

```

using System;
using System.Diagnostics;
using Microsoft.Office.Server.Diagnostics;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Microsoft.SharePoint;
using Microsoft.SharePoint.Utilities;
using Microsoft.SharePoint.WebControls;
using Microsoft.Office.Server.UserProfiles;
using System.Collections;
using System.Data;
using System.Security.Principal;
using System.Runtime.InteropServices;
using System.Web.UI.WebControls.WebParts;
using Microsoft.SharePoint.WebPartPages;
using Microsoft.SharePoint.Portal.WebControls;

namespace Reuben.TestProj.SharePoint.WebControls
{
    /// <summary>
    /// Create TEAM Control. Here, we create a new team site
    /// and add users to the current security of the site.
    /// </summary>
    public partial class CreateTEAMControl : System.Web.UI.UserControl
    {
        #region [LOCAL VARIABLES]

        /// <summary>
        /// ViewState variable to track Referrer Url
        /// </summary>
        public string ReferralUrl
        {
            get
            {
                // look for current page in ViewState
                object o = this.ViewState["_ReferralUrl"];
                if (o == null)
                    return string.Empty; // default page index of 0
                else
                    return (string)o;
            }
            set
            {
                this.ViewState["_ReferralUrl"] = value;
            }
        }

        /// <summary>
        /// ViewState variable to track Referrer Url
        /// </summary>
        public bool InEditMode
        {
            get
            {
                // look for current page in ViewState
                object o = this.ViewState["_InEditMode"];
                if (o == null)
                    return false; // default page index of 0
                else
                    return (bool)o;
            }
            set
            {
                this.ViewState["_InEditMode"] = value;
            }
        }

        private string teamSiteName = string.Empty;
        protected string teamSiteType = string.Empty;
        protected TextBox txtTeamName;
        protected TextBox txtDescription;
        protected TextBox txtWelcomeMessageTitle;

```

```

protected TextBox txtWelcomeMessageBody;
protected Label lblErrorMessage;
protected ImageButton imgCreate;
protected ImageButton imgCancel;
protected PeopleEditor peOwners;
protected PeopleEditor peMembers;
protected PeopleEditor peTeamLead;
protected TemplatePicker tpTemplate;
string m_ErrorMessage = string.Empty;
string m_CurrentUserName = string.Empty;

public List<string> Owners { get; set; }
public List<string> Members { get; set; }

public enum AssociatedGroupTypeEnum { Owners, Members, Vistors };

#endregion

/// <summary>
/// Page load logic
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
protected void Page_Load(object sender, EventArgs e)
{
    lblErrorMessage.Text = string.Empty;

    string url = System.Web.HttpContext.Current.Request.Url.ToString();
    if (url.ToLower().Contains("community"))
    {
        teamSiteType = "TestProj TEAM";
    }
    else
    {
        teamSiteType = "Agency TEAM";
    }

    if (!IsPostBack)
    {
        this.ReferralUrl = string.Empty;
        this.InEditMode = false;

        //If the user has landed on this page by clicking
        //the "Edit Membership" link on a team site.
