TicTacToe Design Report

Adalheidur Hreinsdottir
Anna Berglind Jonsdottir
Gretar Atli Gretarsson
Kristin Asgeirsdottir
Kristjan Thor Jonsson

November 23, 2013

1 Introduction

The main idea was to program a game of TicTacToe by using test driven development. GitHub is used as a source control from the start of the project. For the intergration service we used travis and Heuroku was chosen to be the staging server.

2 The web based version

2.1 TicTacToe.java

The whole serverside implementation was in one class: TicTacToe.java. The whole design was implemented in the functions. We focused on creating a code that is loosely coupled and following Test Driven Development.

2.1.1 Variables

- char grid[] - Holds the information about the status of the gameboard
- int x - Holds the information: ”what cell in the grid will be changed”
- int player - Holds information about ”who has turn”. Takes value 1 for Player one, 2 for player two.

2.1.2 Functions

- TicTacToe() - The constructor makes a new instance of grid and player and uses the function initializedGrid() to initialize the grid.
- initializedGrid() - Initializes our grid by filling each array entry with -
- checkForWin() - Checks if either player is a winner by checking if some of the functions winningRow(), winningColumn() and winningDiagonal() returnes true.

- initializePlayer() - When the initializePlayer() function is called the player is initialized as player 1.
- winningRow() - Returns true if any row is full of the same symbol, and that symbol can not be -.
- winningColumn() - Returns true if any column is full of the same symbol, and that symbol can not be -.
- winningDiagonal() - Returns true if any diagonal line is full of the same symbol, and that symbol can not be -.
- checkIfValidInput() - Check if the input for grid is whithin the limit of the array
- checkForTie() - Check for tie is only called after check for win has ben called and returned false. Runs through the grid and returns true if no array entry contains -.
- changePlayer() - Changes who is the current player.
- insert() - inserts 'X' or 'O' into the grid, depending on which player has turn.

2.1.3 Playing

The main idea was to create 1 gameboard. The gameboard can be seen in Figure 1. Every cell in the grid is a button. Two person can play the game. Player 1 has the symbol X and player 2 has Y. The following steps are repeated until we get a result:

1. When a player presses a button the icon of the button will change to either x or y, depending on which player has turn.
2. A http post method is executed. By using a javascript and ajax post method we should be able to post the id of the grid over to the servlet.
3. Serverside the id is valified. If it is valid the cell that correspond to that id in the grid is updated: Insert grid[id] = X for player 1 and grid[id] = y for player 2.
4. The player is changed
5. Check for win. If a winner is found then the http post returns a string, either "player 1" or "player 2". Client side the return parameter is used to change the icon of the cell that was last pressed to "1!" if player one won but "2!" if player 2 won.
6. check for tie. If no one has won then a tie is checked. If a tie is found the http post returns the string "tie". Client side the return parameter is used to change the icon of the cell that was last pressed to "tie".
7. If no result is found then the http post returns either "X" or "Y". Client side the return parameter is used to change the icon of the cell that was last pressed to either "X" or "Y";

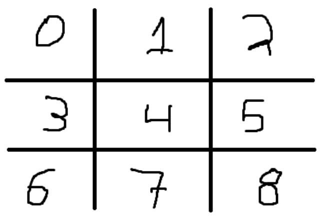

Figure 1: This is the gameboard.

3 Testing

The program was coded by using Test Driven Developement.Unit test were done on every function before the function was implemented. Selenium tests were used to do end to end tests (from UI to the database and back).

4 The console version

The first design of the project was a console program. The server side code is nearly the same for the web based version as it was implemented for the console version in the beginning. The difference is that the console verion contained a function called play. That function asked the user to choose a number from 0 - 8 depending on where he wants to put a sympol. Then the play function handled the answare like I descriped here above. We desited to create a web based version so this version will never be published.