```
using System;
using System.Collections.Generic;
using System.Data.Entity;
using System.IO;
using System.Linq;
using System.Reflection;
using System.Web;
using System.Web.Mvc;
using System.Web.UI;
using System.Web.UI.WebControls;
//using Excel = Microsoft.Office.Interop.Excel;
using Excel;
using OfficeOpenXml;
using CupPlaner.Helpers;
using System.Data;

namespace CupPlaner.Controllers
{
    public class TournamentController : Controller
    {
        CupDBContainer db = new CupDBContainer();
        ScheduleManager sm = new ScheduleManager();

        // GET: Tournament/Details/5
        // Define a list with divisions, fields and times.
        // Going through all the divisiosns, add some divisions with name and id.
        // Return a JSON object if there is a tournamet with a password, id, name, fields
and divisions.
        // If there doesnt exist a tournament it will return a JSON object and get a
error.
        public ActionResult Details(int id)
        {

            try
            {
                Validator validator = new Validator();
                Tournament t = db.TournamentSet.Find(id);
                List<object> divs = new List<object>();
                List<object> fields = new List<object>();
                List<object> times = new List<object>();

                //isScheduleReady - for frontend use.
                bool FrontendValidation = validator.IsScheduleReady(t.Id);

                // Get all divisions
                if (t.Divisions != null)
                {
                    foreach (Division d in t.Divisions)
                    {
                        divs.Add(new { Id = d.Id, Name = d.Name });
                    }
                }
                // Get all time intervals
                if (t.TimeIntervals != null)
                {
                    foreach (TimeInterval ti in t.TimeIntervals)
                    {
                        times.Add(new { Id = ti.Id, StartTime = ti.StartTime, EndTime =
ti.EndTime });
                    }
                }
                // Get all fields
                if (t.Fields != null)
                {
                    foreach (Field f in t.Fields)
                    {
                        fields.Add(new { Id = f.Id, Name = f.Name, fieldSize = f.Size });
                    }
                }
                object obj = new { status = "success", Id = t.Id, Name = t.Name, Password
= t.Password, Divisions = divs, TimeIntervals = times, Fields = fields, isValid =
FrontendValidation, IsScheduled = t.IsScheduled };

                return Json(obj, JsonRequestBehavior.AllowGet);
            }
            catch (Exception ex)
            {
                return Json(new { status = "error", message = "Could not find
tournament", details = ex.Message }, JsonRequestBehavior.AllowGet);
            }
        }

        [HttpPost]
        public ActionResult ChangeIsScheduled(int tournamentId)
        {
            Tournament t = db.TournamentSet.Find(tournamentId);
            t.IsScheduled = true;
            db.Entry(t).State = EntityState.Modified;
            db.SaveChanges();
            return Json(new { State = "success" });
        }

        // This function will create a password as a string.
        // Find a password to the tournament in the databse.
        [HttpPost]
        public ActionResult IdFromPass(string password)
        {
            Tournament t = db.TournamentSet.SingleOrDefault(x => x.Password == password);
            if (t == null)
            {
                return Json(new { Id = 0 });
            }
            return Json(new { Id = t.Id });
        }

        // POST: Tournament/Create
        // This function will have name, password starttime and endtime.
        //

        public ActionResult Create(string name, string password, string startTimes,
string endTimes)
        {
            try
            {
                if (!db.TournamentSet.Any(x => x.Password == password))
                {
                    Tournament t = new Tournament();
                    List<TimeInterval> tis = new List<TimeInterval>();

                    HttpPostedFileBase file = null;
                    int poolStart = 1;

                    List<DateTime> startTimesList =
startTimes.Split(',').Select(DateTime.Parse).ToList();
                    List<DateTime> endTimesList =
endTimes.Split(',').Select(DateTime.Parse).ToList();
                    for (int i = 0; i < startTimesList.Count; i++)
                    {
                        tis.Add(new TimeInterval() { StartTime = startTimesList[i],
EndTime = endTimesList[i] });
                        db.TimeIntervalSet.Add(new TimeInterval() { StartTime =
startTimesList[i], EndTime = endTimesList[i] });
                    }

                    if (Request != null && Request.Files.Count > 0 && Request.Files[0] !=
null && Request.Files[0].ContentLength > 0)
                    {
                        file = Request.Files[0];
                        IExcelDataReader excelReader;
                        switch (Path.GetExtension(file.FileName))
                        {
                            case ".xlsx":
                                excelReader =
ExcelReaderFactory.CreateOpenXmlReader(file.InputStream);
                                break;
                            case ".xls":
                                excelReader =
ExcelReaderFactory.CreateBinaryReader(file.InputStream);
                                break;
                            default:
                                excelReader =
ExcelReaderFactory.CreateBinaryReader(file.InputStream);
                                break;
                        }

                        DataSet result = excelReader.AsDataSet();

                        List<string> divisions = new List<string>();
