VAASAN AMMATTIKORKEAKOULU
VASA YRKESHÖGSKOLA
UNIVERSITY OF APPLIED SCIENCES

Heng Song

Online Event Calendar Application

Department of Technology and Communication

2013

VAASA UNIVERSITY OF APPLIED SCIENCES
Information Technology

ABSTRACT

Author	Heng Song
Title	Web Event Calendar Application
Year	2013
Language	English
Pages	75
Name of Supervisor	Ghodrat Moghadampour

Event Calendar is a J2EE application developed with SSH (Struts, Spring, Hibernate) and mySql which basically is divided into two parts in terms of roles: user and administrator.

For the user, it offers a view of current events by date and search function search events by event properties (name, type, place, date .etc) as well. For administrator, on the other hand, all the functions relative to the management of events and users are available. In specific, administrator is able to add, delete, update, and query the information of events and users.

The idea was to implement the application with given functions with SSH, a combination of three frameworks which are widely used for J2EE projects in industrial production around the last few years and up to now.

So far, all core functions as planed are developed and deployed successfully, the progress of the project was most rewarding and generated an excellent experience in programming.

Keywords Java, Struts, Spring, Hibernate, MySql, Online Event Calendar Application

ACKNOWLEDGEMENT

My best appreciation belongs to all the people who helped me and supported me during my research and study progress. First of all, I would like to give my greatest respect to dear supervisor Dr. Ghodrat Moghadampour, he guided me a lot since the beginning of my thesis, from issuing topic, developing, and to the end of deployment for the project. Not only his outstanding professional skills but also a perseverance attitude to scientific research inspires me to overcome obstacles from both technique and life.

Secondly, I would thank to Dr. Rapila Ritva who has worked hard on revising this article and over the years as my language teacher.

At last, I must give my best appreciation to all the teachers and classmates during the bachelor's degree student career; they both brought me rich experience in life and professional specialization.

CONTENTS

1 INTRODUCTION

Since the blooming of electronic commerce J2EE applications have always had a wide market in the field. However, Struts, Spring, Hibernate, these three frameworks which always appear as a combination and be well-known as SSH, has become one of the most frequently chosen technologies for J2EE projects. I determined to seek for a career as a java programmer which somehow motivates me a lot to develop a project based on the technology mentioned above.

Places like cinema, museum or theater they may need some fast and convenient ways to show an event list with necessary event information which they are going to hold. Online application---Event Calendar is exactly the key to meet the requirements.

A background management platform and a foreground user interface have been designed for Event Calendar. User-friendly is the priority target during the development.

Administrator will be able to manage events and users by using management platform. The user can view the events ordered by date as well as search event by its properties on public pages.

2 TECHNOLOGY OVERVIEW

The core logic part of Event Calendar is developed with struts framework, spring framework and hibernate framework while graphic design is based on the application of Dreamweaver, Java Script, jQuery UI and Adobe Photoshop CS4. MySql is chosen as the database for the project. Followings will be mainly a general introduction to core frameworks in the web application.

2.1 Struts Framework

"The Apache Struts web framework is a free open-source solution for creating Java web applications." /1/

Web applications differ in terms of web page types------conventional websites deliver only static pages while dynamic pages can interact with databases and business logic engines to customize a response. /1/

Database code, page design code, and control flow code are sometimes commingled in web applications based on Java server pages. In practice, it is difficult to maintain larger applications unless these concerns are separated./1/

Model-View-Controller (MVC) architecture is one of ways to separate concerns in a software application. The Model represents code of the business or database, the View represents code of the page design, and the Controller represents the code of navigation. The Struts framework is designed to help developers create web applications that utilize a MVC architecture. /1/

"The framework provides three key components:

A "request" handler provided by the application developer that is mapped to a standard URI./1/

A "response" handler that transfers control to another resource which completes the response./1/

A tag library that helps developers creates interactive form-based applications with server pages.”/1/

2.2 Spring Framework

The Spring Framework provides a comprehensive programming and configuration model for J2EE applications regardless of platform. Infrastructural support at the application level is a key element of Spring: focuses on the “plumbing” of enterprise applications so that teams can focus on application-level business logic, without unnecessary ties to specific deployment environments. /2/

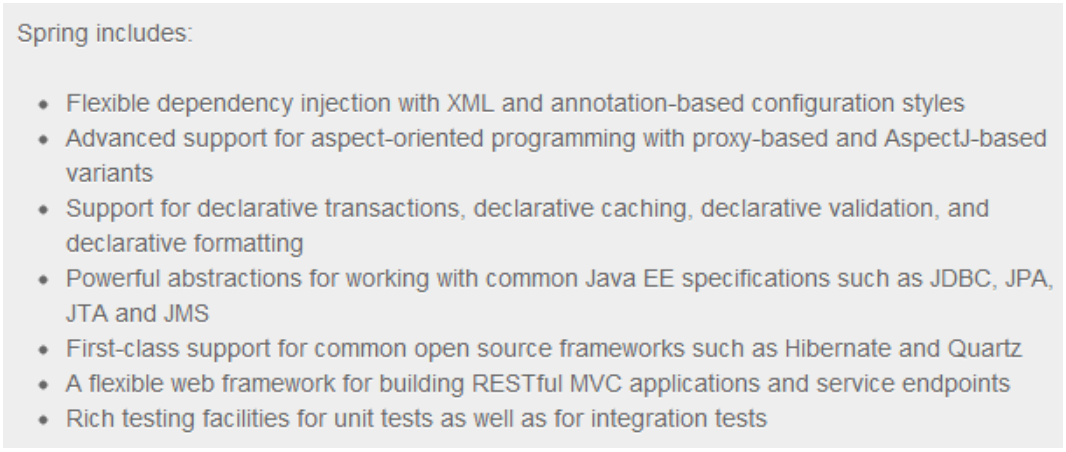

Figure 1. Spring overview /2/

Spring is modular in design, individual parts such as the JDBC support or the core container can be adopted incrementally. /2/

2.2.1 Spring Security

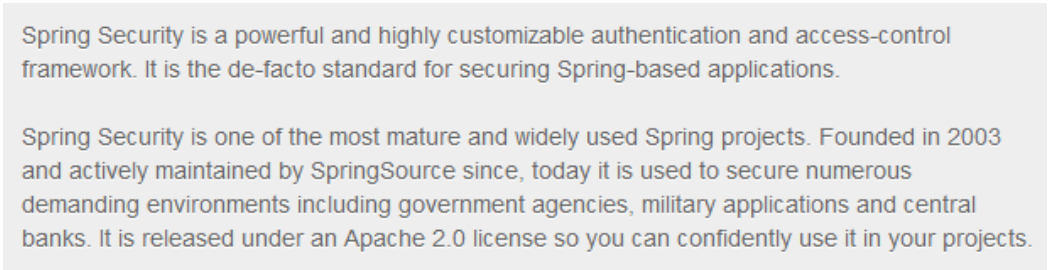

Figure 2. Spring security /3/

2.2.2 Spring Integration

The well-known Enterprise Integration Patterns supported by an extension of the Spring programming model provided by Spring Integration. The primary goal of Spring Integration is to provide a simple model for J2EE projects while separating concerns which is essential for generating testable, maintainable code. /4/

2.2.3 Spring Data

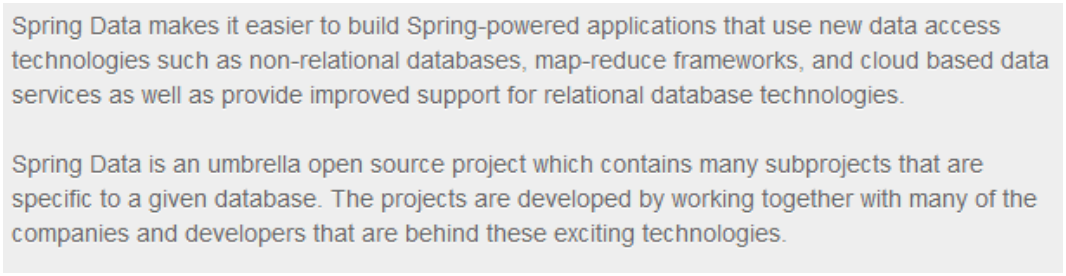

Figure 3. Spring data

The following table describes features of Spring data projects, which covers a large range of frequently used technologies in software industry.

Table 1. Spring data projects /5/

Category	Sub-project	Description
RelationalDatabases	JPA	Spring Data JPA – Simplifies the development of creating a JPA-based data access layer
	JDBC Extensions	Support for Oracle RAC, Advanced Queuing, and Advanced datatypes. Support for usingQueryDSL with JdbcTemplate.
Big Data	ApacheHadoop	The Apache Hadoop project is an open-source implementation of frameworks for reliable, scalable, distributed computing and data

		storage.
Data-Grid	GemFire	Vmwarev Fabric Gem Fire is a distributed data management platform providing dynamic scalability, high performance, and database-like persistence. It blends advanced techniques like replication, partitioning, data-aware routing, and continuous querying.
HTTP	REST	Spring Data REST – Perform CRUD operations of your persistence model using HTTP and Spring Data Repositories.
Key Value Stores	Redis	Redis is an open source, advanced key-value store.
Document Stores	MongoDB	MongoDB is a scalable, high-performance, open source, document-oriented database.
GraphDatabases	Neo4j	Neo4j is a graph database, a fully transactional database that stores data structured as graphs.
Column Stores	Hbase	Apache Hbase is an open-source, distributed, versioned, column-oriented store modeled after Google'Bigtable. Hbase functionality is part of the Spring for Apache Hadoop project.
Common Infrastructure	Commons	Provides shared infrastructure for use across various data access projects. General support for cross-databasepersistence is locatedhere

2.3 Hibernate Framework

“Hibernate is a powerful, high performance object/relational persistence and query service for Java. It lets you develop persistent objects following common Java idiom, including composition, association, inheritance, polymorphism, and the Java collections framework. To allow a rapid build procedure, Hibernate rejects the use of code generation or by tecode processing. Instead, runtime reflection is used and SQL generation occurs at system startup time. It supports Oracle, DB2, MySQL, PostgreSQL, Sybase, Interbase, Microsoft SQL Server, Mckoi SQL, Progress, SAP DB, and HypersonicSQL.” /6/

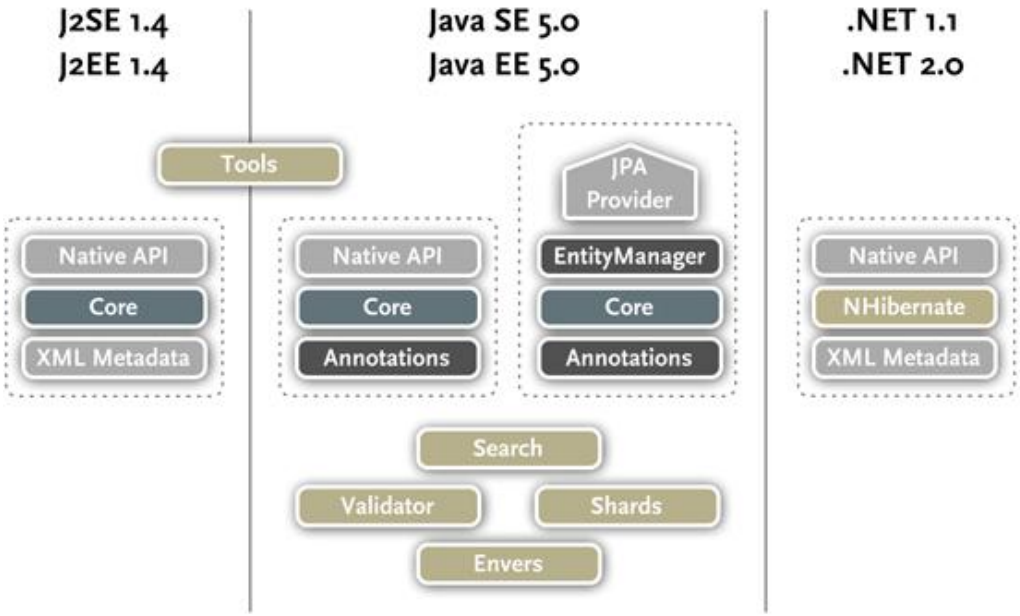

Figure 4. Relational persistence for Java and .NET /7/

3 APPLICATION DESCRIPTION

Below a detailed description will be given of project, requirements, objectives and constraints.

3.1 Functional Description

Event Calendar will be divided into two parts in terms of roles, background for administrator and public pages for user. Functions corresponding to each role will be introduced precisely in the followings.

