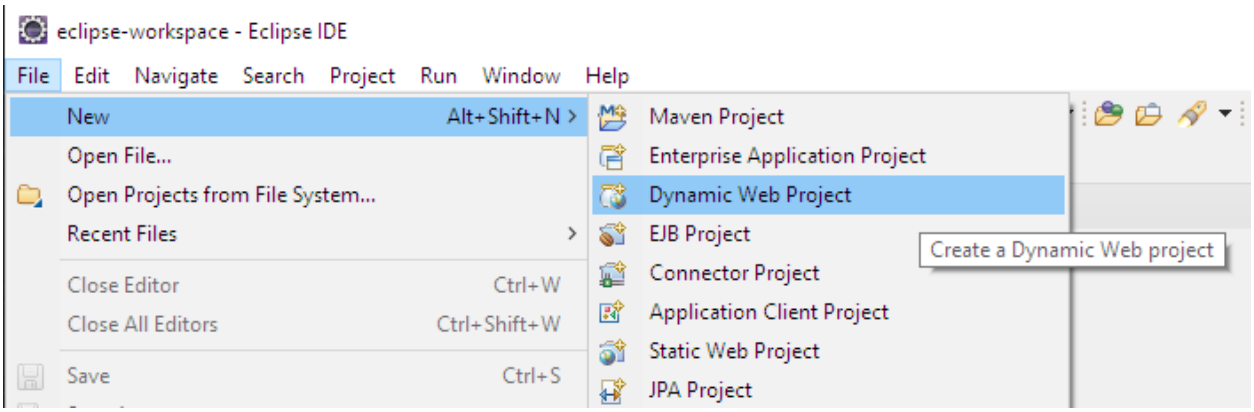
Laboratorio de Sistemas Telemáticos II
Microproyecto 2

Objetivo: Desarrollo de aplicaciones Web sobre J2EE empleando Eclipse.

1. Cree un Dynamic Web Project en Eclipse llamado TicTacToe. TicTacToe es el popular juego tres en raya.



New Dynamic Web Project

Dynamic Web Project

Create a standalone Java-based Web Application or add it to a new or existing Enterprise Application.

Project name: TicTacToe

Project location

☑ Use default location

Location: C:\Users\carlo\eclipse-workspace\TicTacToe Browse...

Target runtime

WildFly 24.0 Runtime New Runtime...

Dynamic web module version

4.0

Configuration

Default Configuration for WildFly 24.0 Runtime Modify...

A good starting point for working with WildFly 24.0 Runtime runtime. Additional facets can later be installed to add new functionality to the project.

EAR membership

☐ Add project to an EAR

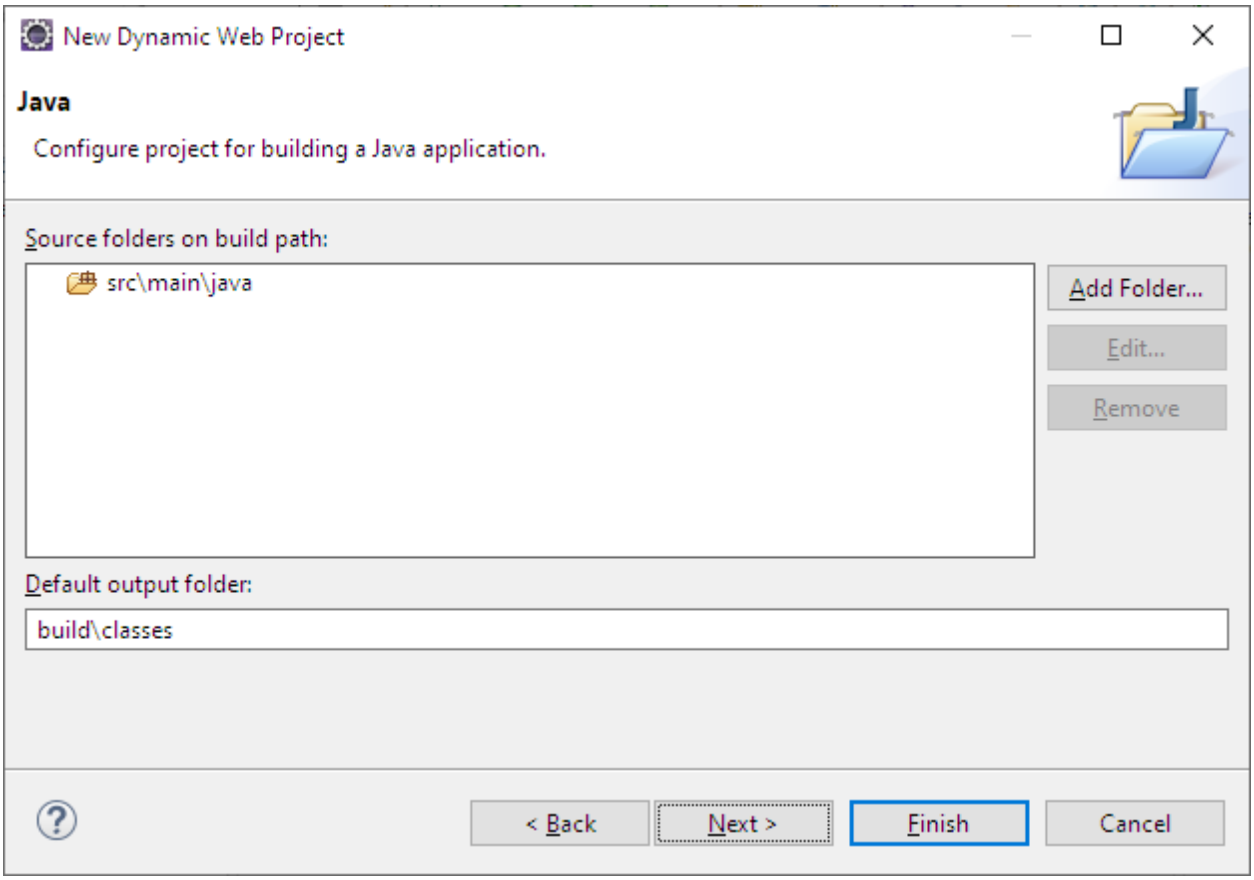
EAR project name: EAR New Project...