                        List<string> pools = new List<string>();
                        List<string> teams = new List<string>();

                        Division d = new Division();
                        Pool p = new Pool();

                        int missingFLs = 0;
                        int flIndex = 0;

                        t.Name = result.Tables[0].Rows[0][0].ToString();
                        object[] stopRow = new object[1];
                        stopRow[0] = "Stop";
                        result.Tables[0].Rows.Add(stopRow);

                        for (int i = 1; i < result.Tables[0].Rows.Count; i++)
                        {
                            if
(!string.IsNullOrEmpty(result.Tables[0].Rows[i][0].ToString()) ||
string.IsNullOrEmpty(result.Tables[0].Rows[i][1].ToString()))
                            {
                                if (t.Divisions.Count > 0)
                                {
                                    for (int j = poolStart; j < i; j++)
                                    {

                                        p = new Pool() { Division = d, Name =
result.Tables[0].Rows[j][1].ToString(), IsAuto = false };

                                        for (int teamIndex = 2; teamIndex < 25;
teamIndex++)
                                        {
                                            try
                                            {
                                                string teamName =
result.Tables[0].Rows[j][teamIndex].ToString();
                                                if
(!string.IsNullOrEmpty(result.Tables[0].Rows[j][teamIndex].ToString()))
                                                {

                                                    Team newTeam = new Team() { Name =
teamName, Pool = p, IsAuto = false, };
                                                    foreach (TimeInterval ti in tis)
                                                    {

newTeam.TimeIntervals.Add(db.TimeIntervalSet.Add(new TimeInterval { StartTime =
ti.StartTime, EndTime = ti.EndTime }));
                                                    }
                                                    p.Teams.Add(newTeam);
                                                    newTeam = db.TeamSet.Add(newTeam);
                                                }
                                                else
                                                    break;
                                            } catch(Exception e)
                                            {
                                                break;
                                            }

                                        }
                                        p = db.PoolSet.Add(p);
                                        if (p.Teams.Count > missingFLs)
                                            missingFLs = p.Teams.Count;
                                    }
                                    for (flIndex = 0; flIndex < missingFLs; flIndex++)
                                    {
                                        d.FinalsLinks.Add(new FinalsLink() { Division =
d, PoolPlacement = flIndex + 1, Finalstage = flIndex + 1 });
                                    }
                                }

                                if
(string.IsNullOrEmpty(result.Tables[0].Rows[i][1].ToString()))
                                    break;
                                d = new Division() { Tournament = t, Name =
result.Tables[0].Rows[i][0].ToString(), FieldSize = FieldSize.ElevenMan, MatchDuration =
60 };
                                d = db.DivisionSet.Add(d);
                                missingFLs = 0;
                                poolStart = i;
                            }

                        }
                        excelReader.Close();
                    }
                    t.Name = name;
                    t.Password = password;
                    t.TimeIntervals = tis;
                    t = db.TournamentSet.Add(t);
                    db.SaveChanges();

                    return Json(new { status = "success", message = "New tournament
added", id = t.Id }, JsonRequestBehavior.AllowGet);

                }
                return Json(new { status = "error", message = "Password already exists"
}, JsonRequestBehavior.AllowGet);
            }
            catch (Exception ex)
            {
                return Json(new { status = "error", message = "New tournament not added",
details = ex.Message }, JsonRequestBehavior.AllowGet);
            }
        }



        /*[HttpPost]
        public ActionResult Create(string name, string password, string startTimes,
string endTimes)
        {
            try
            {
                if (!db.TournamentSet.Any(x => x.Password == password))
                {
                    Tournament t = new Tournament();
                    List<Team> teams2 = new List<Team>();
                    List<TimeInterval> tis = new List<TimeInterval>();

                    HttpPostedFileBase file = null;
                    int poolStart = 2;

                    List<DateTime> startTimesList =
startTimes.Split(',').Select(DateTime.Parse).ToList();
                    List<DateTime> endTimesList =
endTimes.Split(',').Select(DateTime.Parse).ToList();
                    for (int i = 0; i < startTimesList.Count; i++)
                    {
                        tis.Add(new TimeInterval() { StartTime = startTimesList[i],
EndTime = endTimesList[i] });
```

```
                        db.TimeIntervalSet.Add(new TimeInterval() { StartTime =
startTimesList[i], EndTime = endTimesList[i] });
                    }

                    if (Request != null && Request.Files.Count > 0 && Request.Files[0] !=
null && Request.Files[0].ContentLength > 0)