3.1.1 Background

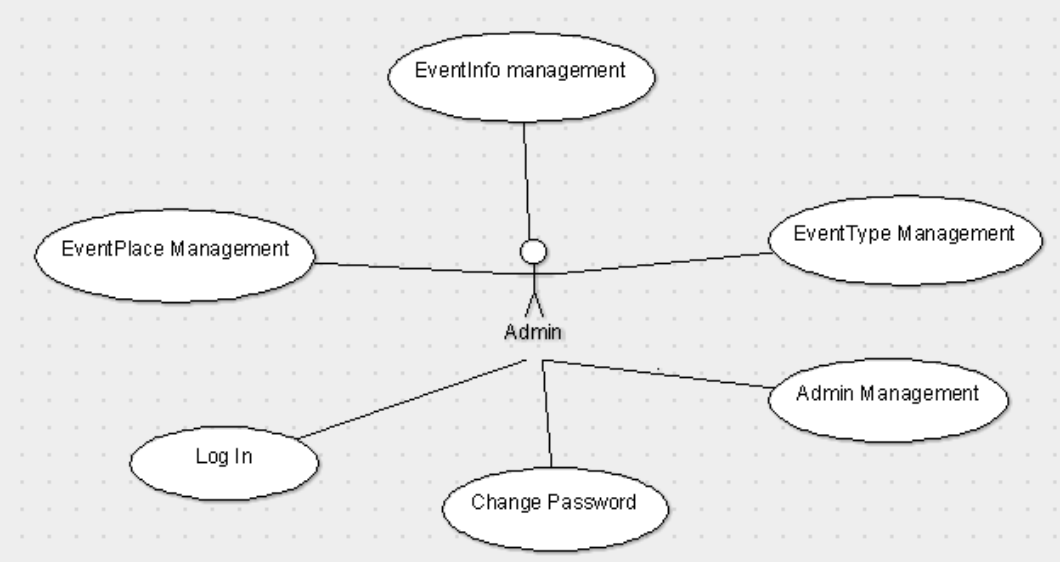

Figure 5. Use case diagram for administrator

3.1.1.1 Log In

Properties:

- Username

- Password

When connected to the background management platform url, a log in page should be displayed and request username and password for administrator to view management pages. Origin username and password is pre-set in database while there is no registration for administrator.

3.1.1.2 Change Password

Properties:

- Username (Unchangeable)
- Password

Administrator can change the password when click the link “Change Password”

3.1.1.3 Admin Management

Admin properties

- Username
- Password

After click the link of “Admin Management”, user will have a view of admin list which shows all administrators’ properties, the priority administrator can delete roles by clicking the link in the end of each row or add administrator to the system by clicking link “add” left down of the list. Group delete is available by clicking the button left down of the list as well.

Two types of administrators are defined, system-admin who both has the authority to manage administrators and events while event-admin can only manage the events. The type of administrator can be set when located in add administrator page.

3.1.1.4 Event Place Management

Place properties:

- ID
- Place Name
- Context

After click the link of “Event Place Query”, a list of event places will be displayed that shows the properties of each place. Admin can update, delete places by clicking the link in the end of each row or add places to the list by clicking link “add” left down of the list. Group delete is available by clicking the button left down of the list as well.

3.1.1.5 Event Type Management

Event Type properties:

- ID
- Type Name
- Context

After click the link of “Event Type Query”, a list of event types will be displayed showing the properties of each event type. Admin can update, delete types by clicking the link in the end of each row or add types to the list by clicking link “add” left down of the list. Group delete is available by clicking the button left down of the list as well.

3.1.1.6 Event Info Management

Event Info properties:

- ID
- EventName

- Image
- EventType
- Place
- StartDate
- EndDate
- Context
- Price

After click the link of “Event Info Query”, a list of events will be displayed showing the properties of each event.

By clicking the name of a certain event will go to a separate page which shows event detail. Admin can update, delete events by clicking the link in the end of each row or add events to the list by clicking link “add” left down of the list. Group delete is available by clicking the button left down of the list as well.

- Supplement description for add/update event
  - Event Place is a property selectable from the places on event place list
  - Event Type is a property selectable from the types on event type list
  - StartDate and EndDate will be supported by a datepicker interface.

3.1.2 Foreground Public Pages

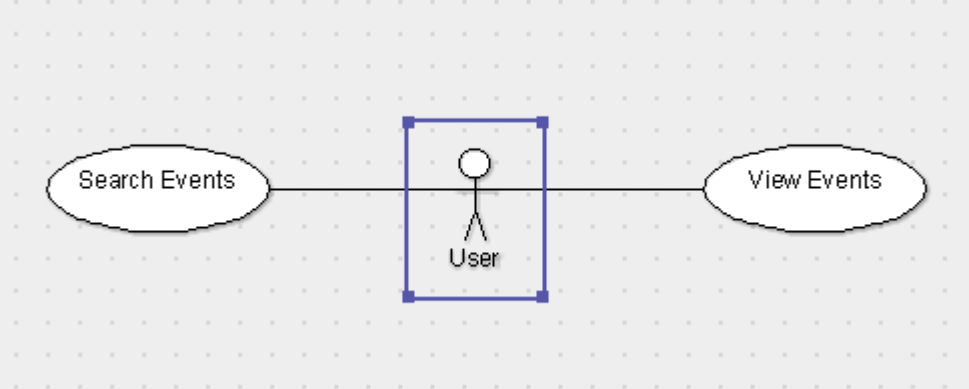

Figure 6. Use case diagram for user

3.1.2.1 Search Event

Event can be searched or group searched by following categories:

- Event Name
- Event Type
- Place
- Date
- Price range

- Event Place is a property selectable from the places on event place list
- Event Type is a property selectable from the types on event type list
- StartDate and EndDate will be supported by a datepicker interface.

After fill in the search information, click search button will forward to search result page which shows event search result with event name and date range. Click event name will go to event detail page shows full properties for the event.

3.1.2.2 View Events

Events is viewed by dates, there are mainly two table lists which Today's Events shows events runs on current day while the other table Later Events shows events coming after current day. Click event name will forward to event detail page however click event type or event place at event detail page will forward to type detail and place detail pages as well.

3.2 Class Hierarchy

Event Calendar is a SSH project which makes full use of classical concept known as MVC, short for model, view and controller. The inside structure will be analyzed precisely in the followings.

3.2.1 Controller Class Diagram

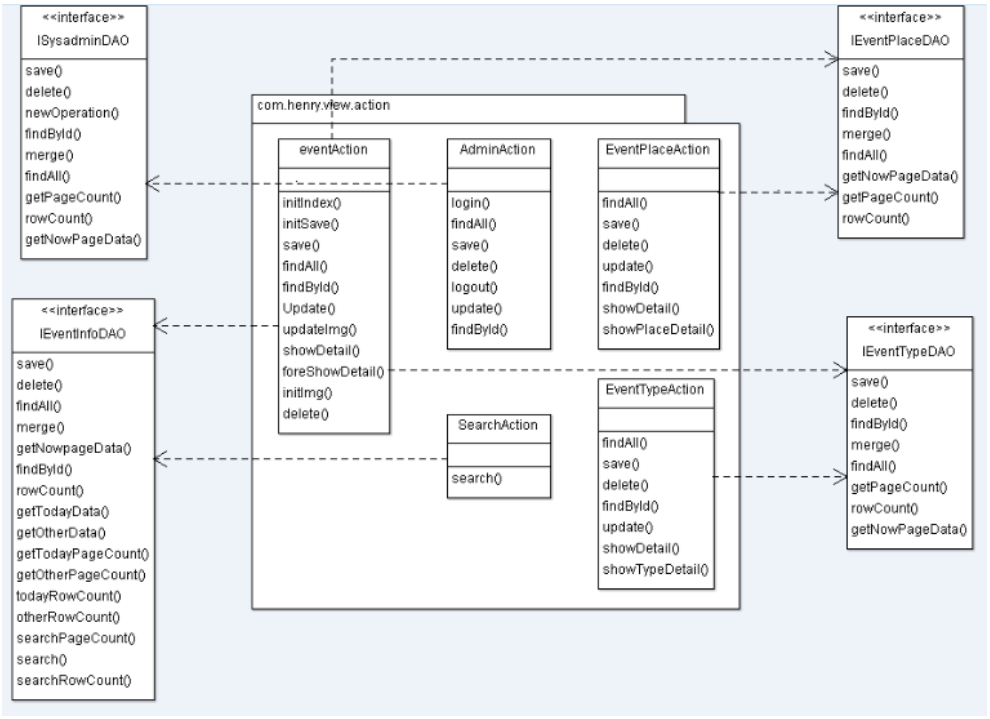

Figure 7. Controller structure

Since five operation objects are defined, correspondingly five struts actions are distributed to control the operation. Here we analyze the "eventAction" as follows.

- initIndex():

  Initialize object and parameter values for search form and event lists as well as makes page division.

- initSave():

  Initializeelement values needed in save() function.

- save():

  Get form parameter values and construct object, then save it in database.

- findAll():

  Query all object values in database

- findById():
  Initialize selected object values, get ready for update object.
- update():
  Get form parameter values except image and construct object, then save it in database.
- initImg():
  Initialize selected object's image value, get ready for update object's image property.
- updateImg():
  Get only form parameter values for image and construct object, then save it in database.
- showDetail():
  Show all properties of selected object and can update meanwhile.
- foreShowDetail():
  Only show all properties of selected object.
- delete():
  Delete selected object(s) from database.

As the figure shows, interfaces IEventInfoDAO, IEventTypeDAO, and IEventPlaceDAO are used to implement functions in this action. In the following model part, specific classes which implement these interfaces will be showed.

3.2.2 Model Class Diagram

In hibernate, POJO class and DAO (Data Access Objects) consist of model part.

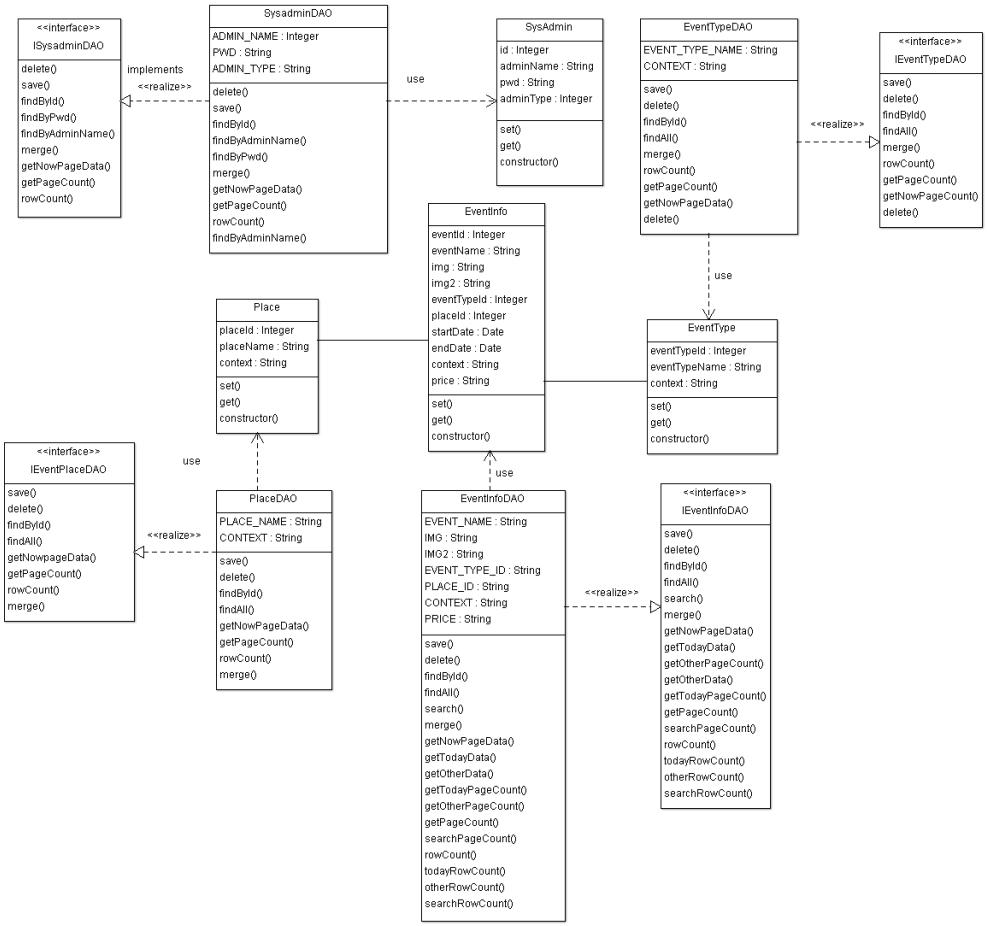

Figure 8. Model structure

There are four POJO classes in this diagram: EventInfo, EventType, EventPlace and SysAdmin, they define 4 entities in this project meanwhile build a mapping with relative tables in database. As a consequence, each row of POJO attribute is relative to corresponding attribute in tables (Object/Relational Mapping). In addition, the entity constructors set and get functions relative to each certain entity's attributes are defined in POJO.

Four DAO in this diagram, they define functions implement the operations oriented to the entities. DAO's attribute refers to the attribute of their corresponding POJO class. Interfaces used in controller part are implemented by DAOs here as well.