        if (Request.QueryString["EditSiteMode"] != null)
        {
            if (!String.IsNullOrEmpty(Convert.ToString(Request.QueryString["EditSiteMode"])))
            {
                DisplayPageInEditMode();
            }
        }
    }
}

/// <summary>
/// Retrieve details of the site
/// </summary>
private void DisplayPageInEditMode()
{
    this.InEditMode = true;

    if (String.IsNullOrEmpty(this.ReferralUrl))
    {
        this.ReferralUrl = Request.UrlReferrer.AbsoluteUri;
    }

    string substringUrl = this.ReferralUrl.ToLower().Replace("/default.aspx", string.Empty);
    string[] siteParams = substringUrl.Split('/');

    if (siteParams.Length > 0)
    {
        teamSiteName = Convert.ToString(siteParams[siteParams.Length - 1]);
    }
}

```

```

        teamSiteName = teamSiteName.Replace("%20", " ");
    }

    PopulateSiteDetails(this.ReferralUrl, teamSiteName);
}

/// <summary>
/// Populate Site Details
/// </summary>
private void PopulateSiteDetails(string referrerUrl, string teamSiteName)
{
    SPSecurity.RunWithElevatedPrivileges(delegate()
    {
        using (SPSite newTeamSite = new SPSite(referrerUrl))
        {
            SPWeb parentWeb = SPContext.Current.Web;
            string siteDescription = parentWeb.Webs[teamSiteName].Description;

            #region [Set Team Name and Description]

            this.txtTeamName.Text = teamSiteName;
            this.txtTeamName.Enabled = false;

            this.txtDescription.Text = siteDescription;
            this.txtDescription.Enabled = false;

            #endregion

            #region [Set Welcome Message Title & Description, Users in Team Lead, Owners and
Members Group]

            using (SPWeb newTeamSiteWeb = newTeamSite.OpenWeb())
            {
                SPList announcementsList = newTeamSiteWeb.Lists["Announcements"];
                SPLListItem announcementItem = announcementsList.Items[0];

                this.txtWelcomeMessageTitle.Text = Convert.ToString(announcementItem["Title"]);
                this.txtWelcomeMessageTitle.Enabled = false;

                string welcomeMessageBody = Convert.ToString(announcementItem["Body"]);
                string[] welcomeMessageDetails = welcomeMessageBody.Split('>');
                string welcomeMessageText = welcomeMessageDetails[1].ToString();
                welcomeMessageText = welcomeMessageText.ToLower().Replace("</div", string.Empty);

                this.txtWelcomeMessageBody.Text = welcomeMessageText;
                this.txtWelcomeMessageBody.Enabled = false;

                #region [ADD OWNERS TO PICKER ENTRY FROM SITE]

                for (int x = 0; x < newTeamSiteWeb.AssociatedOwnerGroup.Users.Count; x++)
                {
                    try
                    {
                        PickerEntity pickerEntity = new PickerEntity();
                        pickerEntity.EntityData["Email"] =
newTeamSiteWeb.AssociatedOwnerGroup.Users[x].Email;
                        pickerEntity.EntityData["DisplayName"] =
newTeamSiteWeb.AssociatedOwnerGroup.Users[x].Name;
                        pickerEntity.DisplayText = Convert.ToString(pickerEntity.EntityData
["DisplayName"]);
                        pickerEntity.Key = Convert.ToString
(newTeamSiteWeb.AssociatedOwnerGroup.Users[x].LoginName);
                        pickerEntity.IsResolved = true;
                        peOwners.Entities.Add(pickerEntity);
                    }
                    catch (Exception ex)
                    {
                        Console.WriteLine(ex.ToString());
                    }
                }

                #endregion

                #region [ADD MEMBERS TO PICKER ENTRY FROM SITE]

```

```

        for (int x = 0; x < newTeamSiteWeb.AssociatedMemberGroup.Users.Count; x++)
        {
            try
            {
                PickerEntity pickerEntity = new PickerEntity();
                pickerEntity.EntityData["Email"] =
newTeamSiteWeb.AssociatedMemberGroup.Users[x].Email;
                pickerEntity.EntityData["DisplayName"] =
newTeamSiteWeb.AssociatedMemberGroup.Users[x].Name;
                pickerEntity.DisplayText = Convert.ToString(pickerEntity.EntityData
["DisplayName"]);
                pickerEntity.Key = Convert.ToString
(newTeamSiteWeb.AssociatedMemberGroup.Users[x].LoginName);
                pickerEntity.IsResolved = true;
                peMembers.Entities.Add(pickerEntity);
            }
            catch (Exception ex)
            {
                Console.WriteLine(ex.ToString());
            }
        }

        #endregion

        #region [ADD TEAM LEAD]

        for (int x = 0; x < 1; x++)
        {
            try
            {
                PickerEntity pickerEntity = new PickerEntity();
                pickerEntity.EntityData["Email"] = newTeamSiteWeb.SiteGroups
[newTeamSiteWeb + " Leaders"].Users[0].Email;
                pickerEntity.EntityData["DisplayName"] = newTeamSiteWeb.SiteGroups
[newTeamSiteWeb + " Leaders"].Users[0].Name;
                pickerEntity.DisplayText = Convert.ToString(pickerEntity.EntityData
["DisplayName"]);
                pickerEntity.Key = Convert.ToString(newTeamSiteWeb.SiteGroups
[newTeamSiteWeb + " Leaders"].Users[0].LoginName);
                pickerEntity.IsResolved = true;
                peTeamLead.Entities.Add(pickerEntity);
            }
            catch (Exception ex)
            {
                Console.WriteLine(ex.ToString());
            }
        }

        #endregion

        #region [SWAP IMAGE SUBMIT BUTTON]

        imgCreate.ImageUrl = "/_layouts/images/reuben/btn-save.png";

        #endregion
    }

    #endregion
});
}

/// <summary>
/// Function to redirect user to the previous page.