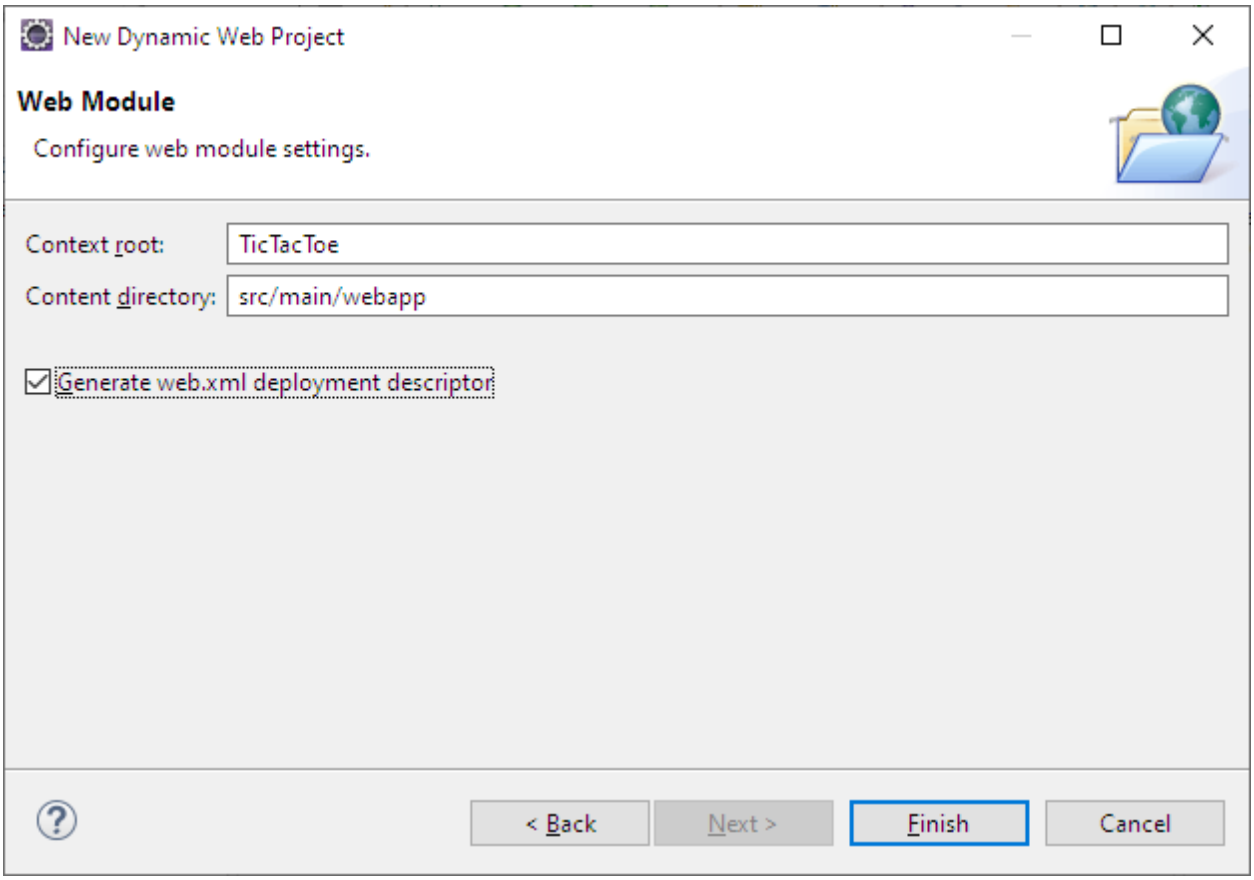
Working sets

☐ Add project to working sets New...

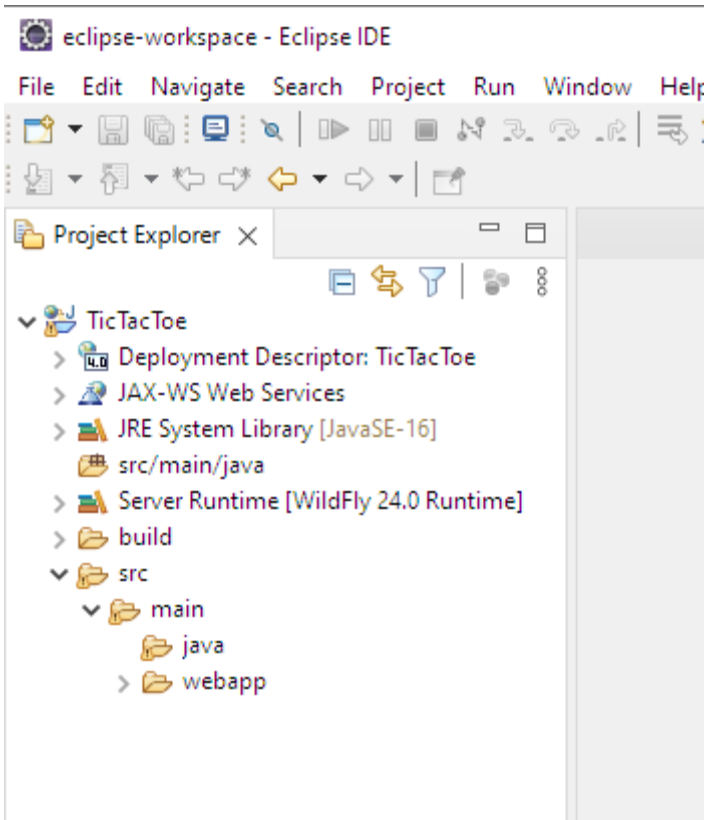
Working sets: Select...