3.2.3 View Class Diagram

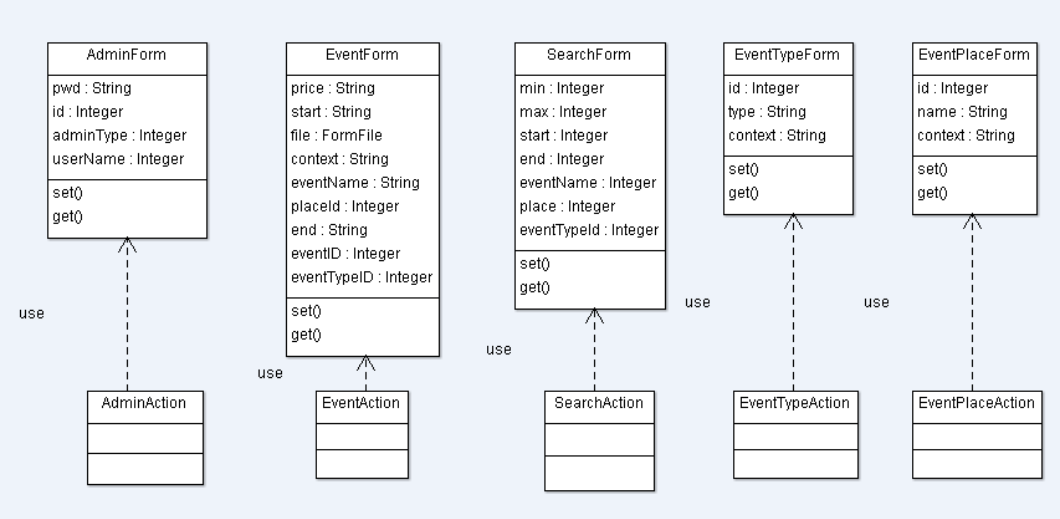

Figure 9. Form class diagram

Forms in struts framework transfer jsp pages' parameter values to the action define in form's attribute named "action" while set and get functions for each parameter are defined in form class to implement the issue.

This diagram shows parameters of each form and the mapping between forms and actions. Here the five actions are exactly the same actions defined in controller part.

3.3 Sequence Diagram

Since eventInfo, eventPlace, eventType management has the same sequence logic, here shows the sequence diagram of eventInfo management only, administrator's login and search sequence will be analyzed in this chapter as well.

3.3.1 Event Management

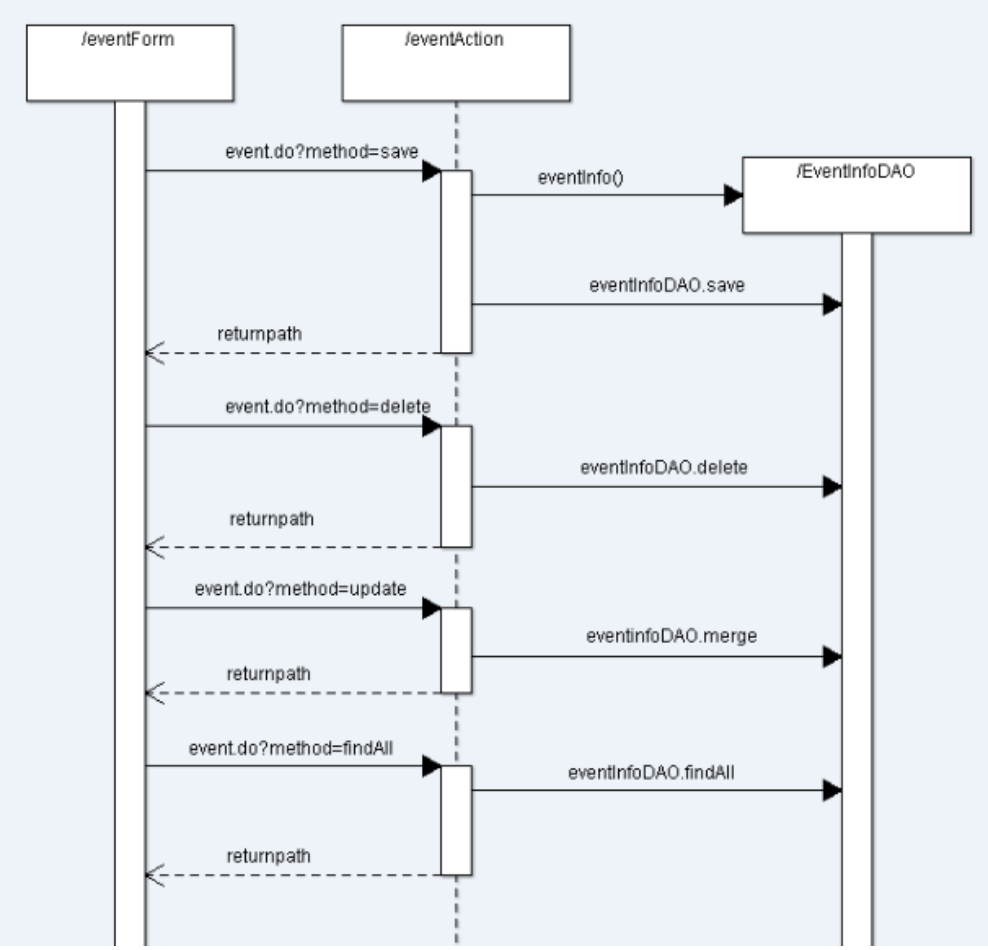

Figure 10. EventInfo management sequence diagram

Form transfers parameters' values to corresponding functions in action through form's <action>tag, after action gets parameters' values and construct object, DAO will be called to complete database operation, at last action will forward to a result feedback page.

3.3.2 Admin Login

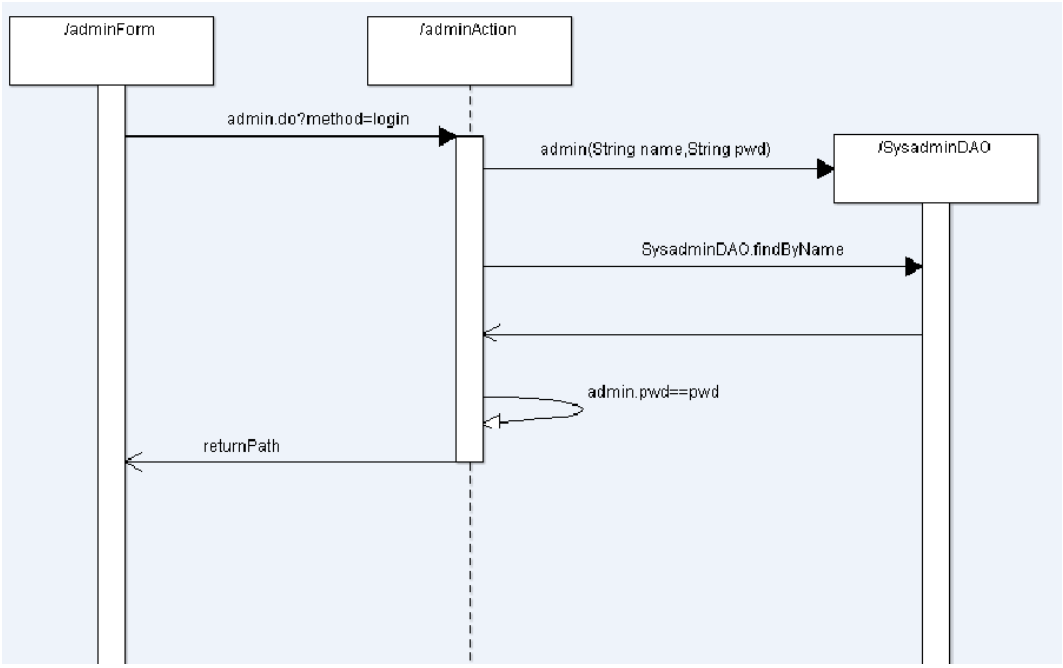

Figure 11. Admin login sequence diagram

Form transfers administrator's name and password to action, after action gets the parameters' values and construct object, a judge of whether username exist in database will be done by DAO, if username exists, the password corresponding to the username will have a comparison with input password, action will forward to different return path based on the judge results.

3.3.3 Search Event

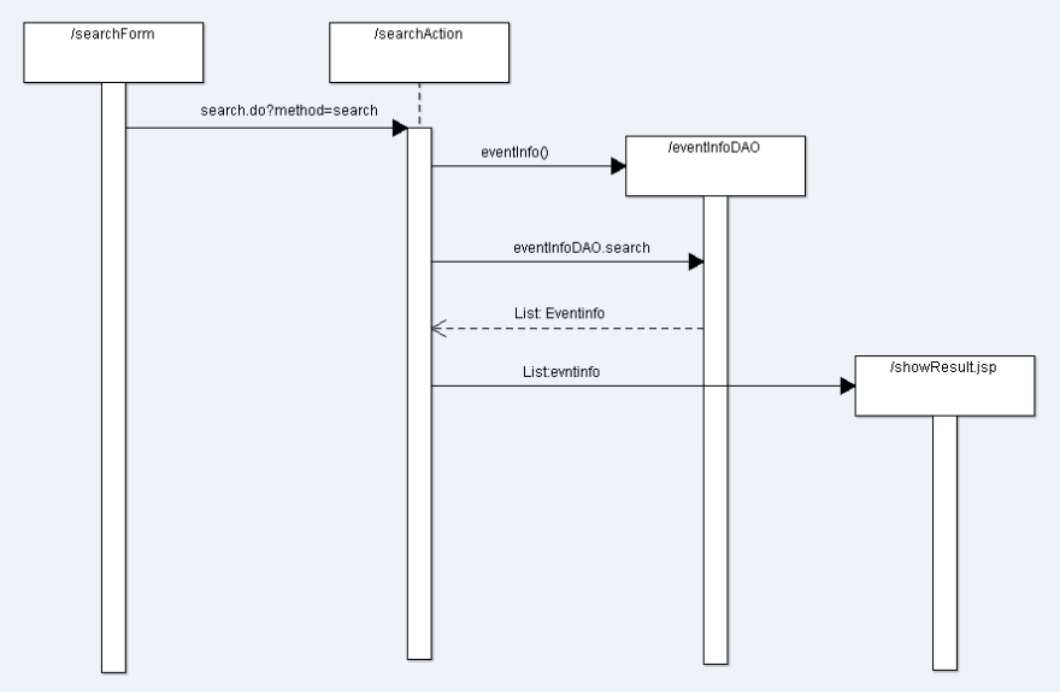

Figure 12. Search event sequence diagram

After action get parameters' values from form and construct object, DAO will be called to search event and return a list of events, then action will send this list to result display page.

4 DATABASE AND GUI DESIGN

4.1 Database Design

MySql is chosen as the database, tables' structures and attributes are as the same as POJO classes in Figure 5. Model Structure shows.

```
Create table eventType(
eventTypeIDint AUTO_INCREMENT primary key,
eventTypeNamevarchar(50) not null,
contextvarchar(200)
);
alter table eventType
add CONSTRAINT eventTypeName_unique unique(eventTypeName);
create table place(
placeIDint AUTO_INCREMENT primary key,
placeNamevarchar(50) unique not null,
contextvarchar(200)
);
create table eventInfo(
25ventideint AUTO_INCREMENT primary key,
eventNamevarchar(50) not null,
imgvarchar(50) ,
img2varchar(50) ,
eventTypeIdint not null,
placeIdint not null,
startDate date not null,
endDate date not null,
contextvarchar(200) ,
pricevarchar(50)
```

```
);
ALTER TABLE `eventInfo`
DROP INDEX `eventName_unique`;
alter table eventInfo
alter price set default "0" ;
alter table eventInfo
add constraint FK_eventTypeId
FOREIGN KEY(eventTypeId) REFERENCES eventType(eventTypeID);
alter table eventInfo
add constraint FK_placeId
FOREIGN KEY(placeId) REFERENCES place(placeID);
create table sysadmin
(
idint AUTO_INCREMENT primary key ,
adminNamevarchar(20) not null ,
pwdvarchar(20) ,
adminTypeint not null
);
```

Snippet 1. Database design

The snippet above shows table constraints and primary/foreign key relationships. In specific, for table eventInfo, it has two foreign keys, placeId and eventTypeId for inner join query with table place and eventType.