                    {
                        file = Request.Files[0];
                        string charRange = "CDEFGHIJKLMNOPQRSTUVXYZ";
                        List<string> divisions = new List<string>();
                        List<string> pools = new List<string>();
                        List<string> teams = new List<string>();

                        Division d = new Division();
                        Pool p = new Pool();

                        int missingFLs = 0;
                        int flIndex = 0;

                        // extract only the filename
                        var fileName = Path.GetFileName(file.FileName);
                        // store the file inside ~/App_Data/uploads folder
                        var path = Path.Combine(Server.MapPath("~/App_Data/Excel"),
fileName);

                        // UNCOMMENT TO SAVE FILE
                        //file.SaveAs(path);
                        var excel = new Excel.Application();
                        excel.Workbooks.Open(path);
                        Excel.Worksheet sheet = excel.Sheets["Cup"] as Excel.Worksheet;
                        Excel.Range range = sheet.get_Range("A1", Missing.Value);
                        Excel.Range range2 = sheet.get_Range("A2", Missing.Value);
                        Excel.Range poolsRange;

                        t.Name = range.Value;

                        for (int i = 2; i < 10000; i++)
                        {
                            range = sheet.get_Range("A" + i.ToString(), Missing.Value);
                            range2 = sheet.get_Range("B" + i.ToString(), Missing.Value);
                            if (range.Value != null || range2.Value == null)
                            {
                                if (t.Divisions.Count > 0)
                                {
                                    for (int j = poolStart; j < i; j++)
                                    {
                                        poolsRange = sheet.get_Range("B" + j.ToString(),
Missing.Value);
                                        p = new Pool() { Division = d, Name =
poolsRange.Value, IsAuto = false };

                                        foreach (char c in charRange.ToList())
                                        {
                                            var teamsRange = sheet.get_Range(c +
j.ToString(), Missing.Value);
                                            if (!string.IsNullOrEmpty(teamsRange.Value))
                                            {
                                                Team newTeam = new Team() { Name =
teamsRange.Value, Pool = p, IsAuto = false, };
                                                foreach (TimeInterval ti in tis)
                                                {

newTeam.TimeIntervals.Add(db.TimeIntervalSet.Add(new TimeInterval { StartTime =
ti.StartTime, EndTime = ti.EndTime }));
                                                }
                                                p.Teams.Add(newTeam);
                                                newTeam = db.TeamSet.Add(newTeam);
                                            }
                                            else
                                                break;
                                        }
                                        p = db.PoolSet.Add(p);
                                        if (p.Teams.Count > missingFLs)
                                            missingFLs = p.Teams.Count;
                                    }
                                    for (flIndex = 0; flIndex < missingFLs; flIndex++)
                                    {
                                        d.FinalsLinks.Add(new FinalsLink() { Division =
d, PoolPlacement = flIndex + 1, Finalstage = flIndex + 1 });
                                    }
                                }

                                if (range2.Value == null)
                                    break;

                                d = new Division() { Tournament = t, Name = range.Value,
FieldSize = FieldSize.ElevenMan, MatchDuration = 60 };

                                d = db.DivisionSet.Add(d);
                                missingFLs = 0;
                                poolStart = i;
                            }
                        }
                        // TODO insert some close workbook
                    }

                    foreach (Team teamitem in teams2)
                    {
                        db.TeamSet.Add(teamitem);
                    }
                    t.Name = name;
                    t.Password = password;
                    t.TimeIntervals = tis;
                    t = db.TournamentSet.Add(t);
                    db.SaveChanges();

                    return Json(new { status = "success", message = "New tournament
added", id = t.Id }, JsonRequestBehavior.AllowGet);
                }
                return Json(new { status = "error", message = "Password already exists"
}, JsonRequestBehavior.AllowGet);
            }
            catch (Exception ex)
            {
                return Json(new { status = "error", message = "New tournament not added", 
details = ex.Message }, JsonRequestBehavior.AllowGet);
            }
        }*/

        public ActionResult ExportExcel(int tournamentId)
        {

            Tournament t = db.TournamentSet.Find(tournamentId);
            FileInfo exportFile = new 
FileInfo(Path.Combine(Server.MapPath("~/App_Data/ExcelExport"), t.Id.ToString() + 
".xls"));
            ExcelPackage pck = new ExcelPackage(exportFile);
            string chars = "ABCDEFGHIJKLMNOPQRSTUVXYZ";

            try
            {
                var cupSheet = pck.Workbook.Worksheets.Add("Cup");

                //Add table headers going cell by cell.
