Monty Hall

Documentation

Introduction

This short document to give a short description about the task (building quick simulation for the well know “Monty Hall” puzzle). The application is build based on the requirements stated by “Com Hem”:

“*The purpose of this test is to see how you design and write your code given a problem. Try to use right level of OOP. The problem is a well-known puzzle that you can read more about at*[*http://en.wikipedia.org/wiki/Monty_Hall_problem*](http://en.wikipedia.org/wiki/Monty_Hall_problem) *. Save reading the link if you do not want to know the answer until you have solved the problem.*

Problem description:

Assume that you are attending a TV show where you can win money by picking the right box. The game show host shows you three boxes explaining that the money is in one of the boxes. He asks you to pick one of them without opening it. After you have picked one of the boxes he opens one of the other two boxes which is empty. Now he turns to you and asks, do you want to change your mind, picking the remaining box?

Your task:

Write a program in Java randomly simulating this event over and over again in the quest of answering following question. Do I stand a better chance to win if I change my mind? Maven should be used. There is no right or wrong solution, but the way you build the application, will tell us how you think and especially how you code. Send us your code in a zipped but rename the file removing the ‘.zip’ suffix. Else the spam filters may catch the e-mail.

Do not hesitate to ask questions if something is unclear. Contact: Koen De Smedt, koen.desmedt@comhem.com”

The next few sections will provide more description about the Java project.

Technologies used:

The following frameworks, libraries, and build tools have been used to build the project:

Spring MVC (4.0.2.RELEASE): MVC framework to build the front layer right before the final views.

JSP & JSTL (1.2): Used to render the model objects returned from Spring MVC.

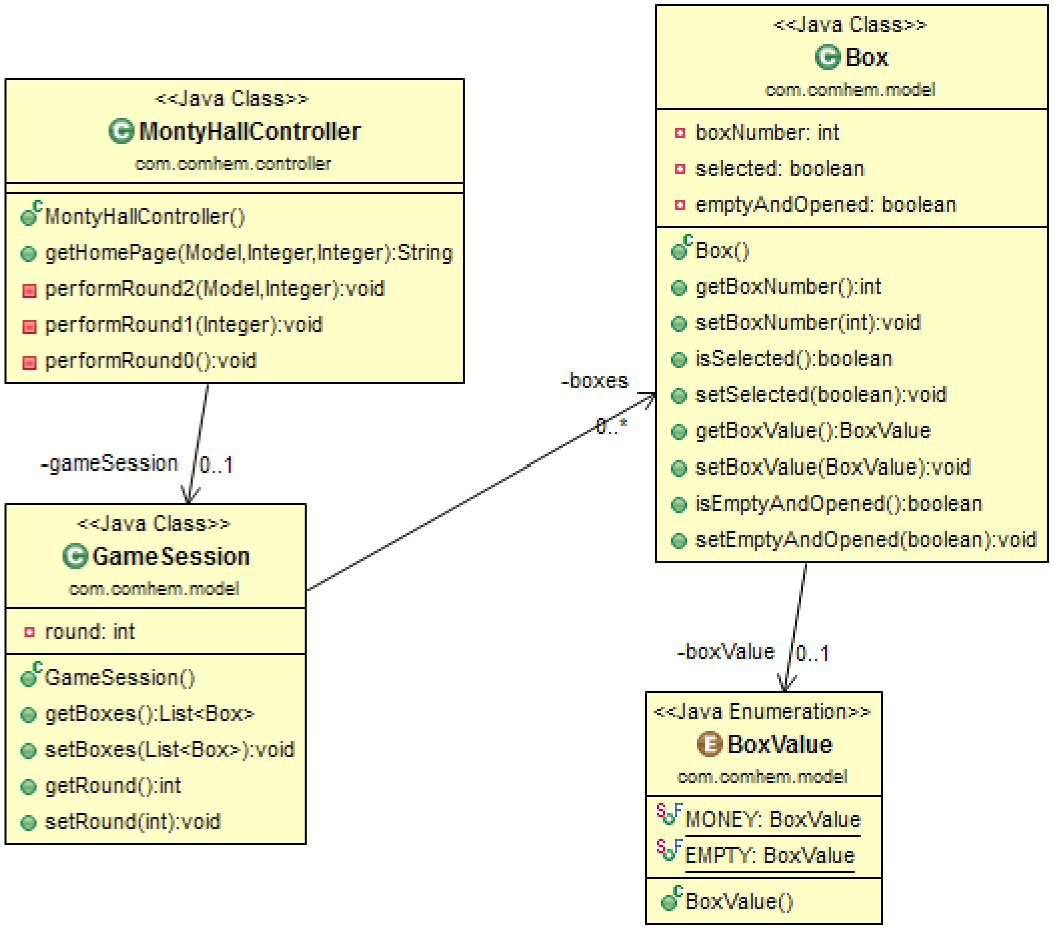
Maven: Used to build and resolve the project dependencies.