4.2 GUI Design

Event Calendar's graphic user interface design mainly depends on css style sheet and java script (jQuery UI).

jQuery UI is a curated set of user interface interactions, effects, widgets, and themes built on top of the jQuery JavaScript Library. Whether you're building highly interactive web applications or you just need to add a date picker to a form control, jQuery UI is the perfect choice. /8/

4.2.1 Background

- Admin Log In page

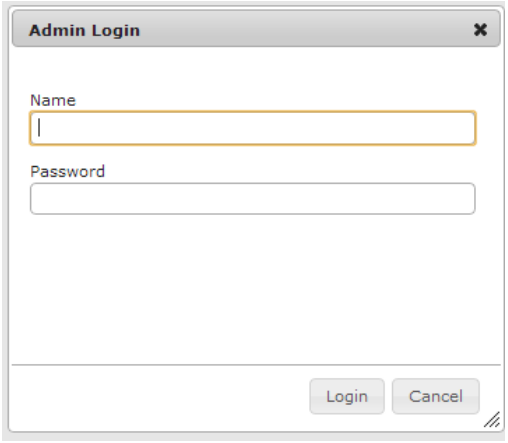

Figure 13. Login page

An application of jQuery UI “#dailog:ui-dialog”

- Home Page

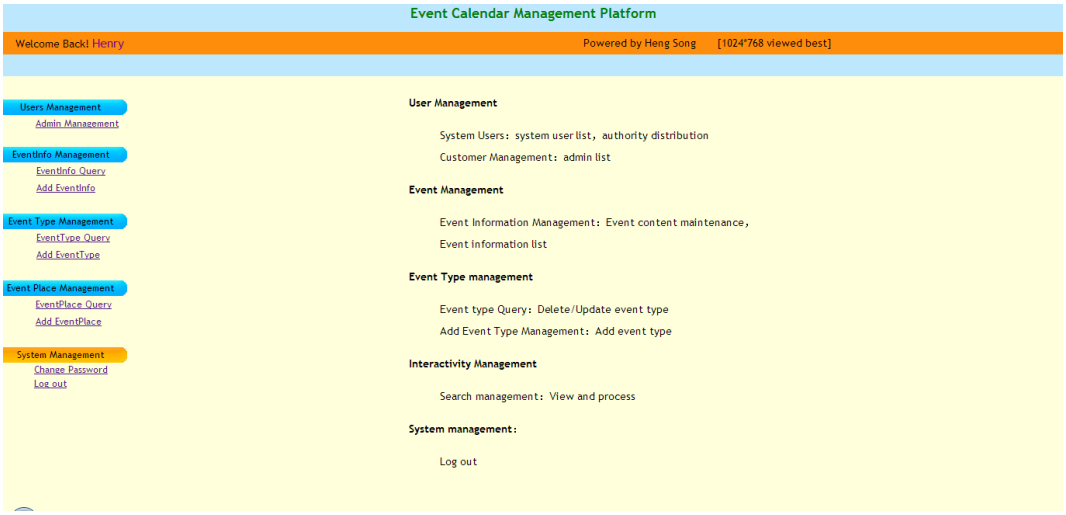

Figure 14. Home page

Use of frame tag for JSP page, constructed by 3 frames: leftFrame, mainframe, topFrame.

- Event Management Page

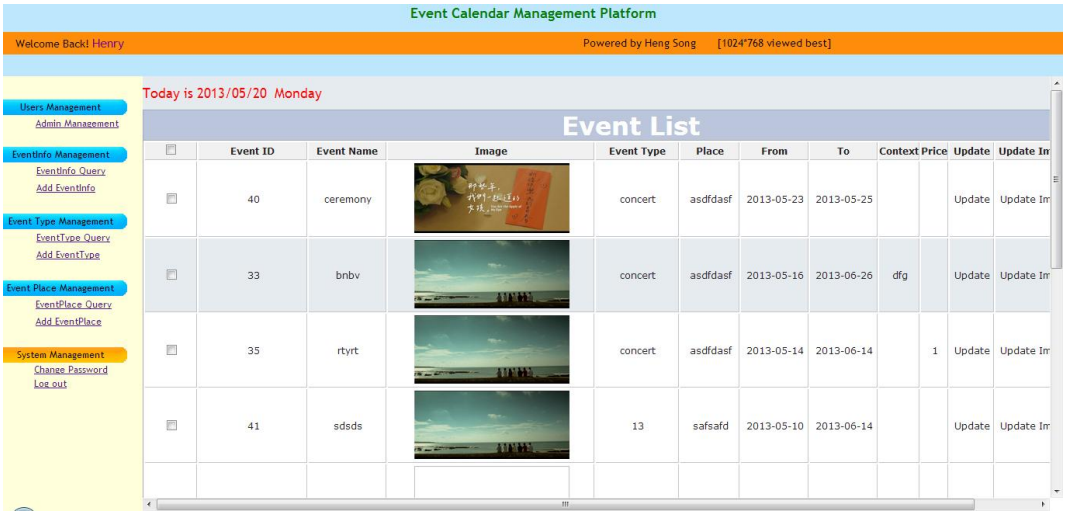

Figure 15. Event management page 1

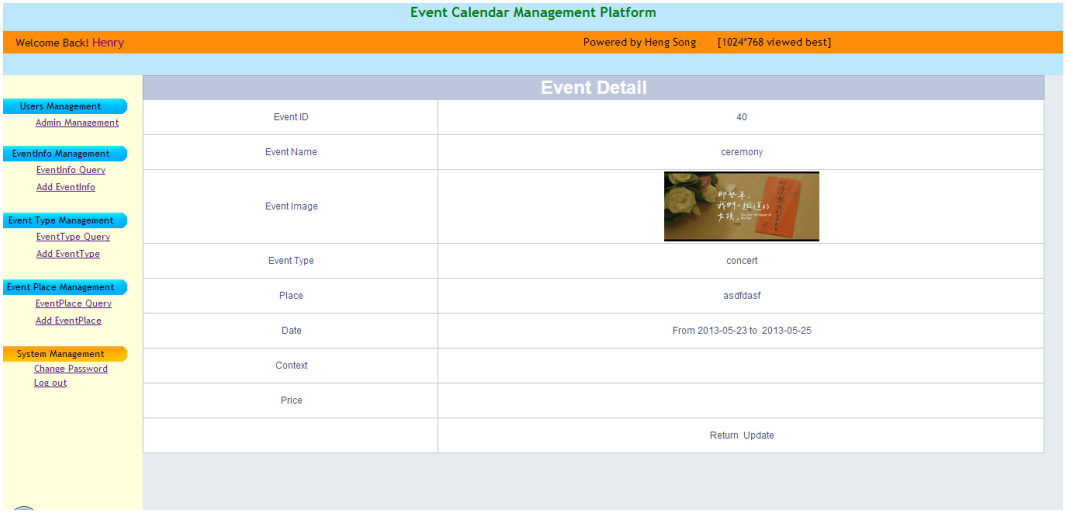

Figure 16. Event management page2

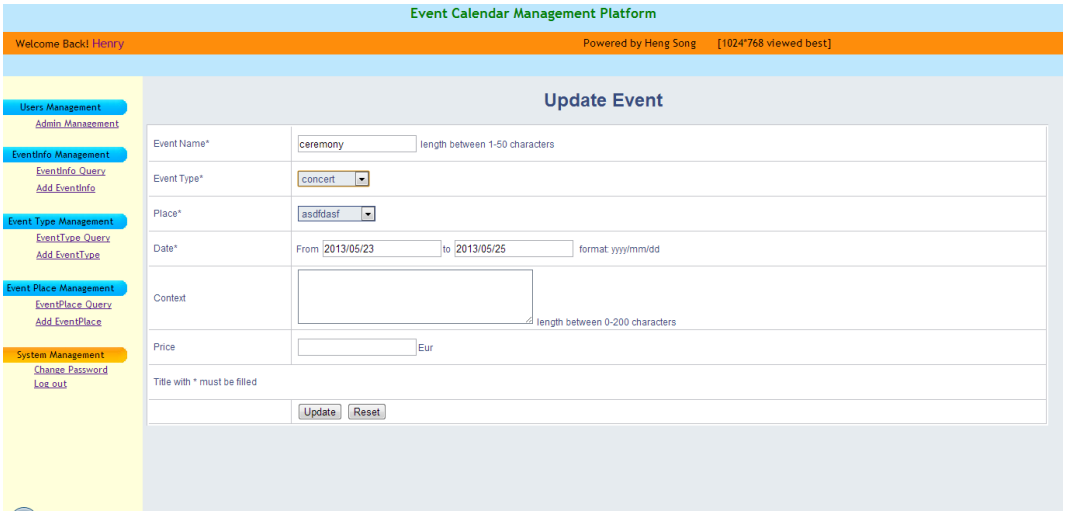

Figure 17. Event management page3

Edited by css style sheet

- Date Picker

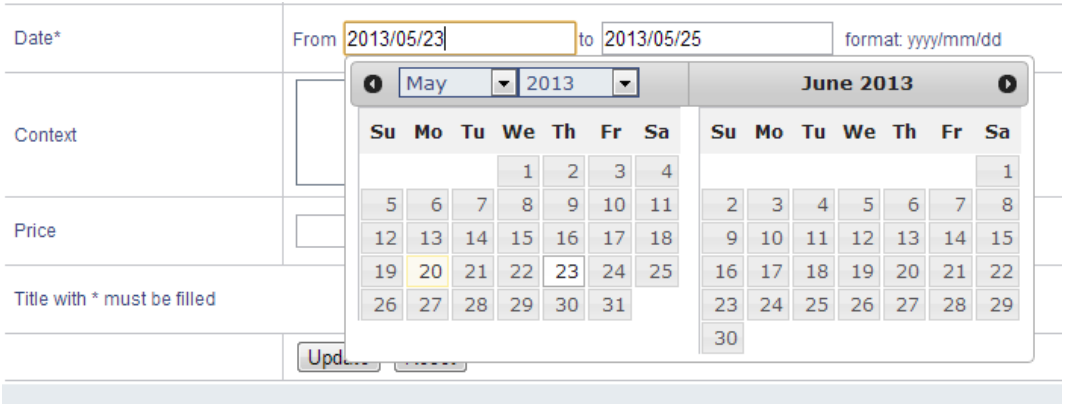

Figure 18. Date picker

Application of jQuery UI #datepicker.

Above is the general view of application's UI style, pages relative to other objects are similar as the above shows.

4.2.2 Foreground

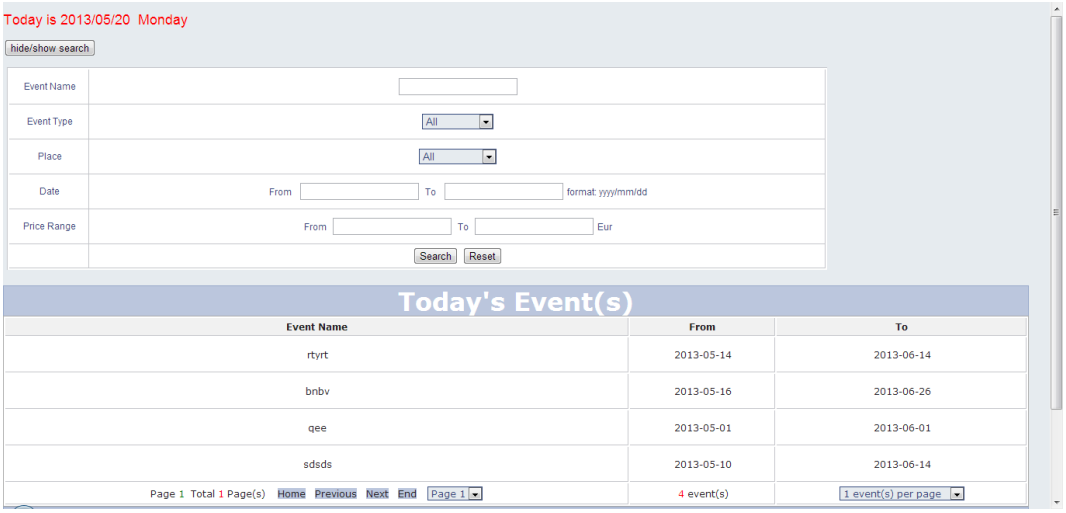

Figure 19. Public page

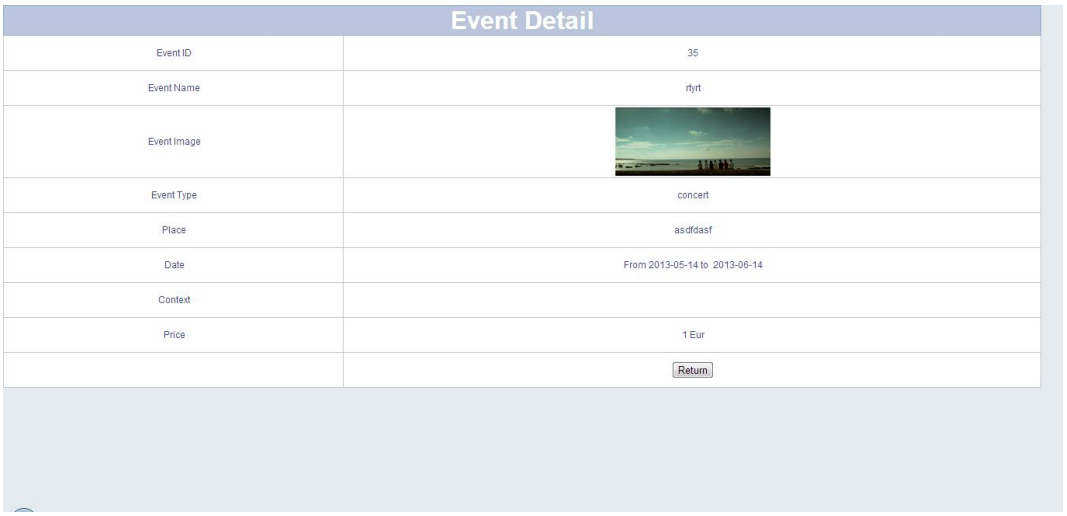

Figure 20. Event detail

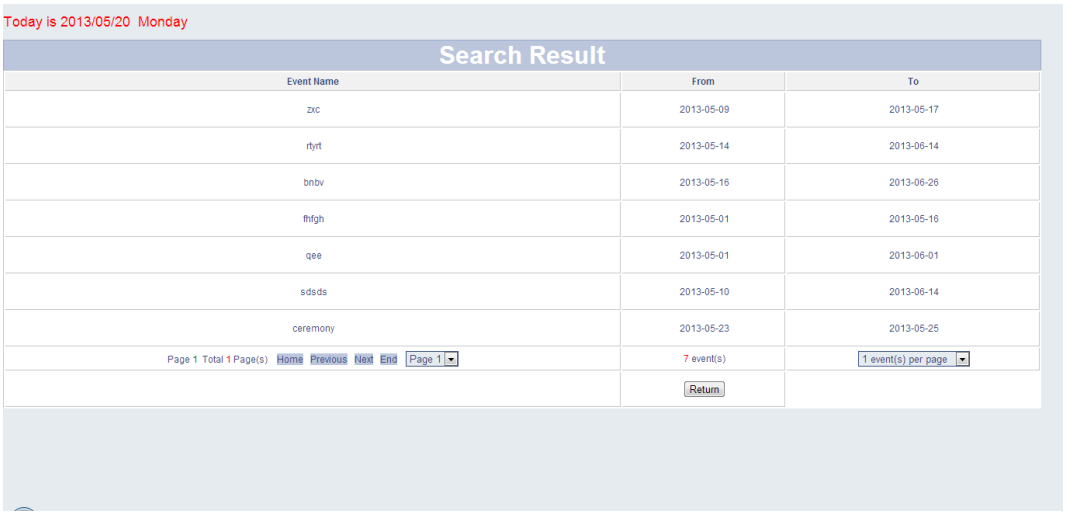

Figure 21. Search result

Make use of both jQuery UI and css file.