/// Here, if the control is loaded from the agencies site,
/// we redirect the user to the "Agencies Default Page".
/// Otherwise, we redirect the user to the "Community Default Page".
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
protected void OnCancelClick(object sender, ImageClickEventArgs e)
{
    string currentUrl = Request.Url.ToString();

```

```

    if (currentUrl.ToLower().Contains("agencies"))
    {
        Response.Redirect("/Agencies/Pages/default.aspx");
    }

    if (currentUrl.ToLower().Contains("community"))
    {
        Response.Redirect("/Community/Pages/default.aspx");
    }
}

/// <summary>
/// Function invoked when "Create" Team Site has been clicked.
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
protected void OnCreateTeamSiteClick(object sender, ImageClickEventArgs e)
{
    if (Request.QueryString["accountname"] != null)
    {
        m_CurrentUserName = Convert.ToString(Request.QueryString["accountname"]);
    }
    else
    {
        m_CurrentUserName = HttpContext.Current.User.Identity.Name;
    }

    if (Request.QueryString["EditSiteMode"] == null)
    {
        //Create new web object
        CreateNewWeb();
    }
    else
    {
        //Update existing web object
        UpdateExistingWeb();
    }
}

#region [UPDATE SITE]

/// <summary>
/// Update existing web
/// </summary>
private void UpdateExistingWeb()
{
    SPSecurity.RunWithElevatedPrivileges(delegate()
    {
        using (SPSite newTeamSite = new SPSite(this.ReferalUrl))
        {
            newTeamSite.AllowUnsafeUpdates = true;
            using (SPWeb newTeamSiteWeb = newTeamSite.OpenWeb())
            {
                newTeamSiteWeb.AllowUnsafeUpdates = true;
                #region [Add users to the owner group / Remove current user from Member Group]

                try
                {
                    //Remove all existing users
                    for (int x = 0; x < newTeamSiteWeb.AssociatedOwnerGroup.Users.Count; x++)
                    {
                        newTeamSiteWeb.AssociatedOwnerGroup.Users.Remove(x);
                    }

                    foreach (PickerEntity entry in peOwners.ResolvedEntities)
                    {
                        string userName = entry.Key;
                        string userDisplayName = string.Empty;
                        string userEmailAddress = string.Empty;

                        if (entry.EntityData["Email"] != null)
                        {
                            userEmailAddress = entry.EntityData["Email"].ToString();

```

```

    }

    if (entry.EntityData["DisplayName"] != null)
    {
        userDisplayName = entry.EntityData["DisplayName"].ToString();
    }

    newTeamSiteWeb.AssociatedOwnerGroup = newTeamSiteWeb.SiteGroups
[newTeamSiteWeb.Name + " Owners"];
    newTeamSiteWeb.AssociatedOwnerGroup.Users.Add(userName, userEmailAddress,
userDisplayName, null);
    }
}
catch (Exception ex)
{
    Console.WriteLine(ex.ToString());
}
}

#endregion

#region [Add users to the members group]

try
{
    //Remove all existing users
    for (int x = 0; x < newTeamSiteWeb.AssociatedMemberGroup.Users.Count; x++)
    {
        newTeamSiteWeb.AssociatedMemberGroup.Users.Remove(x);
    }

    foreach (PickerEntity entry in peMembers.ResolvedEntities)
    {
        string userName = entry.Key;
        string userDisplayName = string.Empty;
        string userEmailAddress = string.Empty;

        if (entry.EntityData["Email"] != null)
        {
            userEmailAddress = entry.EntityData["Email"].ToString();
        }

        if (entry.EntityData["DisplayName"] != null)
        {
            userDisplayName = entry.EntityData["DisplayName"].ToString();
        }

        newTeamSiteWeb.AssociatedMemberGroup = newTeamSiteWeb.SiteGroups
[newTeamSiteWeb.Name + " Members"];
        newTeamSiteWeb.AssociatedMemberGroup.Users.Add(userName,
