CODER FOUNDRY

Project Helper Class and Methods for Project Management

For certain repetitive functions, particularly those that involve working with Many to Many tables (e.g. UserRoles), it is often beneficial to create Helper classes to simplify the implementation of such functions. We might consider, for example creating a helper class to assist us in managing the assignment/un-assignment of user to projects, determining whether a user is assigned to a particular project so on and so forth…

It should be noted that building a Helper class to perform these functions is certainly not mandatory. It is simply *one* way of completing these tasks. The code for such Helper classes could very easily be integrated into your application's controller actions.

Building a Project Helper Class

1) You can construct a Helper class anywhere in your project. It might be placed in either the Models or Controllers folder, or you might construct a separate **Helpers** folder, particularly if you expect to create multiple Helper classes for your project.

2) Once the class is defined you must instantiate an object of that type (e.g. ProjectHelper) through which you will retrieve all project related information.

3) Within the ProjectHelper class, methods exist to help you
    a. Assign/Unassign users to projects
    b. Determine whether a user is assigned to a particular project
    c. Get a list of users on a certain project
    d. List projects to which a user is assigned.

```
using BugTracker_Reboot.Models;
using Microsoft.AspNet.Identity;
using Microsoft.AspNet.Identity.EntityFramework;
using System;
using System.Collections.Generic;
using System.Data.Entity;
using System.Data.Entity.Validation;
using System.Linq;
using System.Web;

namespace BugTracker_Reboot.Helpers
{
    public class ProjectsHelper
    {

        ApplicationDbContext db = new ApplicationDbContext();

        public bool IsUserOnProject(string userId, int projectId)
        {
            var project = db.Projects.Find(projectId);
            var flag = project.Users.Any(u => u.Id == userId);
            return (flag);
        }

        public ICollection<Project> ListUserProjects(string userId)
        {
            ApplicationUser user = db.Users.Find(userId);

            var projects = user.Projects.ToList();
            return (projects);
        }

        public void AddUserToProject(string userId, int projectId)
        {
            if (!IsUserOnProject(userId, projectId))
            {
                Project proj = db.Projects.Find(projectId);
                var newUser = db.Users.Find(userId);

                proj.Users.Add(newUser);
                db.SaveChanges();
            }
        }
```

```
 public void RemoveUserFromProject(string userId, int projectId)
     {
        if(IsUserOnProject(userId, projectId))
        {
        Project proj = db.Projects.Find(projectId);
        var delUser = db.Users.Find(userId);

        proj.Users.Remove(delUser);
        db.Entry(proj).State = EntityState.Modified; // just saves this obj instance.
        db.SaveChanges();
        }
     }

     public ICollection<ApplicationUser> UsersOnProject(int projectId)
     {
        return db.Projects.Find(projectId).Users;
     }

     public ICollection<ApplicationUser> UsersNotOnProject(int projectId)
     {
        return db.Users.Where(u => u.Projects.All(p => p.Id != projectId)).ToList();
     }
  }

  // Duplicate >>>>>>>>>>>>>>>>>>>>>>>>>>>>>

  public class UserProjectsHelper
  {
     private UserManager<ApplicationUser> manager = new UserManager<ApplicationUser>(new
UserStore<ApplicationUser>(new ApplicationDbContext()));
     private ApplicationDbContext db= new ApplicationDbContext();

     public bool IsOnProject(string userId, int projectId)
     {

     if(db.Projects.Find(projectId).Users.Contains(db.Users.Find(userId)))
        {
           return true;
        }
        return false;
     }
 public void AddUserToProject(string userId, int projectId)
        {
            if(!IsOnProject(userId, projectId))
            {
                var project = db.Projects.Find(projectId);
                    project.Users.Add(db.Users.Find(userId));
                db.Entry(project).State = EntityState.Modified; // just saves this obj instance.
                db.SaveChanges();
            }
        }

        public void RemoveUserFromProject(string userId, int projectId)
        {
            if (IsOnProject(userId, projectId))
            {
                var project = db.Projects.Find(projectId);
                project.Users.Remove(db.Users.Find(userId));
                db.Entry(project).State = EntityState.Modified; // just saves this obj instance.
                db.SaveChanges();
            }
        }

        public ICollection<ApplicationUser> ListUsersOnProject(int projectId)
        {
            return db.Projects.Find(projectId).Users;
        }

        public ICollection<Project> ListProjectsForUser(string userId)
        {
            return db.Users.Find(userId).Projects;
        }

        public ICollection<ApplicationUser> ListUsersNotOnProject(int projectId)
        {
            return db.Users.Where(u=> u.Projects.All(p => p.Id != projectId)).ToList();
        }
    }
}
```