5 IMPLEMENTATION

5.1 General Description

Generally, implementation can be divided into UI implementation and functions implementation. As mentioned above, UI mainly implemented by css and jQuery UI based on JSP page, functions are implemented by the ways showed in class diagrams and sequence diagrams. In the following, precise description will be showed with analysis of code.

5.2 Implementation for GUI

5.2.1 Layout

```
<frameset rows="93,*" cols="*" frameborder="NO" border="0"
framespacing="0">

<frame name="topFrame"  scrolling="NO"
   noresizesrc="${pageContext.request.contextPath}/background/top
.jsp" >

<frameset cols="180,*" frameborder="NO" border="0" framespacing="0"
rows="*">

<frame name="leftFrame"
   noresizescrolling="AUTO"src="${pageContext.request.contextPath
}/background/left.jsp">

<framename="mainFrame"src="${pageContext.request.contextPath}/back
ground/main.jsp">

</frameset>

</frameset>
```

Snippet 2. Layout structure

<frameset> tag defines display page into 3 frames, and set top.jsp shows at top Frame, left.jsp shows at leftFrame, main.jsp shows at mainFrame.

5.2.2 Responsive Layout

```
<tr>

<td width="40"> </td>

<td    class="wr4"
          width="120"><ahref="<%=request.getContextPath()%>/event
       .do?method=findAll"target="mainFrame">EventInfo
       Query</a></td>

</tr>
```

Snippet 3. Responsive layout

Above is only a fragment from codes for responsive layout however it's a good instance for showing how responsive pages display on target frame.

This code implements that responsive page of findAll function ineventAction display on mainFrame.

Other responsive layout implemented as the same way as the above code shows.

5.2.3 Use of jQuery UI

There are several jQuery UI applications in GUI implementation, here gives one of the applications: datepicker as an instance.

- Step 1: Import library and style sheet

```
<script
src="${pageContext.request.contextPath}/js/jquery/jquery-1.7.2.js"
></script>

<script
src="${pageContext.request.contextPath}/js/jquery/jquery.ui.core.j
s"></script>
```

```
<script
src="${pageContext.request.contextPath}/js/jquery/jquery.ui.widget
.js"></script>

<script
src="${pageContext.request.contextPath}/js/jquery/jquery.ui.datepi
cker.js"></script>

<link
    rel="stylesheet"href="${pageContext.request.contextPath}/js/jq
uery/jquery.ui.all.css">

<link  rel="stylesheet"
href="${pageContext.request.contextPath}/js/jquery/demos.css">
```

Snippet 4. Import jQuery and css libry

- Step 2: Programfunctions

```
<script>

$(function() {

var dates = $( "#from, #to" ).datepicker({  //set a mark for invoking
UI

defaultDate: "+1w",

changeMonth: true,

changeYear: true,        //set whether year and month can be changed

numberOfMonths: 2,        //control the number of date picker appeared

onSelect: function( selectedDate ) {

var option = this.id == "from" ? "minDate" : "maxDate",

instance = $( this ).data( "datepicker" ),

date = $.datepicker.parseDate(

instance.settings.dateFormat ||

$.datepicker._defaults.dateFormat,

selectedDate, instance.settings );

dates.not( this ).datepicker( "option", option, date );

}

});
```

```
});
</script>
```

Snippet 5. Datepicker function

Tips: output date format after the pick from datepicker can be edited in jquery.ui.datepicker.js.

- Step 3: Call functions

```
<tr>
<td width="96" height="40" align="right">Date*</td>
<td height="40" colspan="3">From
<input type="text"
name="start" id="from">to  //call the mark set in snippet5 #from
<input type="text"
//call the mark set in snippet5#to
name="end" id="to"> format: yyyy/mm/dd
</td>
</tr>
```

Snippet 6. Call jQuery UI example

After completing all the three steps above, when users click input ”from” or ”to”, they will view datepicker and make agile operations for input date.

There is another several jQuery UIs despite datepicker, implemented in a similar way.

5.3 Implementation of Functions

5.3.1 Integration of Struts, Spring, Hibernate

- Step 1: ImportSSHlibaries

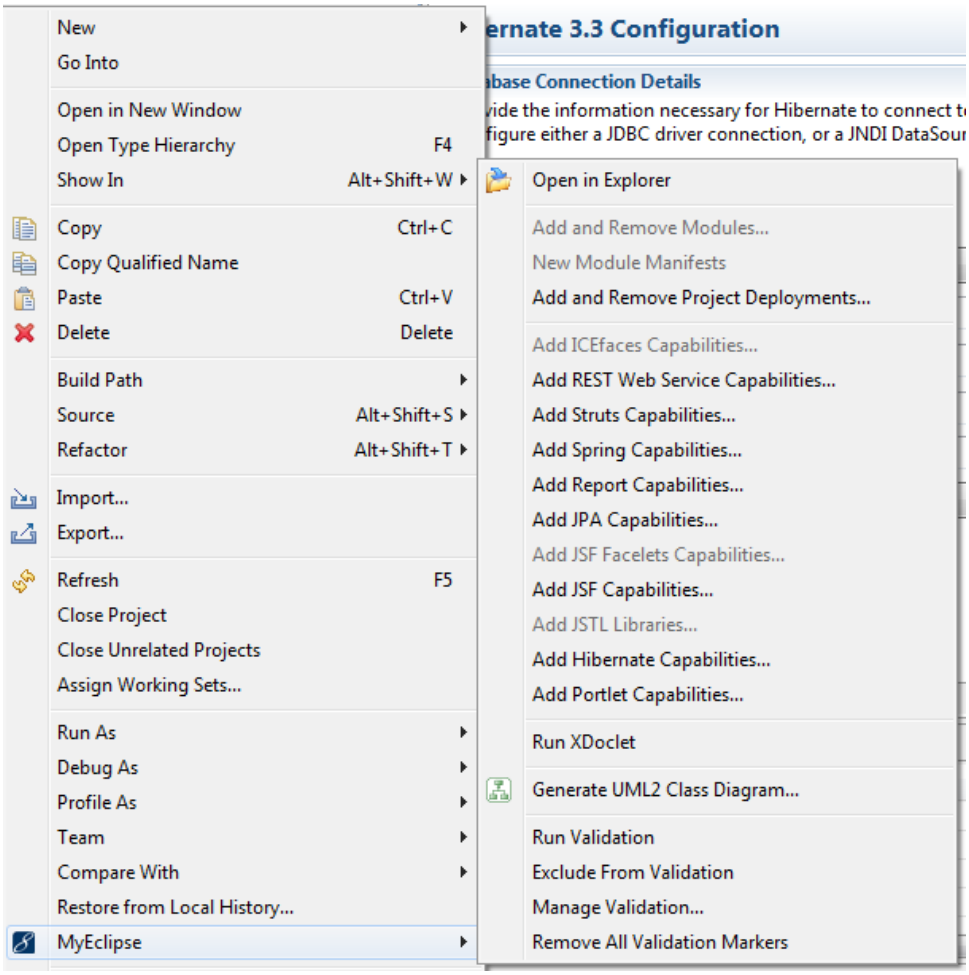

Figure 22. Import libraries for SSH

MyEclipse offers graphic integration for SSH, right click on project name, after select MyEclipse, a list of library will be showed, and then “Add Hibernate Capabilities”, “Add Spring Capabilities”, “Add Struts Capabilities”. All necessary libraries will be added to the project as the following shows.

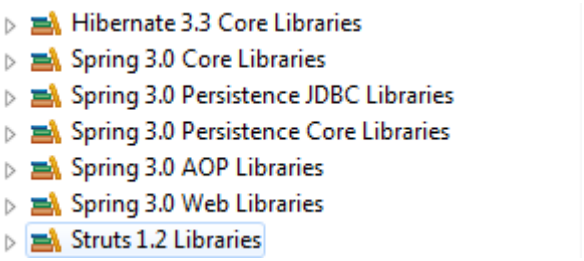

Figure 23. View of libraries' version

- Step 2: Build database connection

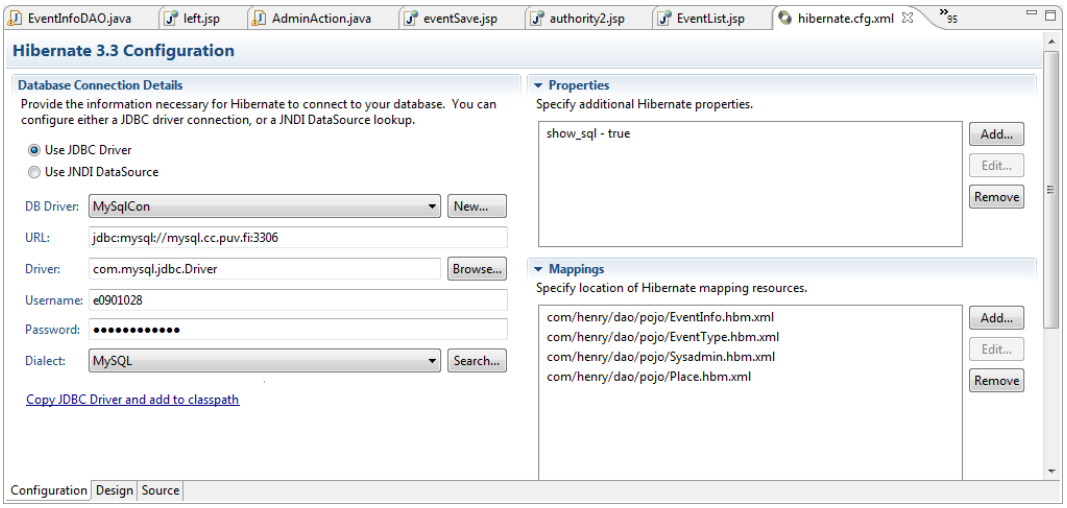

Figure 24. Hibernate configuration

MyEclipse offers GUI for hibernate's configuration as well, select configuration for MySql and necessary jars for support connection. Alternatively, this step can be done when import hibernate's jars.

- Step 3: Integrate Hibernate and Spring

Quite simple, in applicationContext.xml which generated automatically when import spring library, create new DataSauce as following shows

```
//connect to database
```

```
<bean id="dataSource"

class="org.apache.commons.dbcp.BasicDataSource">

<property name="driverClassName"

value="com.mysql.jdbc.Driver">

</property>

<property name="url"

value="jdbc:mysql://mysql.cc.puv.fi:3306">

</property>

<property name="username" value="e0901028"></property>

<property name="password" value="mNETTJqqBSKX"></property>

</bean>

//mapping with hibernate pojo, control injection

<bean id="sessionFactory"

class="org.springframework.orm.hibernate3.LocalSessionFactoryBean"
>

<property name="configLocation"

value="classpath:hibernate.cfg.xml">

</property>

</bean>
```

Snippet 7. Integration of Spring and Hibernate

- Step 4: Integrate Spring and Struts

In struts-config.xml which generated automatically when import struts libraries, input following code:

```
<controllerprocessorClass="org.springframework.web.struts.Delegati
ngRequestProcessor"></controller>

<message-resources parameter="com.henry.view.ApplicationResources"
/>
```

```
<plug-in
className="org.springframework.web.struts.ContextLoaderPlugIn">

<set-property
    property="contextConfigLocation"value="classpath:applicationCo
ntext.xml" />

</plug-in>
```

Snippet 8. Integration of Spring and Struts

Now SSH have been integrated successfully.

5.3.2 Event Management

Since event type, event place, and event management has the same hierarchy of implementation, here only event management is analyzed as instance.

A dispatch action named “EventAction” is defined as controller for event management, for navigating different functions in dispatch action, a “pointer” need to be set in struts-config.xml, in addition, mapping between forms and actions will be set at the same time.

