpringSource-certified architecture was chosen as main structure to design and implement WHA web application. Spring framework integrated following technologies and frameworks: Hibernate framework[3] (ORM

1 http://static.springsource.org/spring/docs/3.0.x/reference/mvc.html
2 http://www.springsource.org/documentation
3 http://hibernate.org/

persistence library) to work with MySQL relational database. Apache Tiles[4](a view templating framework), Jasperreports[5] (report engine library), AJAX, JSON and using Dojo[6] toolkit, Spring-Security, JavaMail, managing theme/i18n/cookie and so on.

For more details please see following table:

Concern	Technology/Framework	Note
Architecture	Spring Framework	
Entity layer	Java Persistence API (JPA)	Hibernate as provider
Web layer	Spring MVC-based front-end that use JSP	
Service layer	Spring's @service	This is optional layer
View Templating	Apache Tiles	
AOP	AspectJ	
Reporting	Jasper Reports	
AJAX and JSON	Dojo Toolkit	
Security	Spring-Security	
Email service	SMTP configured by Spring	

2 Technology

2.1 Project Tools

2.1.1 Spring ROO

Spring ROO[7] is a lightweight Java developer tool that increases software productivity. We started coding[8] the project by using ROO version 1.0.0.RC2[9] in October 2009. Later we upgraded the project to use newer versions, and the latest one was version 1.0.2.RELEASE. During this time, we did not have incompatibility issues and switching to newer version was easy and smooth. Spring ROO has very active forum[10] and community with developers that offering good product and support. For using ROO remember there is a 'hint' (without quote) command that will help you to

4 http://tiles.apache.org/
5 http://www.jaspersoft.com/jasperreports
6 http://dojotoolkit.org/
7 http://www.springsource.org/roo
8 http://code.google.com/p/wha/source/detail?r=2
9 http://code.google.com/p/wha/wiki/SpringROO
10 http://forum.springsource.org/forumdisplay.php?f=67

learn and use available commands. For example you may use below commands in ROO's console to add new entity:

```
roo> hint
roo> hint entities
roo> entity --class ~.domain.Foo
Created SRC_MAIN_JAVA/nl/hajari/wha/domain/Foo.java
Created SRC_MAIN_JAVA/nl/hajari/wha/domain/Foo_Roo_Entity.aj
Created SRC_MAIN_JAVA/nl/hajari/wha/domain/Foo_Roo_ToString.aj
Created SRC_MAIN_JAVA/nl/hajari/wha/domain/Foo_Roo_Configurable.aj
~.domain.Foo roo>hint fields
```

2.1.2 Maven

Apache Maven[11] is a software project management and Java build tool. "Based on the concept of a project object model (POM), Maven can manage a project's build, reporting and documentation from a central piece of information." You could use the Maven's website[11] to found more details about Maven. But here we have our short user guide. It is important to know that all of maven's commands should be execute from the root folder of the project that contain POM (pom.xml) file. Following table shows the most common commands that we used during development:

COMMAND	ACTION
mvn tomcat:run	Tomcat deploy WHA on localhost:8080
mvn jetty:run	Jetty deploy WHA on localhost:8080
mvn test	Run tests
mvn clean	Remove build (target) directory
mvn package	Make the 'war' file
mvn -P prod clean package	Make the 'war' file for the production
mvn eclipse:eclipse	Generate required files to import WHA to eclipse

2.2 Data Layer

Hibernate[12] as a JPA[13] implementation was our preferred technology to manage data layer requirements. Java Bean Validation (JSR-303) including hibernate validator was

11 http://maven.apache.org/
12 http://hibernate.org/
13 http://en.wikipedia.org/wiki/Java_Persistence_API

another integrated library. In addition Java transaction API via Spring transaction abstraction is another used library in WHA project.

2.3 Service Layer

Spring frameworks provide us optional service layer. We used service layer in some specific scenarios. For example we moved business logics to service layer instead of having them in controller layer. Spring's @services stereotype annotation[14] simplifies using of the service layer as simple as creating a class and annotating that class with @Service.

2.4 Web Layer

Spring MVC features (version 3.0) simplifies[15] the implementations in web layer. WHA also includes Apache Tails, Dojo toolkit (via Spring JavaScript), Spring security, JSON (methods in classes for serialization, deserialization and REST support), email. Moreover Spring MVC has nice integration with Jasperreports library thay help us to generate reports on the fly.