< Back Next > Finish Cancel

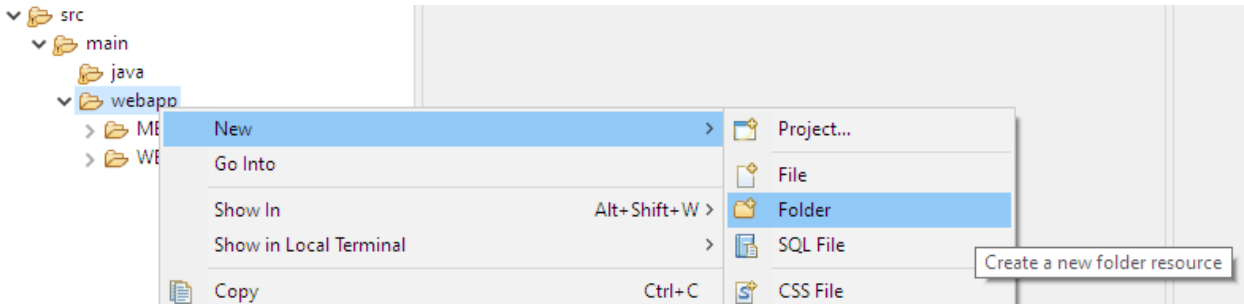

New Dynamic Web Project
Java
Configure project for building a Java application.
Source folders on build path:
src\main\java
Add Folder...
Edit...
Remove
Default output folder:
build\classes
< Back
Next >
Finish
Cancel

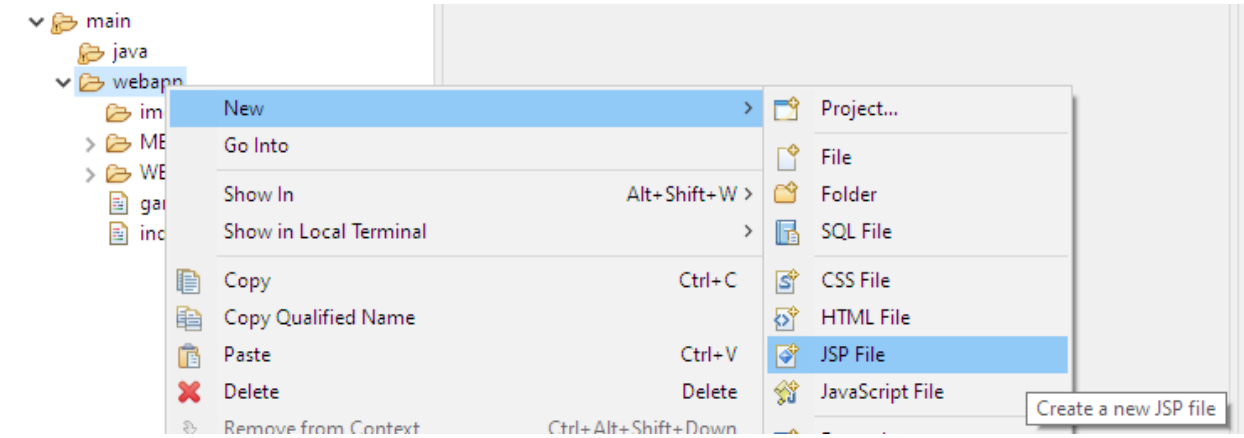


La estructura del proyecto debe ser la siguiente.

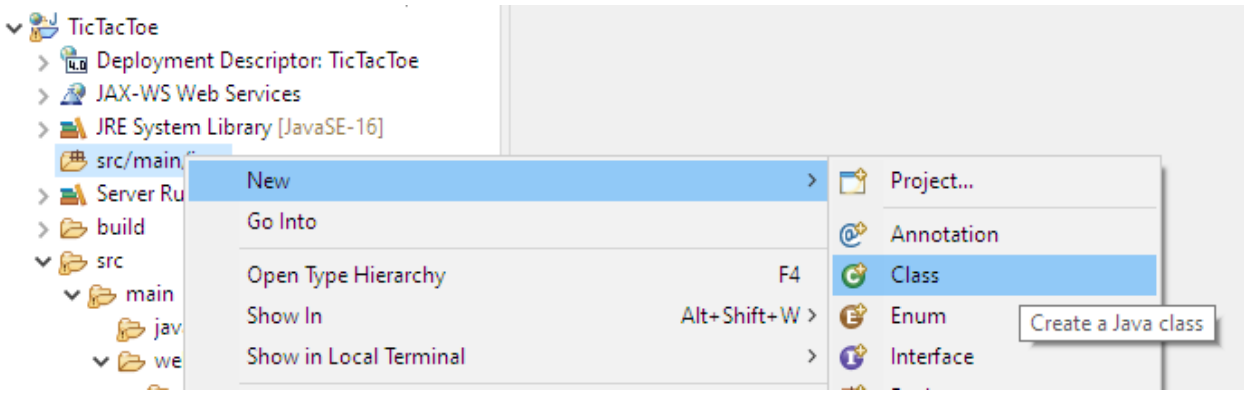


2. En la carpeta webapp, cree un folder llamado img, y dos archivos jsp llamados game.jsp e index.jsp.





3. En la paquete src/main/java, cree las clases Cell.java, GameBean.java, Line.java, EntryServlet.java y GameServlet.java. Estas clases deben pertenecer al package game.