userEmailAddress, userDisplayName, null);
    }
}
catch (Exception ex)
{
    Console.WriteLine(ex.ToString());
}
}

#endregion

#region [Add leaders to the leader group]

try
{
    //Remove all existing users
    for (int x = 0; x < newTeamSiteWeb.SiteGroups[newTeamSiteWeb.Name + "
Leaders"].Users.Count; x++)
    {
        newTeamSiteWeb.SiteGroups[newTeamSiteWeb.Name + " Leaders"].Users.Remove
(x);
    }

    foreach (PickerEntity entry in peTeamLead.ResolvedEntities)
    {
        string userName = entry.Key;
        string userDisplayName = string.Empty;

```

```

        string userEmailAddress = string.Empty;

        if (entry.EntityData["Email"] != null)
        {
            userEmailAddress = entry.EntityData["Email"].ToString();
        }

        if (entry.EntityData["DisplayName"] != null)
        {
            userDisplayName = entry.EntityData["DisplayName"].ToString();
        }

        newTeamSiteWeb.SiteGroups[newTeamSiteWeb.Name + " Leaders"].Users.Add
(userName, userEmailAddress, userDisplayName, null);
    }

    SPLimitedWebPartManager webPartManager =
newTeamSiteWeb.GetLimitedWebPartManager(newTeamSiteWeb.Url + "/Default.aspx",
PersonalizationScope.Shared);

    if (webPartManager.WebParts.Count > 0)
    {
        for (int x = 0; x < webPartManager.WebParts.Count; x++)
        {
            switch (webPartManager.WebParts[x].Title.ToLower())
            {
                //Change Team Lead Property
                case "team lead":

                    try
                    {
                        ContactFieldControl contactFieldWebPart =
(ContactFieldControl)webPartManager.WebParts[x];
                        contactFieldWebPart.CachePerUser = false;
                        CacheableWebPart cachWp = (CacheableWebPart)
contactFieldWebPart;
                        webPartManager.CacheInvalidate(contactFieldWebPart,
Storage.Shared);
                        contactFieldWebPart.PartCacheInvalidate();
                        webPartManager.SaveChanges(contactFieldWebPart);

                        PickerEntity pickerEntity = (PickerEntity)
contactFieldWebPart.Contact = Convert.ToInt32
(pickerEntity.EntityData["SPUserID"]);
                        webPartManager.SaveChanges(contactFieldWebPart);
                    }
                    catch (Exception ex)
                    {
                        Console.WriteLine(ex.ToString());
                    }

                    break;
            }
        }
    }
    catch (Exception ex)
    {
        Console.WriteLine(ex.ToString());
    }

    #endregion

    //Update Team Site
    newTeamSiteWeb.Update();
}
});

Response.Redirect(this.ReferalUrl);
}

#endregion

```

```

#region [CREATE SITE IN CURRENT PATH USING .STP]

/// <summary>
/// Create new web
/// </summary>
void CreateNewWeb()
{
    SPSecurity.RunWithElevatedPrivileges(delegate()
    {
        using (SPLongOperation operation = new SPLongOperation(this.Page))
        {
            SPWeb newTeamSite = null;
            string addressComplete = string.Empty;
            string queryString = string.Empty;
            operation.Begin();
            operation.LeadingHTML = "Please wait while your Site is provisioned";
            operation.TrailingHTML = "Your current operation is currently being executed.

Please be patient";

            try
            {
                //Get the current site collection
                using (SPSite site = SPContext.Current.Site)
                {
                    //Find the selected template in the site collection gallery
                    SPWebTemplateCollection templates = site.GetCustomWebTemplates(1033);
                    SPWebTemplate template = templates["TestProj Team Site"];

                    //Get our current web, our new site will be a direct descendant of the
current site.