- Step 1: Configure struts

```
<action

attribute="eventForm"  //build mapping between view and controller

name="eventForm"

parameter="method"// set the “pointer”

path="/event"

scope="request"

type="com.henry.view.action.EventAction"

validate="false">

<set-property property="cancellable" value="false" />
```

```
//set responsive pages for functions in dispatch action

<forward name="index" path="/foreground/EventList.jsp" />

<forward name="save" path="/background/message.jsp" />

<forward  name="findAllOK"
path="/background/eventInfo/eventList.jsp" />

<forward  name="initImgOK"
path="/background/eventInfo/imgUpdate.jsp" />

<forward  name="findById"
path="/background/eventInfo/eventUpdate.jsp" />

<forward  name="initOK"  path="/background/eventInfo/eventSave.jsp"
/>

<forward name="delete" path="/background/message.jsp" />

<forward name="foreShowDetail" path="/foreground/showDetail.jsp" />

<forward  name="showDetail"
path="/background/eventInfo/showDetail.jsp" />

<forward name="updateOK" path="/background/eventInfo/updateOK.jsp"
/>

</action>
```

Snippet 9. Configure struts-config.xml

- Step 2: Generate POJO and Spring DAO

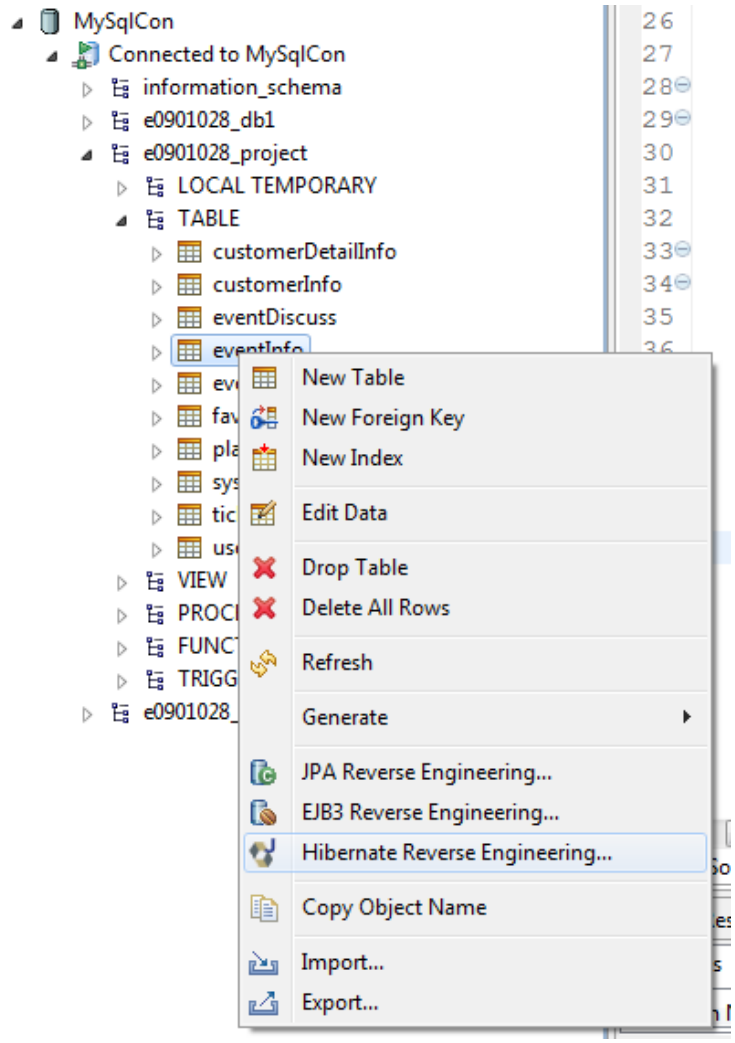

Figure 25. Generate POJO and Spring DAO

Hibernate Reverse Engineering will help to complete this procedure, necessary modification need to be made in some functions of Spring DAO generated automatically, detail code of DAO will be showed in the subsequent description.

- Step 3: Generate set functions of DAO used in dispatch action (controller)

```
public void setEventInfoDAO(IEventInfoDAO eventInfoDAO) {
this.eventInfoDAO = eventInfoDAO;
}
public void setEventTypeDAO(IEventTypeDAO eventTypeDAO) {
```

```
this.eventTypeDAO = eventTypeDAO;
}
public void setEventPlaceDAO(IEventPlaceDAO eventPlaceDAO) {
this.eventPlaceDAO = eventPlaceDAO;
}
```

Snippet 10. Generate set functions for DAO

In order to build mapping between controller and model layer which implements Dependency Injection, set functions of DAO enable a reference in Spring configure file appicationContext.xml

- Step 4: Set bean in Spring configure file.

```
<bean name="/event" class="com.henry.view.action.EventAction">
<property name="eventInfoDAO" ref="EventInfoDAO"></property>
<property name="eventTypeDAO" ref="EventTypeDAO"></property>
<property name="eventPlaceDAO" ref="PlaceDAO"></property>
</bean>
```

Snippet 11. Complete mapping between controller and model

Until now, preparation for implement functions is done.

5.3.2.1 Add Event

- Step 1: Initialize properties of event

Since event type and place will be selectable, their values needed to be gained before forwarding to add event page.

```
<td class="wr4"    width="120"><a
href="<%=request.getContextPath()%>/event.do?method=initSave"
target="mainFrame">Add EventInfo</a></td>
```

Here “method” is the “pointer” set in Snippet 9, this means invoke the function “initSave” in dispatch action “eventAction”.

- Step 2: Function initSave

```
Public ActionForward initSave(ActionMapping mapping, ActionForm form,

HttpServletRequest request, HttpServletResponse response){

List<EventType>eventTypes   =   eventTypeDAO.findAll();//query   all
evenTypes

List<Place> places = eventPlaceDAO.findAll();//query all eventPlaces

request.setAttribute("eventTypes", eventTypes);

request.setAttribute("places", places);

returnmapping.findForward("initOK"); //forward to evnetSave.jsp set
in snippet9

}
```

Snippet 12. InitSave function

Get the values of places and event types in advance then forward to save page.

- Step 3:eventSave.jsp

```
<formmethod="post"id="register"action="<%=request.getContextPath()
%>/event.do?method=save"  enctype="multipart/form-data">   //invoke
save function in eventAction, set form as uploading file form

<tr>

<td width="96" height="40" align="right">Event Type*</td>

<td height="40" colspan="3" >

<select name="eventTypeID" id="selectTypeId">
```

```
//get values transferred by action

<c:forEach items="${requestScope.eventTypes}" var="event">

<option
value="${event.eventTypeId}">${event.eventTypeName}</option>

</c:forEach>

</select>

</td>

</tr>

<tr>

<td width="96" height="40" align="right">Place*</td>

<td height="40" colspan="3" >

<select name="placeId" id="selectTypeId">

<c:forEach items="${requestScope.places}" var="place">

<option value="${place.placeId}">${place.placeName}</option>

</c:forEach>

</select>

</td>

</tr>
```

Snippet 13. Event save page

- Step 4: save() function

```
publicActionForward save(ActionMapping mapping, ActionForm form,

HttpServletRequest request, HttpServletResponse response) throws
ParseException {

//file upload

EventFormeventForm = (EventForm) form;

FormFileformFile = eventForm.getFile();

String filename = formFile.getFileName();

//set a unique new name for every upload file
```

```
String nfilename = MyTools.getNewFileName(filename);

if (formFile != null) {

// define path

String dir
   =this.getServlet().getServletContext().getRealPath("/upload");

OutputStreamfos = null;

try {

fos = new FileOutputStream(dir + "/" + nfilename);

fos.write(formFile.getFileData(), 0, formFile.getFileSize());

fos.flush();

} catch (Exception e) {

e.printStackTrace();

} finally {

try {

fos.close();

} catch (Exception e) {

e.printStackTrace();

}

}

}

//transform String to Date

SimpleDateFormat dateFormat = new SimpleDateFormat("yyyy/MM/dd");

Date start = dateFormat.parse(eventForm.getStart());

Date end = dateFormat.parse(eventForm.getEnd());

//construct object by values get from form

EventInfo eventInfo = new EventInfo(eventForm.getEventName(),
filename,nfilename,eventForm.getEventTypeID(),eventForm.getPlaceId
(),start,end,eventForm.getContext(),eventForm.getPrice());

String msg = "error";

String returnPath = "/background/eventInfo/eventSave.jsp";
```

```
try {

eventInfoDAO.save(eventInfo);  // save to database

msg = "succeed!";

returnPath = "/event.do?method=findAll";

} catch (Exception e) {

e.printStackTrace();

}

request.setAttribute("msg","Save "+ msg);

request.setAttribute("returnPath",returnPath);

returnmapping.findForward("save");

};
```

Snippet 14. Save function in action

5.3.2.2 Update Event

Pages forwarding and logic operations is implemented in the same way as the four steps above showed, however when users update an event, sometimes they do not update the image for the event, to solve this, update function is divided into two parts update and update image.

- Update EventInfo

```
Int eventId = eventForm.getEventID();

EventInfo event = eventInfoDAO.findById(eventId);

//image is static here, only properties except it can be updated

String filename = event.getImg();

String nfilename = event.getImg2();

EventInfo eventInfo = new EventInfo(eventForm.getEventName(),
filename,nfilename,eventForm.getEventTypeID(),eventForm.getPlaceId
(),start,end,eventForm.getContext(),eventForm.getPrice());

eventInfo.setEventId(eventId);
```

```
try {

eventInfoDAO.merge(eventInfo);  //update the event to database  }
catch (Exception e) {

e.printStackTrace();

}
```

Snippet 15. Update

- Update Event Image

```
EventFormeventForm = (EventForm) form;

FormFile formFile = eventForm.getFile();

String filename = formFile.getFileName(); //image is dynamic now

String nfilename = MyTools.getNewFileName(filename);

Int eventId = eventForm.getEventID();

EventInfo event = eventInfoDAO.findById(eventId);

//other properties become static

EventInfo   eventInfo   =   new   EventInfo(event.getEventName(),
filename,nfilename,event.getEventTypeId(),event.getPlaceId(),event
.getStartDate(),event.getEndDate(),event.getContext(),event.getPri
ce());

eventInfo.setEventId(eventId);

try {

eventInfoDAO.merge(eventInfo);  //update to database


} catch (Exception e) {

e.printStackTrace();

}
```

Snippet 16. Update image

5.3.2.3 Delete/Group Delete Event

Event can be delete individually or as a group, individual delete has the same logic as add event, here gives a description to group delete, mainly on Java Script.

```
$(function(){

Var deleteUId;

$( "#dialog-confirm" ).dialog({

autoOpen: false,

resizable: false,

height:140,

modal: true,

buttons: {

Confirm: function() {

//transfer deleteUId offered by #deleteQu or #delete-user to action
and execute delete

window.location="event.do?method=delete&eventId="+deleteUId;

},

"Cancel": function() {

$( this ).dialog( "close" );

}

}

});

//Individual delete

$('#deleteQu').live('click', function() {

$( "#dialog-confirm" ).attr("title", "Confirm Delete");

varmsg = "Confirm Delete["+ $(this).attr("title") + "]?";

$("#dialog-confirm p span").last().html(msg);

//assign title's value to deleteUId

deleteUId = $(this).attr("title");

//invoke dialog-comfirm action execute delete
```

```
$( "#dialog-confirm" ).dialog( "open" );

return false;

});

//select all

$('#selectAll').bind('click', function() {

$("#users tbodytr td input").attr("checked",
$(this).attr("checked"));

});

//Group Delete

$( "#delete-user" ).button().click(function() {

//record checked input parameter values in the table which has an Id
as”users”

varuserIds = $("#users tbodytr td input:checked");

deleteUId="";

if(userIds.length == 0) {

$( "#dialog" ).dialog( "open" );

return false;

}

varmsg = "";

//Assign recorded userIds to deleteUId

userIds.each(function(){

deleteUId += (this.value + "&eventId=");

msg += this.value + "  ";

});

$( "#dialog-confirm" ).attr("title", "Confirm Delete");

varmsg= "Confirm delete selected?"; ["+ msg + "]

$("#dialog-confirm p span").last().html(msg);

//invoke dialog-confirm functionexecute delete by using parameter
deleteUId

$( "#dialog-confirm" ).dialog( "open" );
```

```
});
$.fx.speeds._default = 1000;
$( "#dialog" ).dialog({
autoOpen: false,
show: "blind",
hide: "explode"
});
});
</script>
```

Snippet 17. Java script enable group delete

Code calls js functions in jsp table:

- Table with id “users”

```
<table id="users" class="ui-widget ui-widget-content">
```

- Call js function selectAll

```
<th><input type="checkBox" id="selectAll"/></th>
```

- Individual Delete

```
<td><a id="deleteQu"
href="${pageContext.request.contextPath}/event.do?method=delete&ev
entId=${event.eventId}"
< -- #deleteQu get deleteUId from here -- >
title="${event.eventId}">Delete</a></td>
```

- Group Delete

```
<td colspan="1">
<button id="delete-user">Delete Selected</button>
</td>
```

- Modification in EventInfoDAO

```
public void delete(String[] ids){
int length = ids.length;
if (length != 1) length--;
for (inti = 0; i< length; i++) {
try {
this.delete(Integer.parseInt(ids[i]));
} catch (Exception e)
}
}
}
```

The code above is added to EventInfoDAO to enable group delete since the original function can only delete id one by one.