                cupSheet.Cells["A1"].Value = "CID";
                cupSheet.Cells["B1"].Value = "MD5ID";
                cupSheet.Cells["C1"].Value = "Navn";
                cupSheet.Cells["D1"].Value = "Start";
                cupSheet.Cells["E1"].Value = "PointForSejr";
                cupSheet.Cells["F1"].Value = "BrugKontakter";
                cupSheet.Cells["G1"].Value = "KampSortering";
                cupSheet.Cells["H1"].Value = "BeregningsMetode";

                cupSheet.Cells[chars[0] + "2"].Value = "892";
                cupSheet.Cells[chars[1] + "2"].Value = "xxxx";
                cupSheet.Cells[chars[2] + "2"].Value = t.Name;
                cupSheet.Cells[chars[3] + "2"].Value = 
t.TimeIntervals.First().StartTime.ToString();
                cupSheet.Cells[chars[4] + "2"].Value = "xxxx";
                cupSheet.Cells[chars[5] + "2"].Value = "0";
                cupSheet.Cells[chars[6] + "2"].Value = "xxxx";
                cupSheet.Cells[chars[7] + "2"].Value = "FBCADE";

                var fieldsSheet = pck.Workbook.Worksheets.Add("Spillesteder");

                fieldsSheet.Cells["A1"].Value = "SID";
                fieldsSheet.Cells["B1"].Value = "CID";
                fieldsSheet.Cells["C1"].Value = "Navn";
                fieldsSheet.Cells["D1"].Value = "Adresse";

                int fieldsIndex = 2;
                foreach (Field f in t.Fields)
                {
                    fieldsSheet.Cells[chars[0] + fieldsIndex.ToString()].Value = 
f.Id.ToString();
                    fieldsSheet.Cells[chars[1] + fieldsIndex.ToString()].Value = 
t.Id.ToString();
                    fieldsSheet.Cells[chars[2] + fieldsIndex.ToString()].Value = f.Name;
                    fieldsSheet.Cells[chars[3] + fieldsIndex.ToString()].Value = "";
                    fieldsIndex++;
                }
                var divSheet = pck.Workbook.Worksheets.Add("Rækker");

                divSheet.Cells["A1"].Value = "RID";
                divSheet.Cells["B1"].Value = "CID";
                divSheet.Cells["C1"].Value = "Navn";

                int divIndex = 2;
                foreach (Division d in t.Divisions)
                {
                    divSheet.Cells[chars[0] + divIndex.ToString()].Value =
d.Id.ToString();
                    divSheet.Cells[chars[1] + divIndex.ToString()].Value =
t.Id.ToString();
                    divSheet.Cells[chars[2] + divIndex.ToString()].Value = d.Name;
                    divIndex++;
                }

                var poolSheet = pck.Workbook.Worksheets.Add("Puljer");

                poolSheet.Cells["A1"].Value = "PID";
                poolSheet.Cells["B1"].Value = "RID";
                poolSheet.Cells["C1"].Value = "Type";
                poolSheet.Cells["D1"].Value = "Navn";
                poolSheet.Cells["E1"].Value = "Footer";
                poolSheet.Cells["F1"].Value = "Header";
                poolSheet.Cells["G1"].Value = "OpdaterStilling";


                int poolIndex = 2;
                foreach (Division d in t.Divisions)
                {
                    foreach (Pool p in d.Pools)
                    {
                        poolSheet.Cells[chars[0] + poolIndex.ToString()].Value =
p.Id.ToString();
                        poolSheet.Cells[chars[1] + poolIndex.ToString()].Value =
d.Id.ToString();
                        poolSheet.Cells[chars[2] + poolIndex.ToString()].Value =
"Normal";
                        poolSheet.Cells[chars[3] + poolIndex.ToString()].Value = p.Name;
                        poolSheet.Cells[chars[4] + poolIndex.ToString()].Value = "Ja";
                        poolSheet.Cells[chars[5] + poolIndex.ToString()].Value = "Ja";
                        poolSheet.Cells[chars[6] + poolIndex.ToString()].Value = "Ja";
                        poolIndex++;
                    }
                }


                var teamSheet = pck.Workbook.Worksheets.Add("Hold");

                teamSheet.Cells["A1"].Value = "HID";
                teamSheet.Cells["B1"].Value = "PID";
                teamSheet.Cells["C1"].Value = "OriginalNavn";
                teamSheet.Cells["D1"].Value = "Navn";
                teamSheet.Cells["E1"].Value = "Dispensation";
                teamSheet.Cells["F1"].Value = "Placering";
                teamSheet.Cells["G1"].Value = "PuljeId";
                teamSheet.Cells["H1"].Value = "TaberId";
                teamSheet.Cells["I1"].Value = "VinderId";
                teamSheet.Cells["J1"].Value = "Kontakt";
                teamSheet.Cells["K1"].Value = "Telefon";
                teamSheet.Cells["L1"].Value = "Email";
                teamSheet.Cells["M1"].Value = "Position";
                teamSheet.Cells["N1"].Value = "K";
                teamSheet.Cells["O1"].Value = "V";
                teamSheet.Cells["P1"].Value = "U";
                teamSheet.Cells["Q1"].Value = "T";
                teamSheet.Cells["R1"].Value = "M1";
                teamSheet.Cells["S1"].Value = "M2";
                teamSheet.Cells["T1"].Value = "P";

                int teamIndex = 2;
                foreach (Division d in t.Divisions)
                {
                    foreach (Pool p in d.Pools)
                    {
                        foreach (Team team in p.Teams)
                        {
                            teamSheet.Cells[chars[0] + teamIndex.ToString()].Value = 
team.Id.ToString();
                            teamSheet.Cells[chars[1] + teamIndex.ToString()].Value = 