Oravle WebLogic Application Server (10.3.5): Used to publish the final project artifact (war file).

Eclipse (Kepler Service Release 1): used as an IDE that provides flexibility to use different plugins.

Layers Design:

The following class diagram describes the classes used to build the application:



Source code

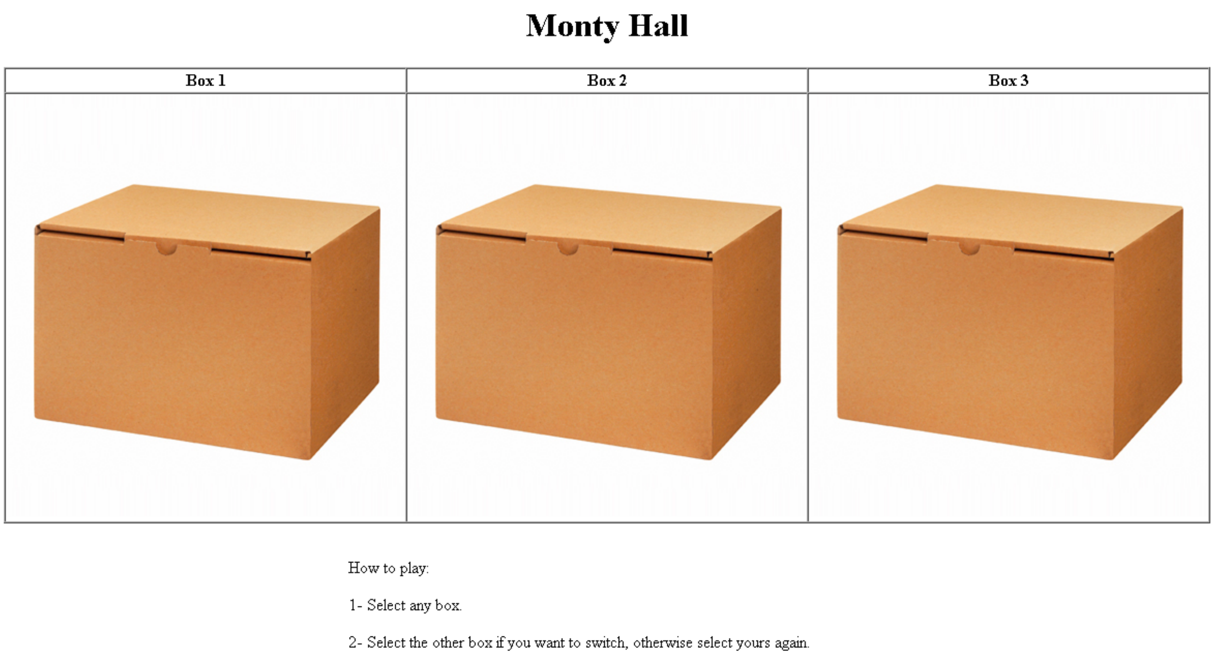
Eclipse project could be checked out from the public “github” repository:

https://github.com/Mohammed-Rady/TestRepo.git

Screenshots

The below is some screenshots from the application:

The home page of the application will look like:



Monty Hall

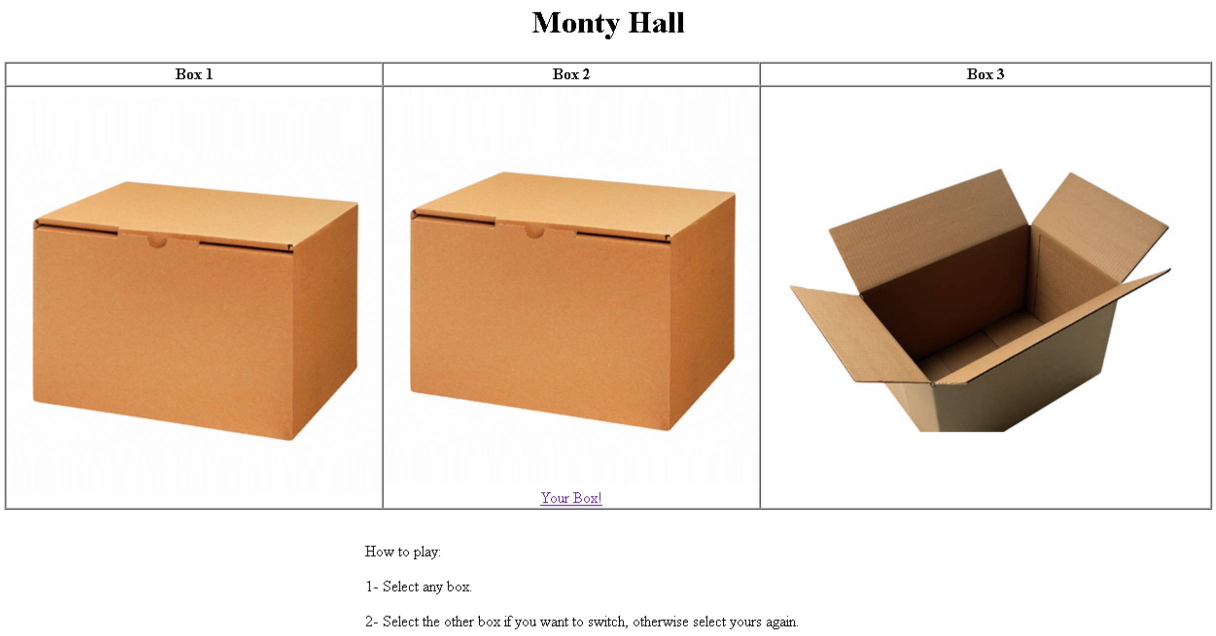
Box 1	Box 2	Box 3

How to play:

1- Select any box.

2- Select the other box if you want to switch, otherwise select yours again.

As per shown in the play instructions, the prompt is to choose a box that you inclined to find the money in:



Monty Hall

Box 1	Box 2	Box 3
	Your Box!	

How to play:

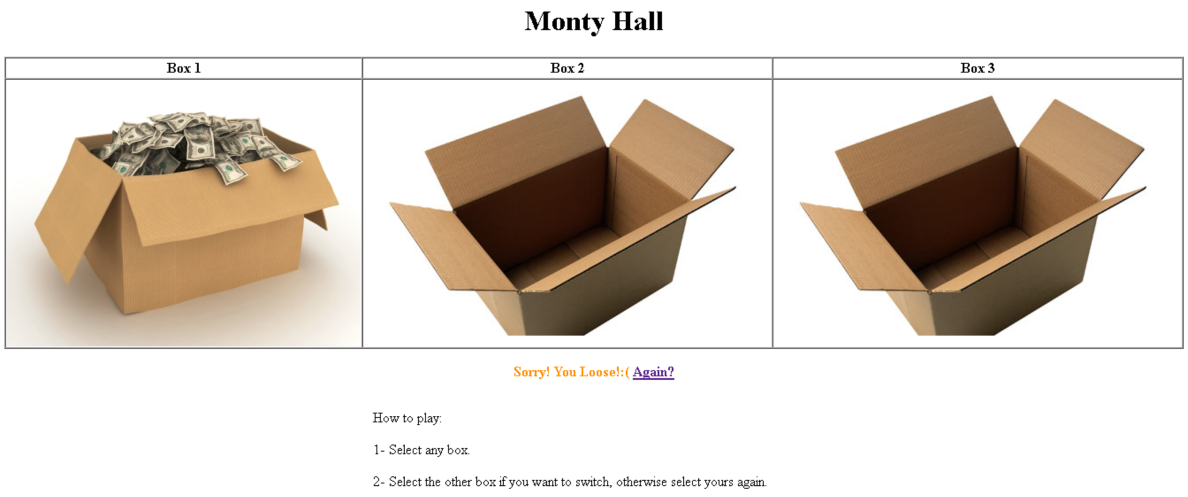
1- Select any box.