5.3.2.4 Query Event/Page Division

This function is mainly implemented by Hibernate query language, HQL. In HQL, attribute in query sentence must be as the same as POJO's attribute. Following is the modification to EventInfoDAO.

- Set the number of events displayed every page

```
public List getNowPageData(intnowPage, intpageSize) {
Configuration config = new Configuration().configure();
SessionFactorysf = config.buildSessionFactory();
org.hibernate.classic.Session session =  sf.openSession();
String hql = "from EventInfo order by startDatedesc";
Query query = session.createQuery(hql);
//Implement the number of events displayed every page
query.setFirstResult((nowPage-1)*pageSize);
query.setMaxResults(pageSize);
List eventInfos = query.list();
session.close();
returneventInfos;
}
```

Snippet 18. Control the number of events display every page

- Count the pages in total according to database record:

```
publicintgetPageCount(intpageSize) {
Configuration config = new Configuration().configure();
SessionFactory sf = config.buildSessionFactory();
org.hibernate.classic.Session session =  sf.openSession();
intpageCount = 0;
String hql = "select count(*) from  EventInfo";
Query query = session.createQuery(hql);
List list=query.list();
Number num=(Number)list.get(0);
introwCount = num.intValue();
pageCount = rowCount / pageSize;
```

```
if (rowCount % pageSize != 0) {
pageCount ++;
}
returnpageCount;
}
```

Snippet 19. Get pages in total

- Get the number of records

```
publicintrowCount(){
Configuration config = new Configuration().configure();
SessionFactory sf = config.buildSessionFactory();
org.hibernate.classic.Session session =  sf.openSession();
String hql = "select count(*) from  EventInfo";
Query query = session.createQuery(hql);
List list=query.list();
Number num=(Number)list.get(0);
introwCount = num.intValue();
return rowCount;
}
```

Snippet 20. Get number of records

- Set and get parameter values in controller

```
Public ActionForward findAll(ActionMapping mapping, ActionForm form,
HttpServletRequest request, HttpServletResponse response){
```

```
// set default value for pageSize, nowPage and pageCount used in DAO

intpageSize  = 10;

intnowPage   = 1;

intpageCount = 0;

introwCount  = 0;

String strNowPage = request.getParameter("nowPage");

if (strNowPage == null) {

strNowPage = "1";

}

String strPageSize = request.getParameter("pageSize");

if (strPageSize == null) {

strPageSize = "10";

}

pageSize =Integer.parseInt(strPageSize);

nowPage = Integer.parseInt(strNowPage);

pageCount= eventInfoDAO.getPageCount(pageSize);//showed in snippet
19

rowCount = eventInfoDAO.rowCount();//showed insnippet 20

List<EventInfo>eventInfos  =  eventInfoDAO.getNowPageData(nowPage,
pageSize);//showed in snippet 17

List<EventType>eventTypes = eventTypeDAO.findAll();

List<Place> places = eventPlaceDAO.findAll();

//shows date and weekday on query page

Date now = new Date();

String today = MyTools.getDate(now);

String weekday = MyTools.getWeekday(now);

//set attributes' values

request.setAttribute("eventTypes", eventTypes);

request.setAttribute("eventInfos", eventInfos);

request.setAttribute("places", places);
```

```
request.setAttribute("nowPage", nowPage);

request.setAttribute("pageSize", pageSize);

request.setAttribute("pageCount", pageCount);

request.setAttribute("rowCount", rowCount);

request.setAttribute("today", today);

request.setAttribute("weekday", weekday);
```

Snippet 21. FindAll function

- Display attributes on jsp

Show current page:

```
Page<fontcolor="green">${requestScope.nowPage}</font> Now&nbs
p; 
```

Show number of pages in total

```
Total <font color="red">${pageCount}</font> Page(s)
```

Show number of events in total

```
<font color="red">${requestScope.rowCount}</font> event(s)
```

- Some implementations of javascript

```
//go to previous page

<a  id="upPage"   href="#"
onclick="goNowPage('${nowPage-1}')">Previous  </a>

// go to a selected page

<select id="selectNowPage" onchange="goNowPage(this.value)">

<c:forEach begin="1" end="${pageCount}" varStatus="sta" >

<option value="${sta.count}" >

 Page ${sta.count}
```

```
</option>

</c:forEach>

</select>

//set number of event displayed(pagesize) every page

<select    id="nowPageSize" onchange="findAll(this.value)">

<c:forEach begin="1" end="15" varStatus="sta">

<option value="${sta.count}">

${sta.count } event(s) per page

</option>

</c:forEach>

</select>
```

Snippet 22. Call java script functions

Java script code called above (findAll, goNowPage)

```
varpageSize = '${pageSize}';

functionfindAll(pageSize) {

//transfer selected pageSize values to action

Varurl="${pageContext.request.contextPath}/event.do?method=findAll
&pageSize=" + pageSize ;

window.location = url;

}

functiongoNowPage(nowPage) {

//transfer selected nowPage, pageSize values to action

Var url =
"${pageContext.request.contextPath}/event.do?method=findAll&nowPag
e=" + nowPage +"&pageSize=" + pageSize;

window.location = url;}
```

Snippet 23. Java script functions

The idea in Snippet 22 and 23 implement goes to next, previous, home, end page, goes to any selected page, and sets the number of events displayed every page together.

5.3.3 Administrator Login

Add, update or delete/group delete administrator they all have the same logic as event management. Hereby is given an emphasis on administrator's login.

```
Public ActionForward login(ActionMapping mapping, ActionForm form,

HttpServletRequest request, HttpServletResponse response) {

AdminFormadminForm = (AdminForm) form;

//check if username exists

List<Sysadmin>    adminInfos                                          =
sysadminDAO.findByAdminName(adminForm.getUserName());

String msg = "";

String returnPath = "/background/admin/adminLogin.jsp";

if((null !=adminInfos)&&(adminInfos.size()>0)){

Sysadmin admin=(Sysadmin)adminInfos.get(0);

// match the username and password

if(admin.getPwd().equals(adminForm.getPwd())){

HttpSession session = request.getSession();

session.setAttribute("adminInfo", admin);

Date now = new Date();

String today = MyTools.getDate(now);

String weekday = MyTools.getWeekday(now);

request.setAttribute("today", today);

request.setAttribute("weekday", weekday);

//login ok forward to index.jsp

returnmapping.findForward("LoginOK");
```

```
}
//error message if username and password doesn’t match
else{
msg="error,incorrect username or password ";
returnPath = "/background/admin/adminLogin.jsp";
request.setAttribute("msg","Login "+ msg);
request.setAttribute("returnPath",returnPath);
returnmapping.findForward("LoginError");
}
}
//error message if username doesn’t exist
else{
msg="error,username is not found";
returnPath = "/background/admin/adminLogin.jsp";
request.setAttribute("msg","Login "+ msg);
request.setAttribute("returnPath",returnPath);
returnmapping.findForward("LoginError");
}
}
```

Snippet 24. Login action

5.3.4 Show Only Today’s Event

Core logic is the function in EventInfoDAO:

```
publicListgetTodayData(intnowPage, intpageSize,String today) {
Configuration config = new Configuration().configure();
SessionFactorysf = config.buildSessionFactory();
org.hibernate.classic.Session session =  sf.openSession();
//make comparison with today which transferred in by action
```

```
String hql = "from EventInfo where startDate<= "+today+"and
endDate>="+today;

Query query = session.createQuery(hql);

//page division mentioned above

query.setFirstResult((nowPage-1)*pageSize);

query.setMaxResults(pageSize);

List eventInfos = query.list();

session.close();

return eventInfos;

}
```

Snippet 25. Show only today's event

5.3.5 Search Function

User can search event by one category or group categories, core logic of this implementation also count on hql in eventInfoDAO

```
publicList search(String eventName,

String eventTypeId, String placeId, String startDate, String endDate,

String price,String price2,int nowPage, intpageSize) throws
ParseException{

Configuration config = new Configuration().configure();

SessionFactorysf = config.buildSessionFactory();

org.hibernate.classic.Session session =  sf.openSession();

String hql = "from EventInfo where 1=1 ";  ;

//search by name

if (eventName!=null&&!"".equals(eventName)) {

hql = hql + " and eventName like '%"+ eventName+"%' " ;

}

//search by event type
```

```
if (eventTypeId!=null&&!"".equals(eventTypeId)) {

//when select event type as 'All'

if (eventTypeId.equals("0")) {

hql = hql + " and eventTypeId<> 0 ";

}

// when select a certain type

else{

hql = hql + " and eventTypeId = "+ eventTypeId;

}

}

//search by event place

if (placeId!=null&&!"".equals(placeId)) {

//when select place as all

if (placeId.equals("0")) {

hql = hql + " and placeId<> 0 ";

}

// when select a certain place

else{

hql = hql + " and placeId = "+placeId;

}

}

//search by date range

if(startDate!=null&&!"".equals(startDate)&&endDate!=null&&!"".equa
ls(endDate)) {

Date start = MyTools.tDate(startDate);

Date end = MyTools.tDate(endDate);

//transfer format of date in order to make comparison

String strStart = MyTools.getCompare(start);

String strEnd = MyTools.getCompare(end);
```

```
hql = hql + " and startDate<= "+ strEnd +" and endDate>= "+ strStart;

}

//search by price range

if
(price!=null&&!"".equals(price)&&price2!=null&&!"".equals(price2))
{

hql = hql + " and price >= "+ price +" and price <= "+price2;

}

Query query = session.createQuery(hql);

//page divison mentioned above

query.setFirstResult((nowPage-1)*pageSize);

query.setMaxResults(pageSize);

List<EventInfo>eventInfos=query.list();

session.close();

return eventInfos;

}
```

Snippet 26. Search function

5.3.6 Show Current Date, Weekday

- Code in action:

```
Datenow = newDate();

String today = MyTools.getDate(now);

String weekday = MyTools.getWeekday(now);

request.setAttribute("today", today);

request.setAttribute("weekday", weekday);
```

- Code in Mytools.java

```
//get current date
```

```
Public static String getDate(Date today){
SimpleDateFormat dateFormat = new SimpleDateFormat("yyyy/MM/dd");
String date = dateFormat.format(today);
return date;
}
//get current weekday
Public static String getWeekday(Date today){
String day = "";
switch (today.getDay()) {
case 0:
day = "Sunday";
break;
case 1:
day = "Monday";
break;
case 2:
day = "Tuesday";
break;
case 3:
day = "Wednesday";
break;
case 4:
day = "Thursday";
break;
case 5:
day = "Friday";
break;
case 6:
day = "Saturday";
```

```
break;
}
return day;
}
```

Snippet 27. Get current date and weekday

- Display on jsp

```
<font  size=4 color=red>Today  is
${requestScope.today}  ${requestScope.weekday}</font>
```

Above is a complete loop of show current date and weekday.

6 TESTING

Test cases here are mainly designed as test the functions with illegal values or updating without new information.

6.1 Test Login

- Input username does not exist

Result:

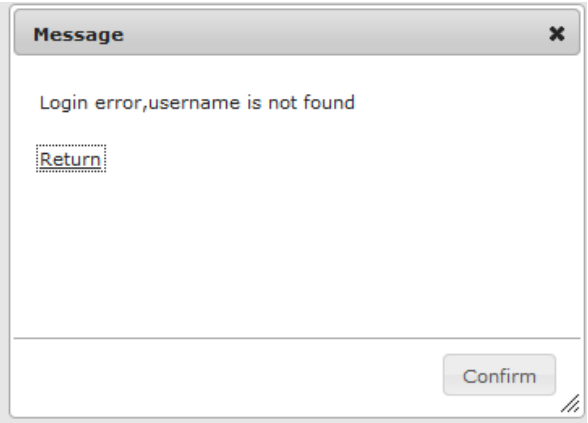

Figure 26. Test login result 1

- Input wrong password

Result:

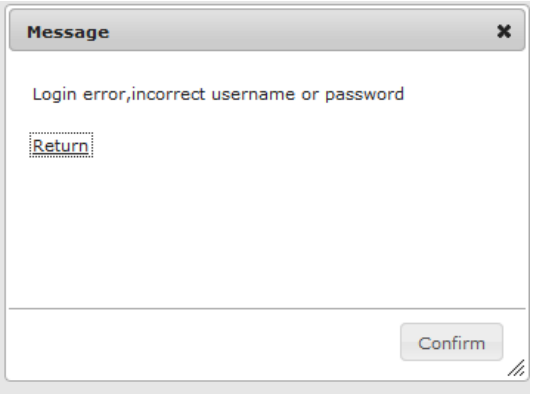

Figure 27. Test login result 2

- Input null to loginform

Result:

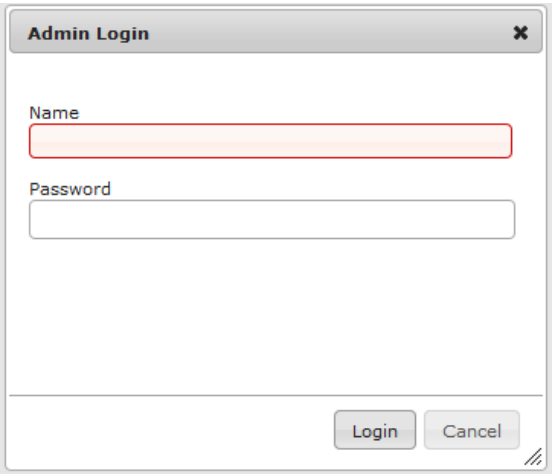

Figure 28. Test login result 3

Can not submit to next step.