                    SPWeb parentWeb = SPContext.Current.Web;
                    parentWeb.AllowUnsafeUpdates = true;

                    newTeamSite = CheckForExistingSite(parentWeb, txtTeamName.Text);

                    if (newTeamSite == null)
                    {
                        newTeamSite = parentWeb.Webs.Add(txtTeamName.Text, txtTeamName.Text,
txtDescription.Text, 1033, template, true, false);

                        //Insert new web into Site Directory
                        InsertWebIntoSiteDirectory();
                    }

                    using (SPWeb websiteRoot = SPContext.Current.Site.RootWeb)
                    {
                        SPGroupCollection groupCollection = websiteRoot.SiteGroups;
                        SPUser spUser = newTeamSite.Users[0];
                        SPMember spMember = newTeamSite.Users[0];

                        #region [CREATE MEMBERS GROUP FOR SITE]

                        try
                        {
                            bool memberGroupFound = false;

                            //Check if a "Members" group exists for the site
                            //If it doesn't then create one using the following
                            //nomenclature: "Team Site Name" + " Members"
                            for (int x = 0; x < newTeamSite.SiteGroups.Count; x++)
                            {
                                if (newTeamSite.SiteGroups[x].Name == (newTeamSite.Name + "
Members"))
                                {
                                    memberGroupFound = true;
                                }
                            }

                            if (!memberGroupFound)
                            {
                                newTeamSite.SiteGroups.Add(newTeamSite.Name + " Members",
spMember, spUser, null);
                            }
                        }
                    }
                }
            }
        }
    });
}

```

```

catch (Exception ex)
{
    Console.WriteLine(ex.ToString());
}

#endregion

#region [CREATE OWNER GROUP FOR SITE]

try
{
    bool ownerGroupFound = false;

    //Check if a "Owners" group exists for the site
    //If it doesn't then create one using the following
    //nomenclature: "Team Site Name" + " Owners"
    for (int x = 0; x < newTeamSite.SiteGroups.Count; x++)
    {
        if (newTeamSite.SiteGroups[x].Name == (newTeamSite.Name + "
Owners"))
        {
            ownerGroupFound = true;
        }
    }

    if (!ownerGroupFound)
    {
        newTeamSite.SiteGroups.Add(newTeamSite.Name + " Owners",
spMember, spUser, null);
    }
}
catch (Exception ex)
{
    Console.WriteLine(ex.ToString());
}

#endregion

#region [CREATE LEADERS GROUP FOR SITE]

bool leadershipGroupFound = false;

//Check if a "Leadership" group exists for the site
//If it doesn't then create one using the following
//nomenclature: "Team Site Name" + " Leaders"
for (int x = 0; x < newTeamSite.SiteGroups.Count; x++)
{
    if (newTeamSite.SiteGroups[x].Name == (newTeamSite.Name + "
Leaders"))
    {
        leadershipGroupFound = true;
    }
}

try
{
    if (!leadershipGroupFound)
    {
        newTeamSite.SiteGroups.Add(newTeamSite.Name + " Leaders",
spMember, spUser, null);
    }
}
catch (Exception ex)
{
    Console.WriteLine(ex.ToString());
}

#endregion

try
{
    //Create our new web, make sure to dispose of the reference
    using (newTeamSite)
    {

```

```

from Member Group]

#region [Add users to the owners group / Remove current user

try
{
    foreach (PickerEntity entry in peOwners.ResolvedEntities)
    {
        string userName = entry.Key;
        string userDisplayName = string.Empty;
        string userEmailAddress = string.Empty;

        if (entry.EntityData["Email"] != null)
        {
            userEmailAddress = entry.EntityData

["Email"].ToString();

        }

        if (entry.EntityData["DisplayName"] != null)
        {
            userDisplayName = entry.EntityData

["DisplayName"].ToString();

        }

        newTeamSite.AssociatedOwnerGroup =
newTeamSite.SiteGroups[newTeamSite.Name + " Owners"];
        newTeamSite.AssociatedOwnerGroup.Users.Add(userName,