p.Id.ToString();
                            teamSheet.Cells[chars[2] + teamIndex.ToString()].Value = "";
                            teamSheet.Cells[chars[3] + teamIndex.ToString()].Value = 
team.Name;
                            teamSheet.Cells[chars[4] + teamIndex.ToString()].Value = "";
                            teamSheet.Cells[chars[5] + teamIndex.ToString()].Value = "";
                            teamSheet.Cells[chars[6] + teamIndex.ToString()].Value = "";
                            teamSheet.Cells[chars[7] + teamIndex.ToString()].Value = "";
                            teamSheet.Cells[chars[8] + teamIndex.ToString()].Value = "";
                            teamSheet.Cells[chars[9] + teamIndex.ToString()].Value = "";
                            teamSheet.Cells[chars[10] + teamIndex.ToString()].Value = "";
                            teamSheet.Cells[chars[11] + teamIndex.ToString()].Value = "";
                            teamSheet.Cells[chars[12] + teamIndex.ToString()].Value = "";
                            teamSheet.Cells[chars[13] + teamIndex.ToString()].Value = "";
                            teamSheet.Cells[chars[14] + teamIndex.ToString()].Value = "";
                            teamSheet.Cells[chars[15] + teamIndex.ToString()].Value = "";
                            teamSheet.Cells[chars[16] + teamIndex.ToString()].Value = "";
                            teamSheet.Cells[chars[17] + teamIndex.ToString()].Value = "";
                            teamSheet.Cells[chars[18] + teamIndex.ToString()].Value = "";
                            teamSheet.Cells[chars[19] + teamIndex.ToString()].Value = "";
                            teamIndex++;
                        }
                    }
                }

                var matchSheet = pck.Workbook.Worksheets.Add("Kampe");

                matchSheet.Cells["A1"].Value = "KID";
                matchSheet.Cells["B1"].Value = "PID";
                matchSheet.Cells["C1"].Value = "Nummer";
                matchSheet.Cells["D1"].Value = "HjemmeholdID";
                matchSheet.Cells["E1"].Value = "UdeholdID";
                matchSheet.Cells["F1"].Value = "Resultat";
                matchSheet.Cells["G1"].Value = "M1";
                matchSheet.Cells["H1"].Value = "M2";
                matchSheet.Cells["I1"].Value = "Tid";
                matchSheet.Cells["J1"].Value = "Dato";
                matchSheet.Cells["K1"].Value = "Timer";
                matchSheet.Cells["L1"].Value = "Minutter";
                matchSheet.Cells["M1"].Value = "HalID";
                matchSheet.Cells["N1"].Value = "ArverResultat";

                int matchIndex = 2;
                foreach (Field f in t.Fields)
                {
                    foreach (Match match in f.Matches)
                    {
                        matchSheet.Cells[chars[0] + matchIndex.ToString()].Value =
match.Id.ToString();
                        matchSheet.Cells[chars[1] + matchIndex.ToString()].Value =
match.Teams.First().Pool.Id.ToString();
                        matchSheet.Cells[chars[2] + matchIndex.ToString()].Value = "";
                        matchSheet.Cells[chars[3] + matchIndex.ToString()].Value =
match.Teams.First().Id.ToString();
                        matchSheet.Cells[chars[4] + matchIndex.ToString()].Value =
match.Teams.Last().Id.ToString();
                        matchSheet.Cells[chars[5] + matchIndex.ToString()].Value = "";
                        matchSheet.Cells[chars[6] + matchIndex.ToString()].Value = "";
                        matchSheet.Cells[chars[7] + matchIndex.ToString()].Value = "";
                        matchSheet.Cells[chars[8] + matchIndex.ToString()].Value = "";
                        matchSheet.Cells[chars[9] + matchIndex.ToString()].Value = "";
                        matchSheet.Cells[chars[10] + matchIndex.ToString()].Value = "";
                        matchSheet.Cells[chars[11] + matchIndex.ToString()].Value = "";
                        matchSheet.Cells[chars[12] + matchIndex.ToString()].Value = "";
                        matchSheet.Cells[chars[13] + matchIndex.ToString()].Value = "";
                        matchIndex++;
                    }
                }
            }
            catch (Exception e)
            {
                return Json(new { State = "Failed", Exception = e },
JsonRequestBehavior.AllowGet);
            }

            Response.ContentType = "application/vnd.openxmlformats-
officedocument.spreadsheetml.sheet";
            Response.AddHeader("content-disposition", "attachment;
filename=Export.xlsx");
            Response.BinaryWrite(pck.GetAsByteArray());
            Response.End();
            return Json(new { State = "Success" }, JsonRequestBehavior.AllowGet);
        }


        //This function will export the scheduled tournament plan to an excel file
        /*public ActionResult ExportExcel(int tournamentId)
        {
            Microsoft.Office.Interop.Excel.Application oXL;
            Microsoft.Office.Interop.Excel._Workbook workbook;
            Microsoft.Office.Interop.Excel._Worksheet cupSheet;
            Microsoft.Office.Interop.Excel.Range oRng;
            object misvalue = System.Reflection.Missing.Value;
            Tournament t = db.TournamentSet.Find(tournamentId);
            string fileName = t.Id.ToString() + ".xls";
            //string fileName = "2.xls";
            try
            {

                //Start Excel and get Application object.