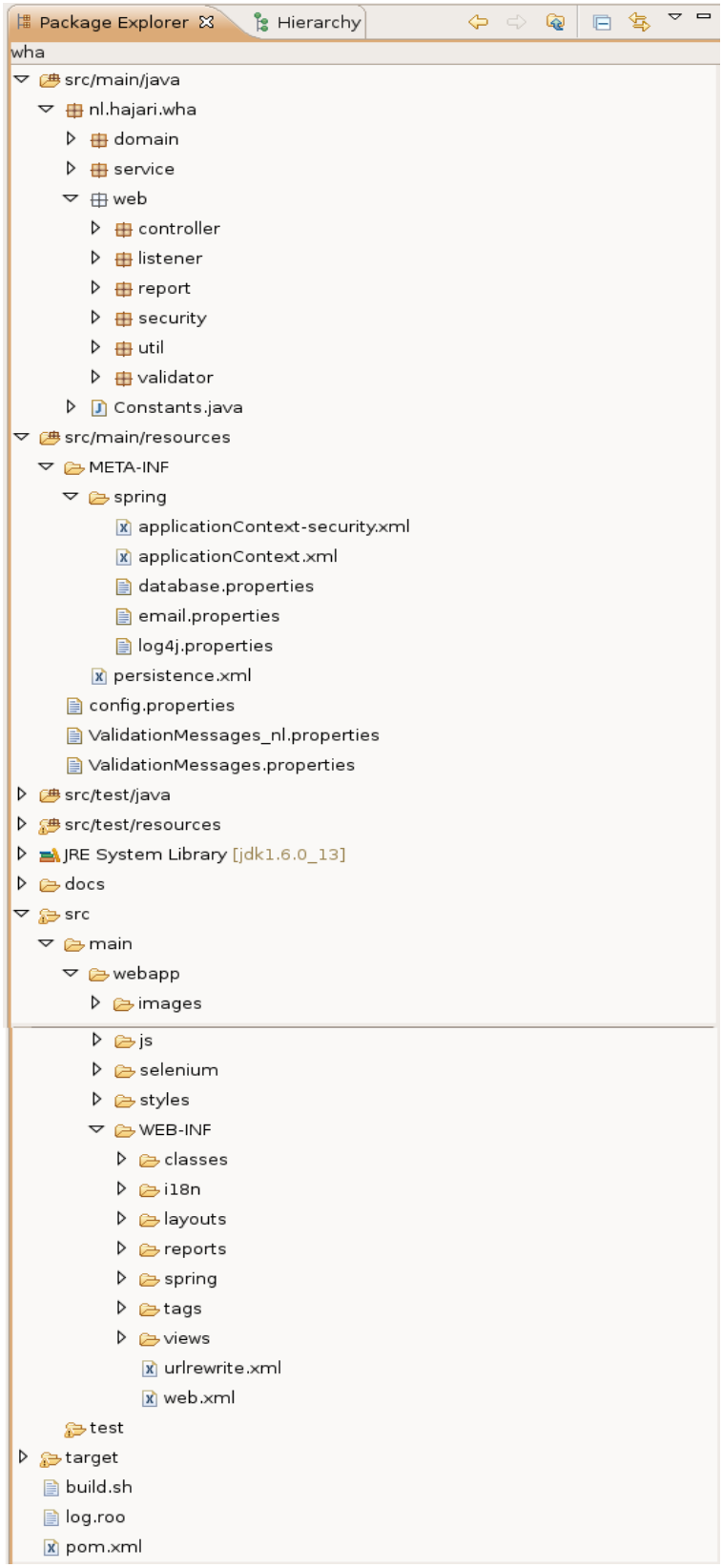
3 Architecture

3.1 Package Structure

WHP project built by Maven build tool and is base on the standard Maven's web application conventions . following picture shows the project package structure from Eclipse IDE.

14 http://static.springsource.org/spring/docs/3.0.0.RELEASE/spring-framework-reference/html/beans.html#beans-stereotype-annotations

15 http://blog.springsource.com/2009/12/21/mvc-simplifications-in-spring-3-0/



3.2 Naming Conventions

We used entity names as convention in different layers of application. For example customer entity has following files in each layer:

FILE	LAYER	NOTE
Customer.java	Data layer	Domain class
CustomerEditor.java	Data layer	Entity select helper class
Customer_Roo_*.aj	Data layer	AspectJ files managed by Roo
Customer_Wha_*.aj	Data layer	AspectJ files managed manually
CustomerService.java	Service layer	
CustomerController.java	Web layer	
CustomerController_\ Roo_Controller.aj	Web layer	AspectJ file managed by Roo
src/main/webapp/WEB-INF/views/customer/*.jspx	Web layer	

3.3 Layers

3.3.1 Entity layer

The role of an entity in WHA application is to model the persistence “domain layer” of the system. As such, a domain object is specific to our problem domain but an entity is a special form of a domain object that is stored in the database. As WHA is a Roo-based application then there is no DAO layer present. Spring-source certificated architecture explain the reasons here[16].

3.3.2 Service layer

3.3.3 Web layer

4 Project Management

WHA project started with the basic requirements on October 2009. First developers prepared simple analysis document, that was just enough to start the project. Extreme programming methodology and it's principle, patterns and practices were used by WHA team. Next they defined series of story/task, that were team's iteration plans to release new features. During development time, developers provide useful document by making connection between issues and source codes. In addition this document will cover all other technical information that future developers may needs to know about WHA project.

16 http://static.springsource.org/spring-roo/reference/html/architecture.html#architecture-dao