userEmailAddress, userDisplayName, null);
    }

    if (newTeamSite.AssociatedOwnerGroup != null)
    {
        SPRoleAssignment assignment = new SPRoleAssignment

(newTeamSite.AssociatedOwnerGroup);
        SPRoleDefinition role = (SPRoleDefinition)
newTeamSite.RoleDefinitions["Full Control"];

        assignment.RoleDefinitionBindings.Add(role);
        newTeamSite.RoleAssignments.Add(assignment);
        newTeamSite.Update();
    }

#region [DELETE ANY EXISTING USERS]

try
{
    bool ownerUserFound = false;
    for (int k = 0; k < peOwners.ResolvedEntities.Count;
k++)
    {
        for (int x = 0; x <
newTeamSite.AssociatedOwnerGroup.Users.Count; x++)
        {
            string userNameFromGroup =
newTeamSite.AssociatedOwnerGroup.Users[x].Name.ToString().ToLower();
            string userNameFromPicker = ((PickerEntity)
peOwners.ResolvedEntities[k]).EntityData["DisplayName"].ToString().ToLower();

            if(userNameFromGroup == userNameFromPicker)
            {
                ownerUserFound = true;
            }
        }

        if(!ownerUserFound)
        {
            newTeamSite.AssociatedOwnerGroup.Users.Remove
(((PickerEntity)peOwners.ResolvedEntities[k]).EntityData["LoginName"].ToString());
        }
    }
}
catch (Exception ex)
{
    Console.Write(ex.ToString());
    this.lblErrorMessage.Text = ex.ToString();
}
}
}

```

```

        #endregion
    }
    catch (Exception ex)
    {
        Console.WriteLine(ex.ToString());
    }

    #endregion

    #region [Add users to the members group]

    try
    {
        foreach (PickerEntity entry in
peMembers.ResolvedEntities)
        {
            string userName = entry.Key;
            string userDisplayName = string.Empty;
            string userEmailAddress = string.Empty;

            if (entry.EntityData["Email"] != null)
            {
                userEmailAddress = entry.EntityData
["Email"].ToString();

            }

            if (entry.EntityData["DisplayName"] != null)
            {
                userDisplayName = entry.EntityData
["DisplayName"].ToString();

            }

            newTeamSite.AssociatedMemberGroup =
newTeamSite.SiteGroups[newTeamSite.Name + " Members"];
            newTeamSite.AssociatedMemberGroup.Users.Add(userName,
userEmailAddress, userDisplayName, null);
        }

        if (newTeamSite.AssociatedMemberGroup != null)
        {
            SPRoleAssignment assignment = new SPRoleAssignment

            SPRoleDefinition role = (SPRoleDefinition)

            assignment.RoleDefinitionBindings.Add(role);
            newTeamSite.RoleAssignments.Add(assignment);
            newTeamSite.Update();
        }

        #region [DELETE ANY EXISTING USERS]

        try
        {
            bool memberUserFound = false;
            for (int k = 0; k < peMembers.ResolvedEntities.Count;
k++)
            {
                for (int x = 0; x <
newTeamSite.AssociatedMemberGroup.Users.Count; x++)
                {
                    string userNameFromGroup =
newTeamSite.AssociatedMemberGroup.Users[x].Name.ToString().ToLower();
                    string userNameFromPicker = ((PickerEntity)
peMembers.ResolvedEntities[k]).EntityData["DisplayName"].ToString().ToLower();

                    if (userNameFromGroup == userNameFromPicker)
                    {
                        memberUserFound = true;
                    }
                }

                if (!memberUserFound)
                {
                    newTeamSite.AssociatedMemberGroup.Users.Remove(((PickerEntity)peMembers.ResolvedEntities[k]).EntityData

```

```

["LoginName"].ToString());
    }
}
catch (Exception ex)
{
    Console.WriteLine(ex.ToString());
    this.lblErrorMessage.Text = ex.ToString();
}
}
#endregion
}
catch (Exception ex)
{
    Console.WriteLine(ex.ToString());
}
}
#endregion

#region [Add leaders to the leader group]
try
{
    foreach (PickerEntity entry in
peTeamLead.ResolvedEntities)
    {
        string userName = entry.Key;
        string userDisplayName = string.Empty;
        string userEmailAddress = string.Empty;

        if (entry.EntityData["Email"] != null)
        {
