

Eagle flights @ Lexicon

By Marcus Bergström & Björn Larsson

Liljeholmen 30/5 2016

Background

- We got our spec and had precious little time to execute.
 - But that's how it usually is...
- We took some inspiration from one of the slides by the guru "Speedy Gonzales"!
- And we both like gaming 😊
- This means we started with outlining how the system should behave and which features it should offer, not bare metal...
- So our first stop was ->

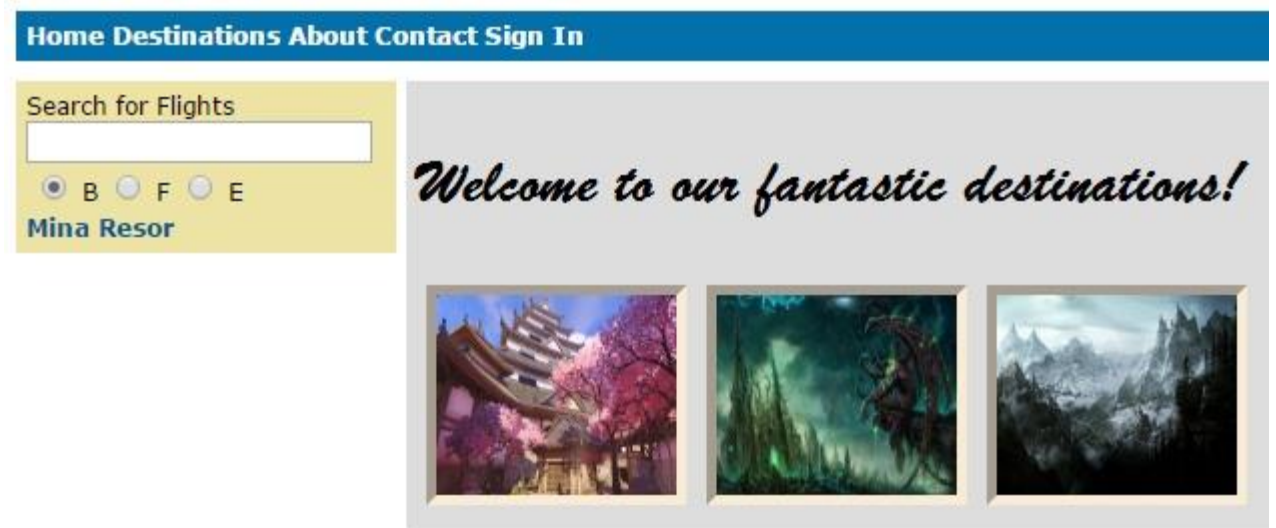
WORLD OF GAMING LOCATIONS



How to run the project & where do we start.