New Java Class

Java Class

Create a new Java class.

Source folder: TicTacToe/src/main/java Browse...

Package: game Browse...

Enclosing type: Browse...

Name: Cell

Modifiers: public package private protected

abstract final static

Superclass: java.lang.Object Browse...

Interfaces: Add... Remove

Which method stubs would you like to create?

public static void main(String[] args)

Constructors from superclass

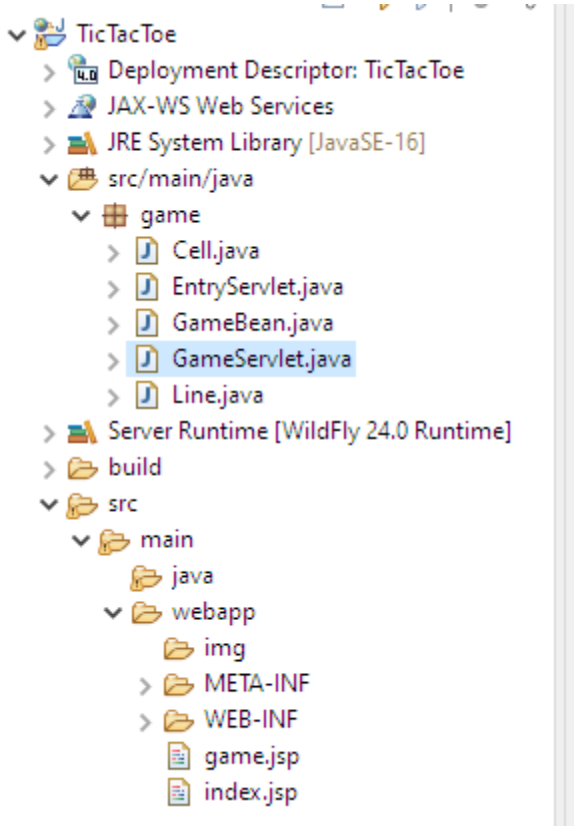
Inherited abstract methods

Do you want to add comments? (Configure templates and default value here)

Generate comments

< Back Next > Finish Cancel

La estructura del proyecto debe ser la siguiente.



4. Copie/reemplace el siguiente código en cada uno de los archivos creados.

Index.jsp

```
<jsp:useBean id="gameBean" scope="session" class="game.GameBean" />

<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>Tic Tac Toe</title>
    </head>
    <body>
        <h1>Tic Tac Toe</h1>
        <form action="entryServlet" method="post">
            <input type="submit" name="User" value="You start"><br/>
            <input type="submit" name="Computer" value="The computer
starts">
        </form>
    </body>
</html>
```

Game.jsp

```
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
```

```
<%@page import="game.GameBean.GameState" %>
<%@page import="game.Cell" %>
<%@page import="game.Line" %>

<jsp:useBean id="gameBean" scope="session" class="game.GameBean" />

<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>Tic Tac Toe</title>
  </head>
  <body>
    <h1>Tic Tac Toe</h1>
    <table border="1">
      <c:forEach var="line" items="${gameBean.gridLines}">
      <tr>
        <c:forEach var="cell" items="${gameBean.getGridStatus(line)}">
        <td>
          <c:choose>
            <c:when test="${cell.state == 'X'}">
              <img src="img/state_x.png" alt="X"/>
            </c:when>
            <c:when test="${cell.state == 'O'}">
              <img src="img/state_o.png" alt="O"/>
            </c:when>
            <c:otherwise>
              <c:if test="${winner == null}">
                <a href="gameServlet?Line=${cell.line}&Col=${cell.col}">
              </c:if>
                <img src="img/state_null.png" alt="null"/>
              <c:if test="${winner == null}">
                </a>
              </c:if>
            </c:otherwise>
          </c:choose>
        </td>
        </c:forEach>
      </tr>
      </c:forEach>
    </table>
    <c:if test="${winner != null}">
      <h2>${winner} Won!</h1>
      <form action="index.jsp" method="post">
        <input type="submit" name="Replay" value="Play again"><br/>
      </form>
    </c:if>
```

```
  </body>
</html>
```

Cell.java

```
package game;

public class Cell{
    private int cellLine;
    private int cellCol;
    private GameBean.GameState state;

    public Cell(GameBean.GameState state, int cellLine, int cellCol) {
        this.state= state;
        this.cellLine = cellLine;
        this.cellCol = cellCol;
    }

    public GameBean.GameState getState(){
        return this.state;
    }

    public int getLine() {
        return this.cellLine;
    }

    public int getCol() {
        return this.cellCol;
    }
}
```