            userEmailAddress = entry.EntityData
["Email"].ToString();
        }

        if (entry.EntityData["DisplayName"] != null)
        {
            userDisplayName = entry.EntityData
["DisplayName"].ToString();
        }

        newTeamSite.SiteGroups[newTeamSite.Name + "
Leaders"].Users.Add(userName, userEmailAddress, userDisplayName, null);
    }

    newTeamSite.SiteGroups[newTeamSite.Name + "
Leaders"].Users.Remove(System.Environment.UserDomainName + "\\\" + System.Environment.UserName);

    if (newTeamSite.AssociatedMemberGroup != null)
    {
        SPRoleAssignment assignment = new SPRoleAssignment
(newTeamSite.SiteGroups[newTeamSite.Name + " Leaders"]);
        SPRoleDefinition role = (SPRoleDefinition)
newTeamSite.RoleDefinitions["Full Control"];

        assignment.RoleDefinitionBindings.Add(role);
        newTeamSite.RoleAssignments.Add(assignment);
        newTeamSite.Update();
    }

    try
    {
        bool leadUserFound = false;
        for (int k = 0; k <
peTeamLead.ResolvedEntities.Count; k++)
        {
            for (int x = 0; x < newTeamSite.SiteGroups
[newTeamSite.Name + " Leaders"].Users.Count; x++)
            {
                string userNameFromGroup =
newTeamSite.SiteGroups[newTeamSite.Name + " Leaders"].Users[x].Name.ToString().ToLower();
                string userNameFromPicker = ((PickerEntity)
peTeamLead.ResolvedEntities[k]).EntityData["DisplayName"].ToString().ToLower();

                if (userNameFromGroup == userNameFromPicker)

```



```

private void SetupTeamLeadAndOwners(SPWeb newTeamSite)
{
    using (newTeamSite)
    {
        SPLimitedWebPartManager webPartManager = newTeamSite.GetLimitedWebPartManager
(newTeamSite.Url + "/Default.aspx", PersonalizationScope.Shared);

        if (webPartManager.WebParts.Count > 0)
        {
            for (int x = 0; x < webPartManager.WebParts.Count; x++)
            {
                switch (webPartManager.WebParts[x].Title.ToLower())
                {
                    //Change Team Owner Property
                    case "team owners":

                        try
                        {
                            Microsoft.SharePoint.WebPartPages.MembersWebPart membershipWebPart =
(Microsoft.SharePoint.WebPartPages.MembersWebPart)webPartManager.WebParts[x];
                            membershipWebPart.MembershipGroupId = newTeamSite.SiteGroups
[newTeamSite.Name + " Owners"].ID;
                            webPartManager.SaveChanges(membershipWebPart);
                        }
                        catch (Exception ex)
                        {
                            Console.WriteLine(ex.ToString());
                        }

                        break;

                    //Change Team Lead Property
                    case "team lead":

                        try
                        {
                            ContactFieldControl contactFieldWebPart = (ContactFieldControl)
webPartManager.WebParts[x];
                            PickerEntity pickerEntity = (PickerEntity)peTeamLead.ResolvedEntities
[0];
                            contactFieldWebPart.Contact = Convert.ToInt32(pickerEntity.EntityData
["SPUserID"]);
                            webPartManager.SaveChanges(contactFieldWebPart);
                        }
                        catch (Exception ex)
                        {
                            Console.WriteLine(ex.ToString());
                        }

                        break;

                }
            }
        }
    }

    /// <summary>
    /// Create a customized welcome message for the site.
    /// Here, we modify the first announcement item found within the
    /// "Announcements" list.
    /// TO-DO: We should (if time permits) modify the architecture used here.
    /// Currently, the Announcements web part assumes that only 1 announcement item
    /// exists in the Announcements list. If multiple announcements are added to the
    /// list, the web part doesn't display information correctly. We need to tweak the
    /// Site Template to display only the "Last Modified and Approved" Announcement
    /// within the site's Annoucement list.