2- Select the other box if you want to switch, otherwise select yours again.

The illustration here has selected the second box as it might has the money, in return the computer opens one of the boxes which should be empty one.

As per the play instruction, the user is prompted again either to keep his first choice of the box or to choose the other one which is not yet opened.

The illustration here will maintain the first choice as is without changing to the second box, and let’s see the result:



Monty Hall

Box 1	Box 2	Box 3

Sorry! You Loose!:(Again?

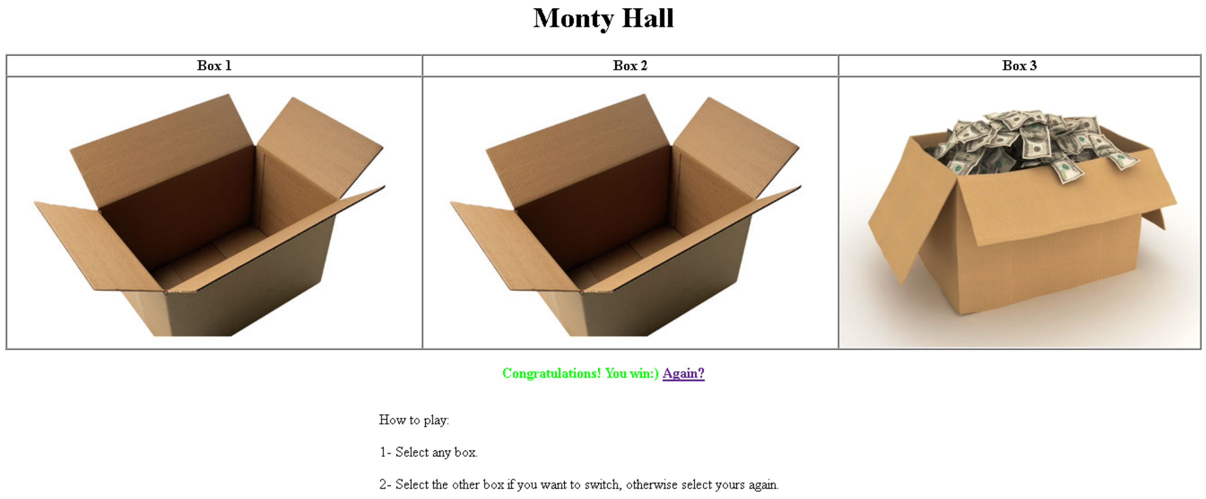
How to play:

1- Select any box.

2- Select the other box if you want to switch, otherwise select yours again.

Unfortunately, the selected box did not contain any money, and the second un-opened box did.

When we repeated the game again but changed the strategy this time to swap the box in the second round, we win!



Monty Hall

Box 1	Box 2	Box 3

Congratulations! You win:) Again?