Line.java

```
package game;

public class Line{
    private GameBean.GameState[] lineDatas;
    private int lineIndex;

    public Line(GameBean.GameState[] lineDatas, int lineIndex){
        this.lineDatas = lineDatas;
        this.lineIndex = lineIndex;
    }

    public GameBean.GameState[] getDatas() {
        return lineDatas;
    }

    public int getIndex() {
        return lineIndex;
    }
}
```

GameBean.java

```
package game;

import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;
import java.util.Random;

public class GameBean {
    private static final int GRID_SIZE = 3;

    public enum GameState{
        NULL, O, X;
    }
    public enum GamePlayer{
        USER(GameState.X),
        COMPUTER(GameState.O),
        NOBODY(GameState.NULL);

        private GameState state;
        private GamePlayer(GameState state){
            this.state = state;
        }
    }

    private boolean userFirst = true;
    private GameState[][] gameStatus;

    public GameBean(){
        this.gameStatus = new GameState[GRID_SIZE][GRID_SIZE];
    }

    public List<Line> getGridLines(){
        List<Line> lines = new ArrayList<>();
        int index = 0;
        for(GameState[] lineDatas : this.gameStatus){
            lines.add(new Line(lineDatas, index));
            index++;
        }
        return lines;
    }
    public List<Cell> getGridStatus(Line line){
        List<Cell> cells = new ArrayList<>();
        int index = 0;
        for(GameState state : line.getDatas()){
            cells.add(new Cell(state, line.getIndex(), index));
            index++;
        }
        return cells;
    }

    public void setStartByUser(boolean userFirst){
        this.userFirst = userFirst;
    }

    public void startGame(){
```

```
        for (int line = 0; line < GRID_SIZE; line++){
            for (int col = 0; col < GRID_SIZE; col++){
                this.gameStatus[line][col] = GameState.NULL;
            }
        }
        if(!this.userFirst){
            this.play(GamePlayer.COMPUTER, 1, 1);
        }
    }

    public void playPlayerTurn(int line, int col){
        this.play(GamePlayer.USER, line, col);
    }
    public void playComputerTurn(){
        int line = this.getRandomLineIndexWithEmptyCell();
        int col = this.getRandomEmptyCell(line);
        this.play(GamePlayer.COMPUTER, line, col);
    }
    private void play(GamePlayer player, int line, int col){
        if(this.gameStatus[line][col] == GameState.NULL){
            this.gameStatus[line][col] = player.state;
        }
    }

    private GamePlayer getPlayer(GameState state){
        for(GamePlayer player : GamePlayer.values()){
            if(player.state.equals(state)){
                return player;
            }
        }
        return null;
    }
    public GamePlayer getWinner(){
        //Lines
        for(int line = 0; line < GRID_SIZE; line++){
            GameState lineState = this.gameStatus[line][0];
            boolean win = true;
            for(int col = 0; col < GRID_SIZE; col++){
                if(!this.gameStatus[line][col].equals(lineState)){
                    win = false;
                    break;
                }
            }
            if(win){
                return this.getPlayer(lineState);
            }
        }
        //Cols
        for(int col = 0; col < GRID_SIZE; col++){
            GameState colState = this.gameStatus[0][col];
            boolean win = true;
            for(int line = 0; line < GRID_SIZE; line++){
                if(!this.gameStatus[line][col].equals(colState)){
                    win = false;
                    break;
                }
            }
```

```
            if(win){
                return this.getPlayer(colState);
            }
        }
        //Cross
        GameState pCrossState = this.gameStatus[0][0];
        GameState nCrossState = this.gameStatus[0][GRID_SIZE - 1];
        boolean pWin = true;
        boolean nWin = true;
        for(int index = 0; index < GRID_SIZE; index++){
            if(!this.gameStatus[index][index].equals(pCrossState)){
                pWin = false;
            }
            if(!this.gameStatus[index][GRID_SIZE - 1 -
index].equals(nCrossState)){
                nWin = false;
            }
        }
        if(pWin){
            return this.getPlayer(pCrossState);
        }
        else if(nWin){
            return this.getPlayer(nCrossState);
        }
        else{
            return GamePlayer.NOBODY;
        }
    }

    private static final Random rand = new Random();
    private int getRandomLineIndexWithEmptyCell(){
        if(!this.hasEmptyCell()){
            return -1;
        }
        List<Integer> indexes = new ArrayList();
        int index = 0;
        for(GameState[] line : this.gameStatus){
            boolean hasEmpty = false;
            for(GameState cell : line){
                if(cell == GameState.NULL){
                    hasEmpty = true;
                    break;
                }
            }
            if(hasEmpty){
                indexes.add(new Integer(index));
            }
            index++;
        }
        return indexes.get(rand.nextInt(indexes.size()));
    }
    private int getRandomEmptyCell(int line){
        if(!this.hasEmptyCell()){
            return -1;
        }
        List<Integer> indexes = new ArrayList();
        int index = 0;
```

```
        for(GameState cell : this.gameStatus[line]){
            if(cell == GameState.NULL){
                indexes.add(new Integer(index));
            }
            index++;
        }
        return indexes.get(rand.nextInt(indexes.size()));
    }
    public boolean hasEmptyCell(){
        for(int line = 0; line < GRID_SIZE; line++){
            for(int col = 0; col < GRID_SIZE; col++){
                if(this.gameStatus[line][col] == GameState.NULL){
                    return true;
                }
            }
        }
        return false;
    }
}
```