                oXL = new Microsoft.Office.Interop.Excel.Application();
                oXL.Visible = true;

                //Get a new workbook.
                workbook =
(Microsoft.Office.Interop.Excel._Workbook)(oXL.Workbooks.Add(""));
                cupSheet = workbook.ActiveSheet;
                cupSheet.Name = "Cup";

                //Add table headers going cell by cell.
                cupSheet.Cells[1, 1] = "CID";
                cupSheet.Cells[1, 2] = "MD5ID";
                cupSheet.Cells[1, 3] = "Navn";
                cupSheet.Cells[1, 4] = "Start";
                cupSheet.Cells[1, 5] = "PointForSejr";
                cupSheet.Cells[1, 6] = "BrugKontakter";
                cupSheet.Cells[1, 7] = "KampSortering";
                cupSheet.Cells[1, 8] = "BeregningsMetode";

                //Format A1:D1 as bold, vertical alignment = center.
                //oSheet.get_Range("A1", "D1").Font.Bold = true;
               // oSheet.get_Range("A1", "H1").VerticalAlignment =
                 //   Microsoft.Office.Interop.Excel.XlVAlign.xlVAlignCenter;

                // Create an array to multiple values at once.
                string[,] tournamentRow = new string[1, 8];

                tournamentRow[0, 0] = "892";
                tournamentRow[0, 1] = "xxxx";
                tournamentRow[0, 2] = t.Name;
                tournamentRow[0, 3] = t.TimeIntervals.First().StartTime.ToString();
                tournamentRow[0, 4] = "xxxx";
                tournamentRow[0, 5] = "0";
                tournamentRow[0, 6] = "xxxx";
                tournamentRow[0, 7] = "FBCADE";


                //Fill A2:B6 with an array of values (First and Last Names).
                cupSheet.get_Range("A2", "H2").Value2 = tournamentRow;

                //AutoFit columns A:H.
                oRng = cupSheet.get_Range("A1", "H1");
                oRng.EntireColumn.AutoFit();

                Excel.Worksheet fieldsSheet = workbook.Worksheets.Add();
                fieldsSheet.Move(Type.Missing, cupSheet);
                fieldsSheet.Name = "Spillesteder";

                fieldsSheet.Cells[1, 1] = "SID";
                fieldsSheet.Cells[1, 2] = "CID";
                fieldsSheet.Cells[1, 3] = "Navn";
                fieldsSheet.Cells[1, 4] = "Adresse";

                string[,] fieldsRow = new string[1, 4];

                int fieldsIndex = 2;
                foreach (Field f in t.Fields)
                {
                    fieldsRow = new string[1, 4];
                    fieldsRow[0, 0] = f.Id.ToString();
                    fieldsRow[0, 1] = t.Id.ToString();
                    fieldsRow[0, 2] = f.Name;
                    fieldsRow[0, 3] = "";

                    fieldsSheet.get_Range("A" + fieldsIndex.ToString(), "D" +
fieldsIndex.ToString()).Value2 = fieldsRow;
                    fieldsIndex++;
                }

                Excel.Worksheet divSheet = workbook.Worksheets.Add();
                divSheet.Move(Type.Missing, fieldsSheet);
                divSheet.Name = "Rækker";

                divSheet.Cells[1, 1] = "RID";
                divSheet.Cells[1, 2] = "CID";
                divSheet.Cells[1, 3] = "Navn";

                string[,] divRow = new string[1, 4];

                int divIndex = 2;
                foreach (Division d in t.Divisions)
                {
                    divRow = new string[1, 3];
                    divRow[0, 0] = d.Id.ToString();
                    divRow[0, 1] = t.Id.ToString();
                    divRow[0, 2] = d.Name;

                    divSheet.get_Range("A" + divIndex.ToString(), "C" +
divIndex.ToString()).Value2 = divRow;
                    divIndex++;
                }

                Excel.Worksheet poolSheet = workbook.Worksheets.Add();
                poolSheet.Move(Type.Missing, divSheet);
                poolSheet.Name = "Puljer";

                poolSheet.Cells[1, 1] = "PID";
                poolSheet.Cells[1, 2] = "RID";
                poolSheet.Cells[1, 3] = "Type";
                poolSheet.Cells[1, 4] = "Navn";
                poolSheet.Cells[1, 5] = "Footer";
                poolSheet.Cells[1, 6] = "Header";
                poolSheet.Cells[1, 7] = "OpdaterStilling";

                string[,] poolRow = new string[1, 7];

                int poolIndex = 2;
                foreach (Division d in t.Divisions)
                {
                    foreach (Pool p in d.Pools)
                    {
                        poolRow = new string[1, 7];
                        poolRow[0, 0] = p.Id.ToString();
                        poolRow[0, 1] = d.Id.ToString();
                        poolRow[0, 2] = "Normal";
                        poolRow[0, 3] = p.Name;
                        poolRow[0, 4] = "Ja";
                        poolRow[0, 5] = "Ja";
                        poolRow[0, 6] = "Ja";

                        poolSheet.get_Range("A" + poolIndex.ToString(), "G" +
poolIndex.ToString()).Value2 = poolRow;
                        poolIndex++;
                    }

                }
                oRng = poolSheet.get_Range("A1", "G1");
                oRng.EntireColumn.AutoFit();