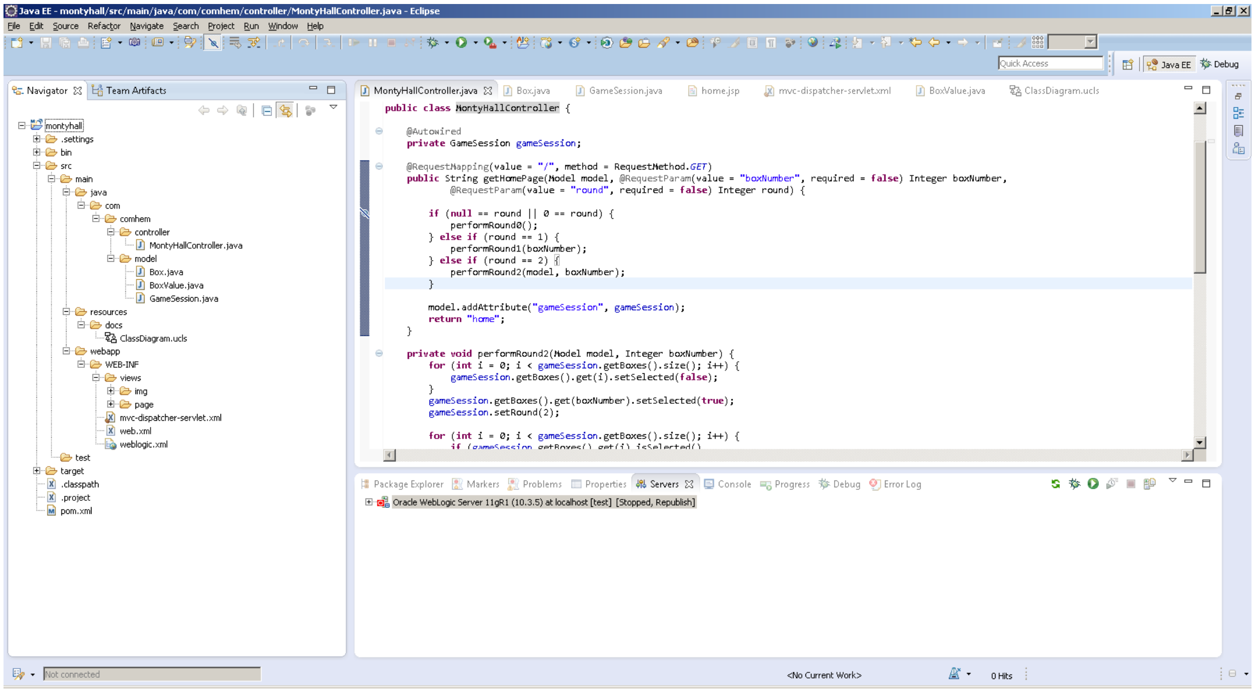
How to play:

1- Select any box.

2- Select the other box if you want to switch, otherwise select yours again.

Of course it does not mean to win every time, but the key thing here is when you swap the box second time, the probably of the winning is greater than selecting and keeping the box from the first round.

IDW Screenshot:



```
montyhall
    .settings
    bin
    src
        main
            java
                com
                    comhem
                        controller
                            MontyHallController.java
                        model
                            Box.java
                            BoxValue.java
                            GameSession.java
            resources
                docs
                    ClassDiagram.ucls
            webapp
                WEB-INF
                    views
                        img
                        page
                    mvc-dispatcher-servlet.xml
                    web.xml
                    weblogic.xml
        test
    target
    .classpath
    .project
    pom.xml
```

```java
public class MontyHallController {

    @Autowired
    private GameSession gameSession;

    @RequestMapping(value = "/", method = RequestMethod.GET)
    public String getHomePage(Model model, @RequestParam(value = "boxNumber", required = false) Integer boxNumber,
            @RequestParam(value = "round", required = false) Integer round) {

        if (null == round || 0 == round) {
            performRound0();
        } else if (round == 1) {
            performRound1(boxNumber);
        } else if (round == 2) {
            performRound2(model, boxNumber);
        }

        model.addAttribute("gameSession", gameSession);
        return "home";
    }

    private void performRound2(Model model, Integer boxNumber) {
        for (int i = 0; i < gameSession.getBoxes().size(); i++) {
            gameSession.getBoxes().get(i).setSelected(false);
        }
        gameSession.getBoxes().get(boxNumber).setSelected(true);
        gameSession.setRound(2);

        for (int i = 0; i < gameSession.getBoxes().size(); i++) {
            if (gameSession.getBoxes().get(i).isSelected()
```

```
Oracle WebLogic Server 11gR1 (10.3.5) at localhost [test] [Stopped, Republish]
```