EntryServlet.java

```
package game;

import java.io.IOException;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import game.GameBean;

public class EntryServlet extends HttpServlet {

    /**
     * Processes requests for both HTTP <code>GET</code> and
<code>POST</code>
     * methods.
     *
     * @param request servlet request
     * @param response servlet response
     * @throws ServletException if a servlet-specific error occurs
     * @throws IOException if an I/O error occurs
     */
    protected void processRequest(HttpServletRequest request,
HttpServletResponse response)
            throws ServletException, IOException {
        String user = request.getParameter("User");
        boolean userFirst = user != null;

        GameBean game = (GameBean)
request.getSession(true).getAttribute("gameBean");
        game.setStartByUser(userFirst);
        game.startGame();

        request.getRequestDispatcher("/game.jsp").forward(request,
response);
```

```
    }

    // <editor-fold defaultstate="collapsed" desc="HttpServlet methods.
Click on the + sign on the left to edit the code.">
    /**
     * Handles the HTTP <code>GET</code> method.
     *
     * @param request servlet request
     * @param response servlet response
     * @throws ServletException if a servlet-specific error occurs
     * @throws IOException if an I/O error occurs
     */
    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse
response)
            throws ServletException, IOException {
        processRequest(request, response);
    }

    /**
     * Handles the HTTP <code>POST</code> method.
     *
     * @param request servlet request
     * @param response servlet response
     * @throws ServletException if a servlet-specific error occurs
     * @throws IOException if an I/O error occurs
     */
    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse
response)
            throws ServletException, IOException {
        processRequest(request, response);
    }

    /**
     * Returns a short description of the servlet.
     *
     * @return a String containing servlet description
     */
    @Override
    public String getServletInfo() {
        return "Process initial form response.";
    }// </editor-fold>

}
```

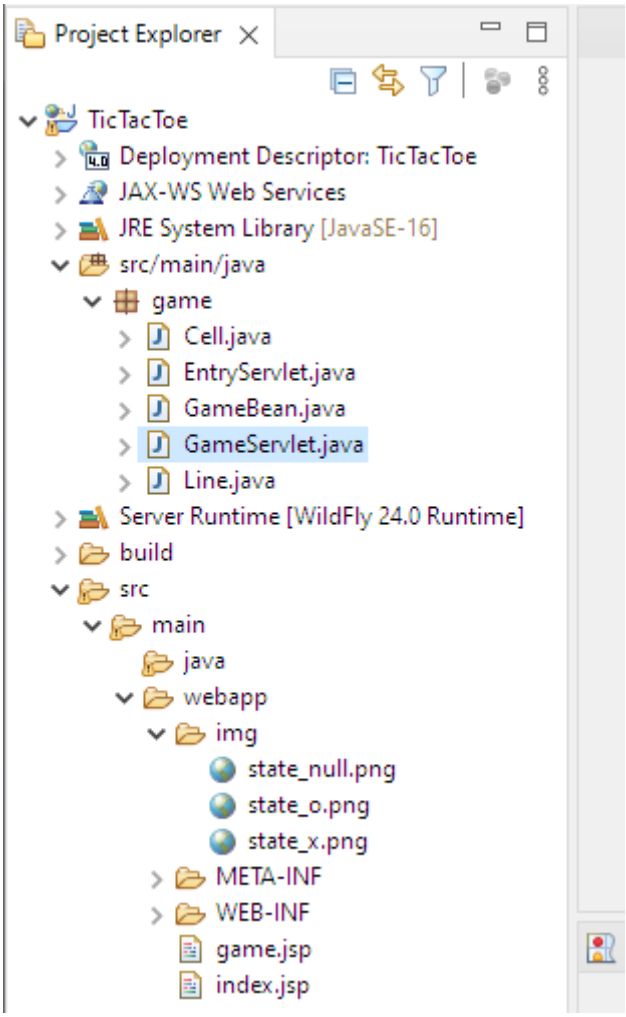
GameServlet.java

```
package game;

import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
```

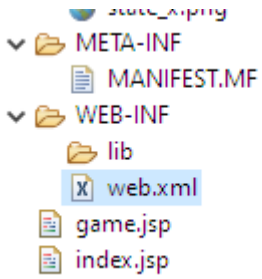
```
import javax.servlet.http.HttpServletResponse;
import game.GameBean;
import game.GameBean.GamePlayer;

public class GameServlet extends HttpServlet {

    /**
     * Processes requests for both HTTP <code>GET</code> and <code>POST</code>
     * methods.
     *
     * @param request servlet request
     * @param response servlet response
     * @throws ServletException if a servlet-specific error occurs
     * @throws IOException if an I/O error occurs
     */
    protected void processRequest(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {

        GameBean game = (GameBean) request.getSession(true).getAttribute("gameBean");

        int line = Integer.parseInt(request.getParameter("Line"));
        int col = Integer.parseInt(request.getParameter("Col"));

        game.playPlayerTurn(line, col);

        GamePlayer winner = game.getWinner();
        switch(winner){
            case NOBODY:
                if(game.hasEmptyCell()){
                    game.playComputerTurn();
                    switch(game.getWinner()){
                        case NOBODY:
                            break;
                        case COMPUTER:
                            request.setAttribute("winner", "The computer");
                            break;
                        case USER:
                            request.setAttribute("winner", "You");
                            break;
                    }
                }
                break;
            case COMPUTER:
                request.setAttribute("winner", "The computer");
                break;
            case USER:
                request.setAttribute("winner", "You");
                break;
```

```
        }
        if(winner == GamePlayer.NOBODY && !game.hasEmptyCell()){
            request.setAttribute("winner", "Nobody");
        }
        request.getRequestDispatcher("/game.jsp").forward(request, response);
    }

    // <editor-fold defaultstate="collapsed" desc="HttpServlet methods. Click on the + sign on the left
to edit the code.">
    /**
     * Handles the HTTP <code>GET</code> method.
     *
     * @param request servlet request
     * @param response servlet response
     * @throws ServletException if a servlet-specific error occurs
     * @throws IOException if an I/O error occurs
     */
    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        processRequest(request, response);
    }

    /**
     * Handles the HTTP <code>POST</code> method.
     *
     * @param request servlet request
     * @param response servlet response
     * @throws ServletException if a servlet-specific error occurs
     * @throws IOException if an I/O error occurs
     */
    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        processRequest(request, response);
    }

    /**
     * Returns a short description of the servlet.
     *
     * @return a String containing servlet description
     */
    @Override
    public String getServletInfo() {
        return "Short description";
    }// </editor-fold>

}
```