                Excel.Worksheet teamSheet = workbook.Worksheets.Add();
                teamSheet.Move(Type.Missing, poolSheet);
                teamSheet.Name = "Hold";

                teamSheet.Cells[1, 1] = "HID";
                teamSheet.Cells[1, 2] = "PID";
                teamSheet.Cells[1, 3] = "OriginalNavn";
                teamSheet.Cells[1, 4] = "Navn";
                teamSheet.Cells[1, 5] = "Dispensation";
                teamSheet.Cells[1, 6] = "Placering";
                teamSheet.Cells[1, 7] = "PuljeId";
                teamSheet.Cells[1, 8] = "TaberId";
                teamSheet.Cells[1, 9] = "VinderId";
                teamSheet.Cells[1, 10] = "Kontakt";
                teamSheet.Cells[1, 11] = "Telefon";
                teamSheet.Cells[1, 12] = "Email";
                teamSheet.Cells[1, 13] = "Position";
                teamSheet.Cells[1, 14] = "K";
                teamSheet.Cells[1, 15] = "V";
                teamSheet.Cells[1, 16] = "U";
                teamSheet.Cells[1, 17] = "T";
                teamSheet.Cells[1, 18] = "M1";
                teamSheet.Cells[1, 19] = "M2";
                teamSheet.Cells[1, 20] = "P";

                string[,] teamRow = new string[1, 20];

                int teamIndex = 2;
                foreach (Division d in t.Divisions)
                {
                    foreach (Pool p in d.Pools)
                    {
                        foreach (Team team in p.Teams)
                        {
                            teamRow = new string[1, 20];
                            teamRow[0, 0] = team.Id.ToString();
                            teamRow[0, 1] = p.Id.ToString();
                            teamRow[0, 2] = "";
                            teamRow[0, 3] = team.Name;
                            teamRow[0, 4] = "";
                            teamRow[0, 5] = "";
                            teamRow[0, 6] = "";
                            teamRow[0, 7] = "";
                            teamRow[0, 8] = "";
                            teamRow[0, 9] = "";
                            teamRow[0, 10] = "";
                            teamRow[0, 11] = "";
                            teamRow[0, 12] = "";
                            teamRow[0, 13] = "";
                            teamRow[0, 14] = "";
                            teamRow[0, 15] = "";
                            teamRow[0, 16] = "";
                            teamRow[0, 17] = "";
                            teamRow[0, 18] = "";
                            teamRow[0, 19] = "";

                            teamSheet.get_Range("A" + teamIndex.ToString(), "T" +
teamIndex.ToString()).Value2 = teamRow;
                            teamIndex++;
                        }
                    }
                }

                oRng = teamSheet.get_Range("A1", "T1");
                oRng.EntireColumn.AutoFit();

                Excel.Worksheet matchSheet = workbook.Worksheets.Add();
                matchSheet.Move(Type.Missing, teamSheet);
                matchSheet.Name = "Kampe";

                matchSheet.Cells[1, 1] = "KID";
                matchSheet.Cells[1, 2] = "PID";
                matchSheet.Cells[1, 3] = "Nummer";
                matchSheet.Cells[1, 4] = "HjemmeholdID";
                matchSheet.Cells[1, 5] = "UdeholdID";
                matchSheet.Cells[1, 6] = "Resultat";
                matchSheet.Cells[1, 7] = "M1";
                matchSheet.Cells[1, 8] = "M2";
                matchSheet.Cells[1, 9] = "Tid";
                matchSheet.Cells[1, 10] = "Dato";
                matchSheet.Cells[1, 11] = "Timer";
                matchSheet.Cells[1, 12] = "Minutter";
                matchSheet.Cells[1, 13] = "HalID";
                matchSheet.Cells[1, 14] = "ArverResultat";

                string[,] matchRow = new string[1, 14];

                int matchIndex = 2;
                foreach (Field f in t.Fields)
                {
                    foreach (Match match in f.Matches)
                    {
                        matchRow = new string[1, 14];
                        matchRow[0, 0] = match.Id.ToString();
                        matchRow[0, 1] = match.Teams.First().Pool.Id.ToString();
                        matchRow[0, 2] = "";
                        matchRow[0, 3] = match.Teams.First().Id.ToString();
                        matchRow[0, 4] = match.Teams.Last().Id.ToString();
                        matchRow[0, 5] = "";
                        matchRow[0, 6] = "";
                        matchRow[0, 7] = "";
                        matchRow[0, 8] = "";
                        matchRow[0, 9] = "";
                        matchRow[0, 10] = "";
                        matchRow[0, 11] = "";
                        matchRow[0, 12] = "";
                        matchRow[0, 13] = "";

                        matchSheet.get_Range("A" + matchIndex.ToString(), "N" +
matchIndex.ToString()).Value2 = matchRow;
                        matchIndex++;
                    }
                }

                oRng = matchSheet.get_Range("A1", "N1");
                oRng.EntireColumn.AutoFit();