6.2 Test Admin Management

- Add username already exist

Result:

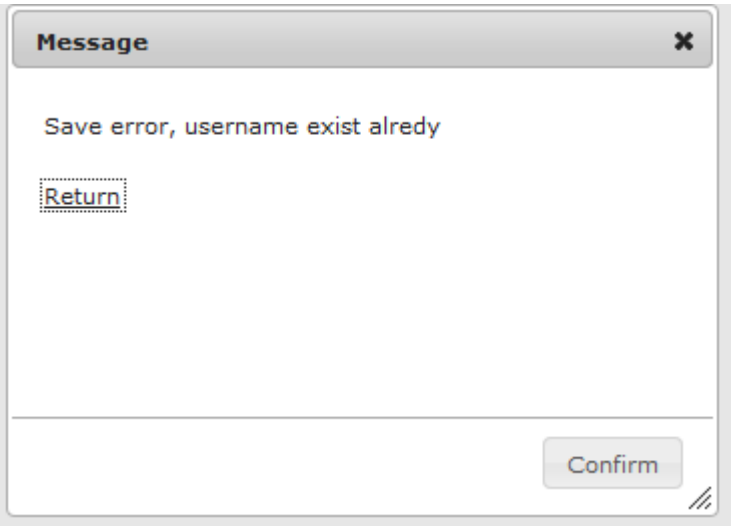

Figure 29. Test add user 1

- Add admin with illegal length username or password

Result:

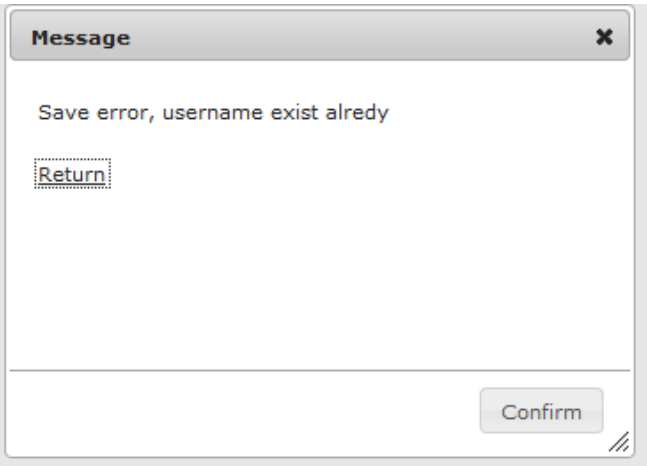

Figure 30. Test add user 2

This is an error, should set a correct error feedback for this case.

6.3 Test Event Management

- Save event without uploading image

Result:

Figure 31. Test add event 1

Not user friendly error feedback, should add an exception tips when nothing is being uploaded.

- Save event with illegal length parameters

Result:

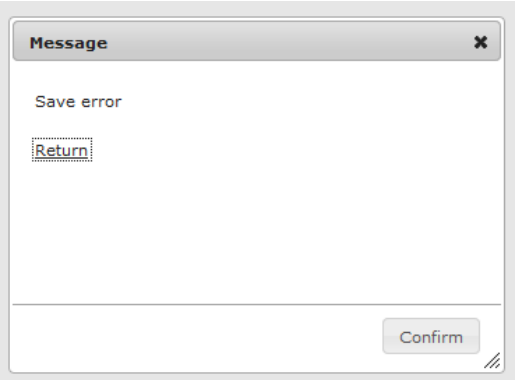

Figure 32. Test add event 2

- Save event with a null date

Result:

Figure 33. Test add event 3

Should add an error message in case input null date .

- Save event with a null event name

Result

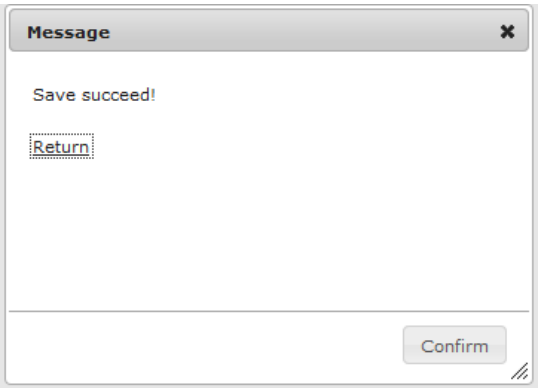

Figure 34. Test add event 4

This is a bug, event name cannot be null, and a validation should be added to action or form directly.

- Update event without new information

Result:

Figure 35. Test update event

Another error, shouldn't execute update without new information.

6.4 Test Search

- Search by single category

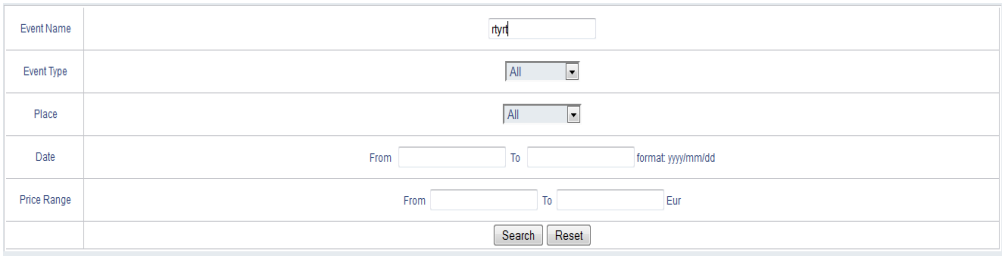

Figure 36. Test search case 1

Result:

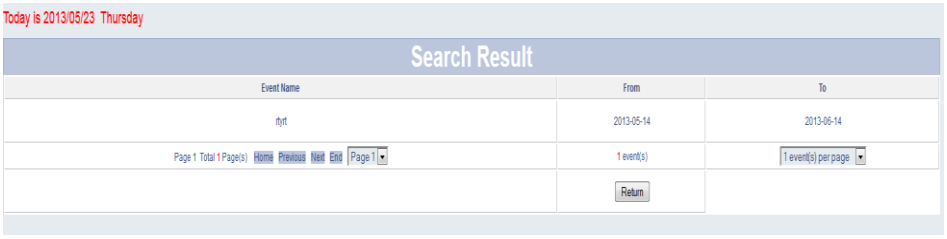

Figure 37. Test search result 1

- Search by multi categories

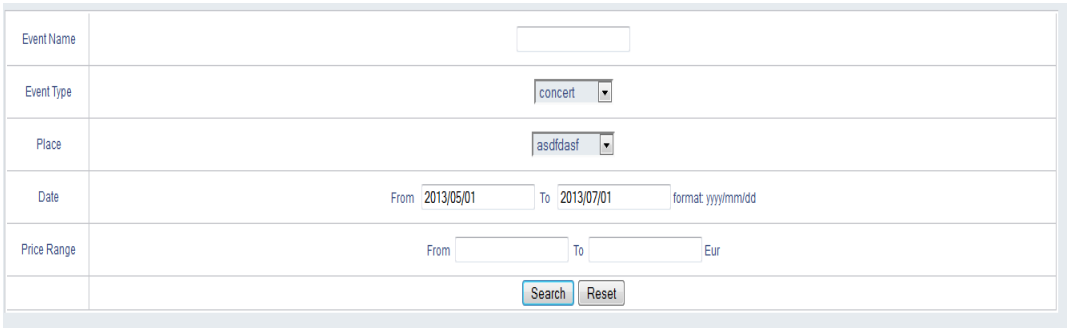

Figure 38. Test search case 2

Result

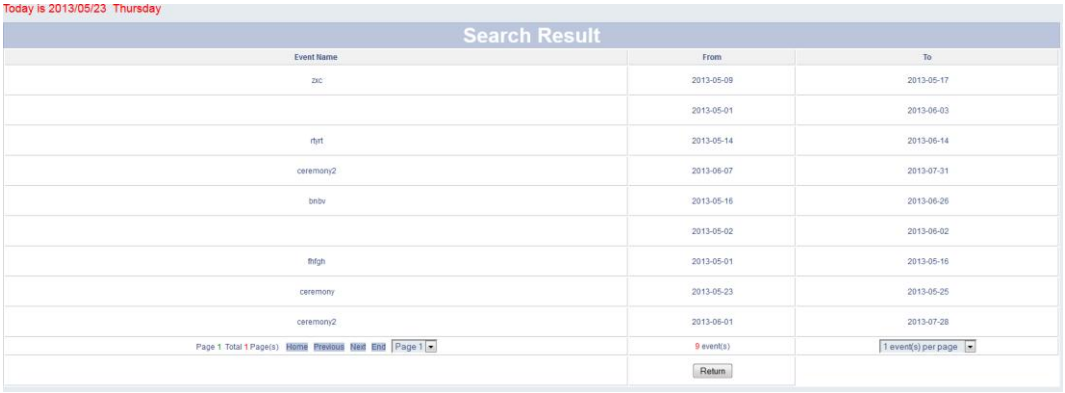

Figure 39. Test search result 2

Search function is implemented properly.

6.5 Possible Improvements

According to the test, handle of null value exception and some error feedback need to be expanded in action.

Alternatively, struts validate method or ajax can be added to form to avoid input illegal values.

7 CONCLUSIONS

The objective of this project was to implement the Event Calendar application with Spring, Struts and Hibernate (SSH) frameworks.

So far, logic of solutions to core functions is found and implemented except for several error message pages and the handle of null value exception remained; additionally the application has been deployed to VAMK's server already.

The administrator can add, update, delete and query information of events and users through the event calendar management platform now. For public page, users will have a view of events running on today, they can also search event by different categories, such as event's date, price, place, name, etc.

Developing the application made it possible to learn and practice the whole processes of agile development with SSH as well as the concepts in Software Engineering, such as UML, requirement analysis and standard of documental work.

In addition, skills of operating mySql and Java programming particularly for debugging and figuring out problems have been enhanced during the process.

What is more, front-end design for the project enables the approach to the technology in Photoshop, Java Script (jQuery UI) and CSS style sheet.

Main Challenges in Developing:

- Get knowledge preparation for the project

  Lots of videos have been watched to practice SSH project and get familiar with it.

- Figuring out logic to implement functions

Database design, idea of MVC, different data types transfer between java and database, front-end display, etc. are major issues in this part. Java programming books, the Internet and hard -working may help to solve them.

In particular, Date type process is one of the most confused problems. First of all, functions of formatting variables in Date type are obstacles unless format() and parse()figured out. Then date format for the comparison in database is another issue. Numbers of samples on the Internet have been viewed and plenty of tries have been done to solve the problem.

- Debug program

It is frequent that Http 404 or 500 errors appear after a function complete and being test. Never get down and be patient to the errors, learn to use debug mode in programming editor, search errors on the Internet can be the keys to solve the errors.

- Loneliness and fatigue in research and development

Set a target for every day, try to be self-controlled and disciplined is the key to solve the problem.

7.1 Future Works

As mentioned in test, a validation for value length and null value needs to be developed; Events date can be accurate to date, hour and minute; the purchase system and comment board can be developed for event; at last, some decoration for front-end display is available.

REFERENCES

/1/Struts Framework Overview-Struts Framework Overview (2013). [WWW]. [referred 5.5.2013] Available on the Internet:

<URL: http://struts.apache.org/>

/2/ Spring Framework Overview-Spring Framework Overview (2013). [WWW].

[referred 5.5.2013] Available on the Internet:

<URL: http://www.springsource.org/spring-framework>

/3/ Spring Framework Security-Spring Framework Overview (2013). [WWW].

[referred 5.5.2013] Available on the Internet:

<URL: http://www.springsource.org/spring-security>

/4/ Spring Framework Integration-Spring Framework Overview (2013). [WWW].

[referred 5.5.2013] Available on the Internet:

<URL: http://www.springsource.org/spring-integration>

/5/ Spring Framework Data-Spring Framework Overview (2013). [WWW].

[referred 5.5.2013] Available on the Internet:

<URL: http://www.springsource.org/spring-data>

/6/ Hibernate Framework Overview-Hibernate Framework Overview (2013). [WWW]. [referred 5.5.2013] Availableon the Internet:

<URL: http://www.ohloh.net/p/hibernate>

/7/ Hibernate Framework Overview- Relational Persistence for Java and .NET (2013). [WWW]. [referred 5.5.2013] Available on the Internet:

<URL: http://www.hibernate.org/>

/8/ jQueryUI Overview-jQueryUI Overview (2013). [WWW]. [referred 5.5.2013] Availableon the Internet:

<URL: http://jqueryui.com/>