5. En la carpeta img, copie los archivos de imagen state_null.png, state_o.png y state_x.png.



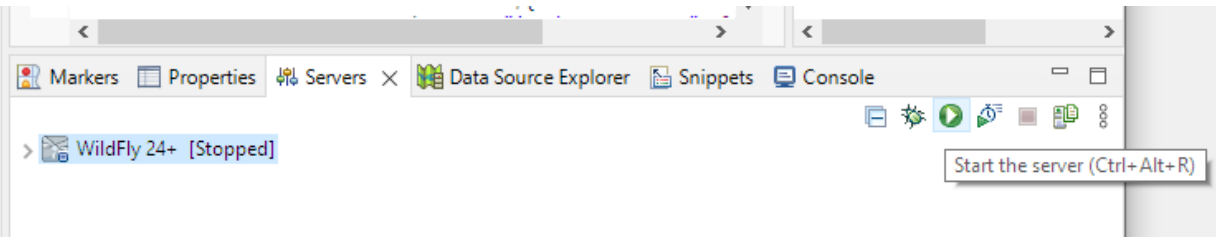
6. Copie/reemplace el siguiente descriptor de despliegue en el archivo web.xml.




```
<?xml version="1.0" encoding="UTF-8"?>
<web-app version="3.1" xmlns="http://xmlns.jcp.org/xml/ns/javaee"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
```

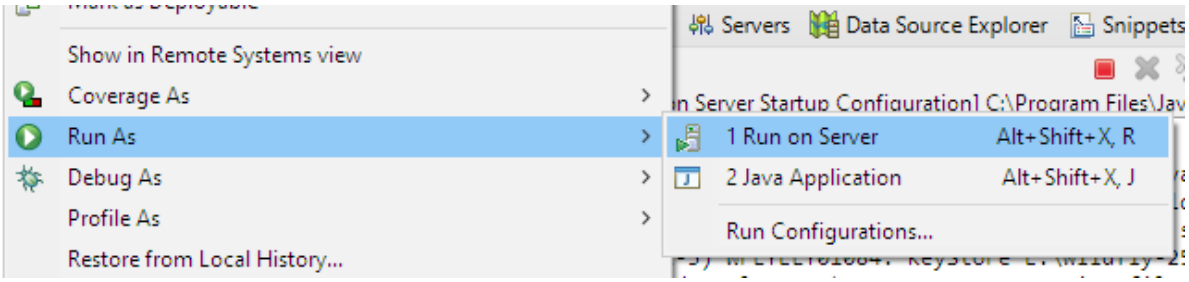
```
xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee http://xmlns.jcp.org/xml/ns/javaee/web-
app_3_1.xsd">
 <servlet>
    <servlet-name>entryServlet</servlet-name>
    <servlet-class>game.EntryServlet</servlet-class>
  </servlet>
  <servlet>
    <servlet-name>gameServlet</servlet-name>
    <servlet-class>game.GameServlet</servlet-class>
  </servlet>
  <servlet-mapping>
    <servlet-name>entryServlet</servlet-name>
    <url-pattern>/entryServlet</url-pattern>
  </servlet-mapping>
  <servlet-mapping>
    <servlet-name>gameServlet</servlet-name>
    <url-pattern>/gameServlet</url-pattern>
  </servlet-mapping>
  <session-config>
    <session-timeout>
      30
    </session-timeout>
  </session-config>
</web-app>
```