PLANNING & EXECUTION

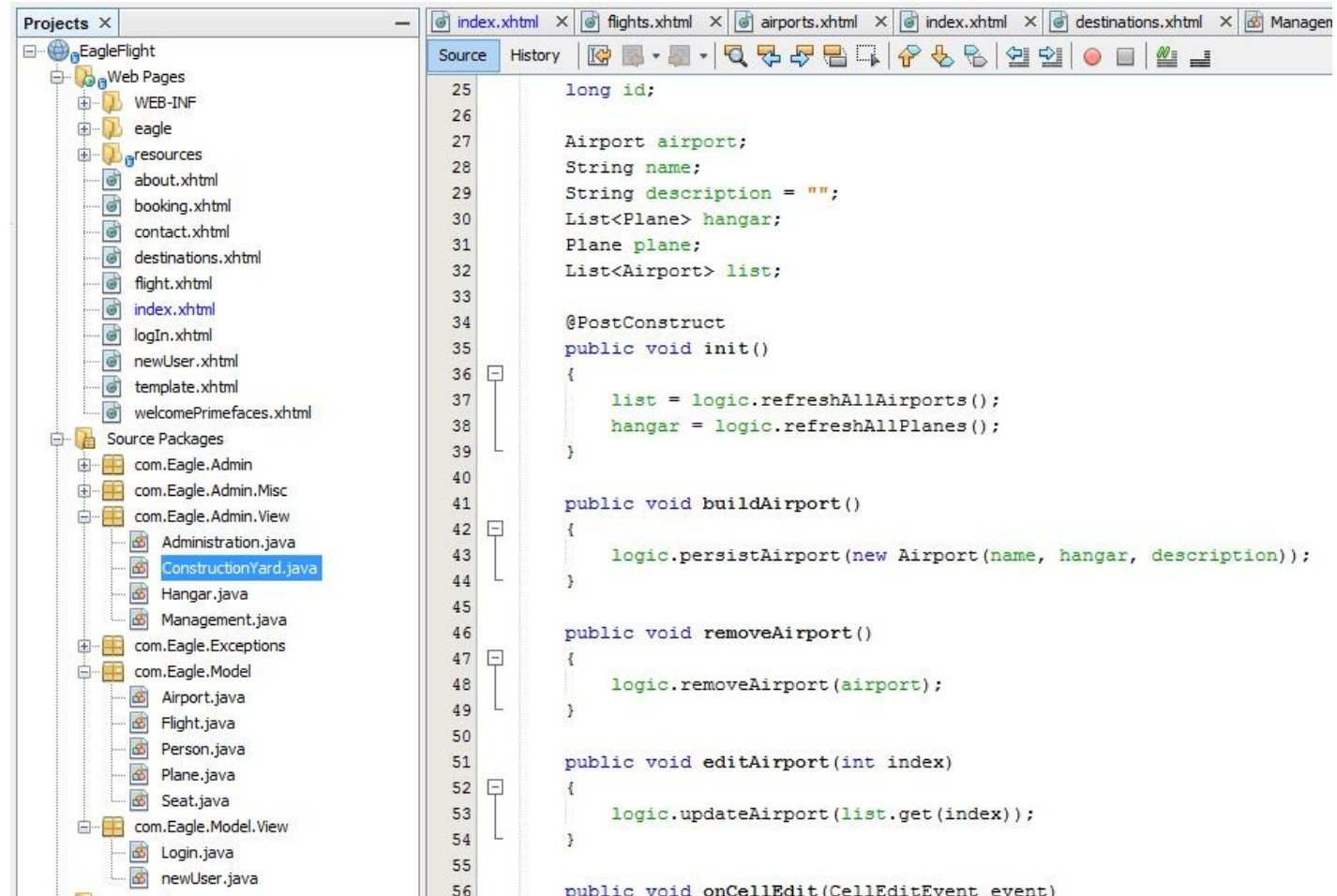
- Use Github as source repository for the project.
- Start by defining the GUI
 - One main page for end users and one admin page
 - Top & left menu for end-users
 - Only top menu for admin
 - Use JSF template for building site skeleton.
- And after that start with the Java coding and model...



Coding & structure

- Split packages into view, model & logic, but also separate view / logic packages for admin & end-user.
- Decided to go for an AJAX approach in updating pages.
- Simple POJOs for Airport, Flight, Person, Plane & Seat.
- Using Constructionyard & Hangar beans for building airports and planes.
- Administration & Management beans for flights and users.

EXECUTION PART II



The screenshot shows an IDE with a project named 'EagleFlight'. The 'Projects' pane on the left displays the following structure:

- Web Pages
 - WEB-INF
 - eagle
 - resources
 - about.xhtml
 - booking.xhtml
 - contact.xhtml
 - destinations.xhtml
 - flight.xhtml
 - index.xhtml
 - login.xhtml
 - newUser.xhtml
 - template.xhtml
 - welcomePrimefaces.xhtml
- Source Packages
 - com.Eagle.Admin
 - Administration.java
 - ConstructionYard.java
 - Hangar.java
 - Management.java
 - com.Eagle.Exceptions
 - com.Eagle.Model
 - Airport.java
 - Flight.java
 - Person.java
 - Plane.java
 - Seat.java
 - com.Eagle.Model.View
 - Login.java
 - newUser.java

The main editor window shows the code for 'index.xhtml', which is a Java class. The code is as follows:

```
25     long id;
26
27     Airport airport;
28     String name;
29     String description = "";
30     List<Plane> hangar;
31     Plane plane;
32     List<Airport> list;
33
34     @PostConstruct
35     public void init()
36     {
37         list = logic.refreshAllAirports();
38         hangar = logic.refreshAllPlanes();
39     }
40
41     public void buildAirport()
42     {
43         logic.persistAirport(new Airport(name, hangar, description));
44     }
45
46     public void removeAirport()
47     {
48         logic.removeAirport(airport);
49     }
50
51     public void editAirport(int index)
52     {
53         logic.updateAirport(list.get(index));
54     }
55
56     public void onCellEdit(CellEditEvent event)
```

RESULT & FINDINGS

Our result

- Learning the quirks of JSF proved a hurdle, but:
 - Primeface coming very handy with builtin support for datatables (AJAX).
 - Glassfish stable AS, even if error printouts hard to digest %-)
- Github made collaboration easier 😊
- Coding proved to be an adventure...
 - We wanted to build an advanced platform, equals timeconsuming!
 - So we would have loved to have some more time 😊
- We can create airplanes, airports and flights, developed in our first Sprints. And some marketing also.