                oXL.Visible = false;
                oXL.UserControl = false;

                workbook.SaveAs(Path.Combine(Server.MapPath("~/App_Data/ExcelExport"),
fileName), Microsoft.Office.Interop.Excel.XlFileFormat.xlWorkbookDefault, Type.Missing,
Type.Missing,
                    false, false,
Microsoft.Office.Interop.Excel.XlSaveAsAccessMode.xlShared,
                    Type.Missing, Type.Missing, Type.Missing, Type.Missing,
Type.Missing);
            }
            catch (Exception e)
            {
                return Json(new { State = "Success", Exception = e },
JsonRequestBehavior.AllowGet);
            }

            workbook.Close();

            Response.AddHeader("content-disposition", "attachment; filename=1.xls");
            //return Content(oWB.ToString(), );

            return File(Path.Combine(Server.MapPath("~/App_Data/ExcelExport"), fileName),
"application/ms-excel");
        }*/

        // POST: Tournament/Edit/5
        // Find a id to the tournament in the database.
        // Generate a list of all Timeinterval.
        // Going through all the divisions, pools and teams
        // Find a id to team in the database
        // Try try remove the first and the last element in timeinterval in the database,
and going through all timeintervals.

        [HttpPost]
        public ActionResult Edit(int id, string name, string password, List<DateTime>
startTimes, List<DateTime> endTimes)
        {
            try
            {
                // Check if new password already exists, not including the old password
                if (db.TournamentSet.Any(x => x.Password == password))
                {
                    string oldPass = db.TournamentSet.Find(id).Password;
                    if (oldPass != password) return Json(new { status = "error", message
= "Password already exists" }, JsonRequestBehavior.AllowGet);
                }
                Tournament t = db.TournamentSet.Find(id);
                // Clear time intervals
                db.TimeIntervalSet.RemoveRange(t.TimeIntervals);
                List<TimeInterval> tis = new List<TimeInterval>();
                // Set new timeintervals
                for (int i = 0; i < startTimes.Count; i++)
                {
                    tis.Add(new TimeInterval() { StartTime = startTimes[i], EndTime =
endTimes[i] });
                }
                t.TimeIntervals = tis;
                // Set teams time interval to that of the tournament
                foreach (Division d in t.Divisions)
                {
                    foreach (Pool p in d.Pools)
                    {
                        foreach (Team tm in p.Teams)
                        {
                            Team team = db.TeamSet.Find(tm.Id);
                            db.TimeIntervalSet.RemoveRange(team.TimeIntervals);
                            foreach (TimeInterval ti in t.TimeIntervals)
                            {
                                TimeInterval timeinterval = new TimeInterval() { Team =
team, StartTime = ti.StartTime, EndTime = ti.EndTime };
                                db.TimeIntervalSet.Add(timeinterval);
                                team.TimeIntervals.Add(timeinterval);
                            }
                        }
                    }
                }
                t.Name = name;
                t.Password = password;

                //Clear the schedule
                sm.DeleteSchedule(t.Id, db);

                db.Entry(t).State = EntityState.Modified;
                db.SaveChanges();

                return Json(new { status = "success", message = "Tournament edited" },
JsonRequestBehavior.AllowGet);
            }
            catch (Exception ex)
            {
                return Json(new { status = "error", message = "Tournament not edited",
details = ex.Message }, JsonRequestBehavior.AllowGet);
```

```
            }
        }



        // POST: Tournament/Delete/5
        // Find a id to a tournament via the database
        // Going through all the division.
        // Return a JSON object if the tournament is deleted with succes, and error when
is not deleted.
        [HttpPost]
        public ActionResult Delete(int id)
        {
            try
            {
                Tournament t = db.TournamentSet.Find(id);
                DivisionController dc = new DivisionController();
                // Remove dependencies
                foreach (Division d in t.Divisions)
                {
                    dc.Delete(d.Id);
                }

                //Clear the schedule
                sm.DeleteSchedule(t.Id, db);

                db.TimeIntervalSet.RemoveRange(t.TimeIntervals);
                db.TournamentSet.Remove(t);
                db.SaveChanges();
                return Json(new { status = "success", message = "Tournament deleted" },
JsonRequestBehavior.AllowGet);
            }
            catch (Exception ex)
            {
                return Json(new { status = "error", message = "Tournament not deleted",
details = ex.Message }, JsonRequestBehavior.AllowGet);
            }
        }
    }
}
```