7. Corra el servidor WildFly integrado en Eclipse.



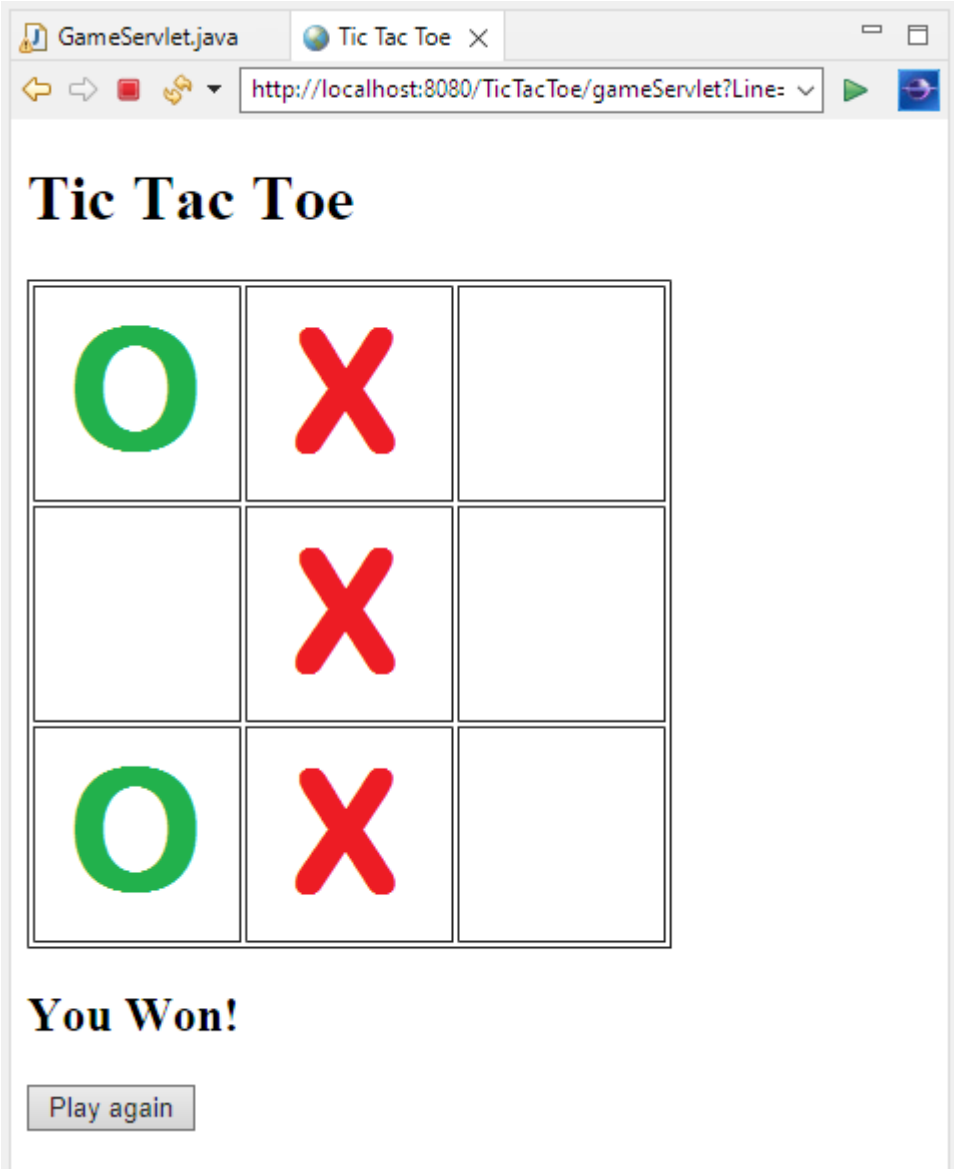
```
WildFly 24+ [JBoss Application Server Startup Configuration] C:\Program Files\Java\jdk-17.0.1\bin\javaw.exe (Dec 21, 2021, 9:07
ice thread 1-3) WFLYNAM0003: Starting Naming Service
MSC service thread 1-3) WFLYMAIL0001: Bound mail session [java:jboss/mail/Default]
 (MSC service thread 1-8) WFLYELY00023: KeyStore file 'E:\wildfly-25.0.1.Final\standalone\conf
] (MSC service thread 1-5) WFLYUT0003: Undertow 2.2.12.Final starting
 (MSC service thread 1-3) WFLYELY01084: KeyStore E:\wildfly-25.0.1.Final\standalone\configurat
] (ServerService Thread Pool -- 75) WFLYUT0014: Creating file handler for path 'E:\wildfly-25.
verService Thread Pool -- 55) WFLYIO001: Worker 'default' has auto-configured to 24 IO threads
e thread 1-8) WFLYEJB0481: Strict pool slsb-strict-max-pool is using a max instance size of 19
e thread 1-7) WFLYEJB0482: Strict pool mdb-strict-max-pool is using a max instance size of 48
ce thread 1-2) JBoss Remoting version 5.0.23.Final
] (MSC service thread 1-2) WFLYUT0012: Started server default-server.
] (MSC service thread 1-7) Queuing requests.
] (MSC service thread 1-7) WFLYUT0018: Host default-host starting
] (MSC service thread 1-1) WFLYUT0006: Undertow HTTP listener default listening on 127.0.0.1:8
e thread 1-6) WFLYEJB0493: Jakarta Enterprise Beans subsystem suspension complete
] (MSC service thread 1-2) WFLYUT0006: Undertow HTTPS listener https listening on 127.0.0.1:84
ems.datasources] (MSC service thread 1-5) WFLYJCA0001: Bound data source [java:jboss/datasourc
vice thread 1-3) WFLYPAT0050: WildFly Full cumulative patch ID is: base, one-off patches incl
.scanner] (MSC service thread 1-6) WFLYDS0013: Started FileSystemDeploymentService for directo
] (MSC service thread 1-7) JBWS022052: Starting JBossWS 5.4.4.Final (Apache CXF 3.4.4)
er Boot Thread) WFLYSRV0212: Resuming server
 Thread) WFLYSRV0025: WildFly Full 25.0.1.Final (WildFly Core 17.0.3.Final) started in 14906ms
 Thread) WFLYSRV0060: Http management interface listening on http://127.0.0.1:9990/management
 Thread) WFLYSRV0051: Admin console listening on http://127.0.0.1:9990
```

8. Corra la aplicación en el servidor.



Debe ver la siguiente aplicación web.

9. Pruebe su aplicación.


GameServlet.java
Tic Tac Toe
http://localhost:8080/TicTacToe/gameServlet?Line=
Tic Tac Toe
O X
X
O X
You Won!
Play again