    /// </summary>
    /// <param name="newTeamSite"></param>
    private void SetupWelcomeMessage(SPWeb teamSite)
    {
        try
        {
            using (teamSite)
            {
                if (!String.IsNullOrEmpty(this.txtWelcomeMessageTitle.Text) && (!

```

```

String.IsNullOrEmpty(this.txtWelcomeMessageTitle.Text))
    {
        SPList announcementsList = teamSite.Lists["Announcements"];

        //Clear all announcements in the list currently.
        if (announcementsList.ItemCount > 0)
        {
            for (int x = 0; x < announcementsList.ItemCount; x++)
            {
                announcementsList.Items.Delete(x);
            }

            //Add new announcement item based on value from user
            SPListItem announcementItem = announcementsList.Items.Add();
            announcementItem["Title"] = this.txtWelcomeMessageTitle.Text;
            announcementItem["Body"] = this.txtWelcomeMessageBody.Text;
            announcementItem.Update();
        }
    }
}
catch (Exception ex)
{
    Console.WriteLine(ex.ToString());
}
}

/// <summary>
/// Check if a site with the same name already exists.
/// We donot allow sites with same name created.
/// </summary>
/// <param name="parentWeb"></param>
/// <param name="TEAMName"></param>
/// <returns></returns>
SPWeb CheckForExistingSite(SPWeb parentWeb, string TEAMName)
{
    SPWeb web = null;

    List<string> names = new List<string>(parentWeb.Webs.Names);
    if (names.Contains(TEAMName))
    {
        web = parentWeb.Webs[TEAMName];
    }

    return web;
}

/// <summary>
/// Setup a new web object
/// </summary>
/// <param name="newTeamSite"></param>
void SetupNewWeb(SPWeb newTeamSite)
{
    newTeamSite.Navigation.TopNavigationBar.Navigation.UseShared = true;
}

/// <summary>
/// Insert web into site directory
/// </summary>
void InsertWebIntoSiteDirectory()
{
    SPWeb rootWeb = SPContext.Current.Site.RootWeb;

    try
    {
        if (rootWeb.AllProperties.ContainsKey("DefaultSiteDirectorySiteId"))
        {
            string webId = rootWeb.AllProperties["DefaultSiteDirectoryWebId"].ToString();

            Guid siteDirectoryId = new Guid(webId);
            using (SPWeb siteDirectory = rootWeb.Webs[siteDirectoryId])
            {
                siteDirectory.AllowUnsafeUpdates = true;

                SPList sitesList = siteDirectory.Lists["Sites"];
            }
        }
    }
}

```

```

        SPListItem newSiteListing = sitesList.Items.Add();

        UpdateListColumn(newSiteListing, "Title", txtTeamName.Text);
        UpdateListColumn(newSiteListing, "Description", txtDescription.Text);
        UpdateListColumn(newSiteListing, "URL", BuildSiteUrl());

        newSiteListing.Update();
    }
}
}
}
catch (Exception ex)
{
    string message = string.Format("Could not list site in the site directory. Please
contact an administrator. Error Message was: '{0}'\nStack Trace:\n{1}", ex.Message, ex.StackTrace);
    PortalLog.LogString(message);
    Debug.WriteLine(message);
    //throw ex;
}
}
}
SPFieldUrlValue BuildSiteUrl()
{
    SPFieldUrlValue urlValue = new SPFieldUrlValue(SPHttpUtility.HtmlEncode(string.Format("{0}/
{1}", SPContext.Current.Web.Url, txtTeamName.Text)));
    urlValue.Description = txtDescription.Text;
    return urlValue;
}

#endregion

/// <summary>
/// Update list column
/// </summary>
/// <param name="listItem"></param>
/// <param name="columnName"></param>
/// <param name="value"></param>
void UpdateListColumn(SPListItem listItem, string columnName, object value)
{
    if (listItem.Fields.ContainsField(columnName) == false)
    {
        throw new ApplicationException(string.Format("Invalid list configuration, column '{0}'
does not exist.", columnName));
    }

    SPField listField = listItem.Fields[columnName];

    object fieldValue = null;

    switch (listField.Type)
    {
        case SPFieldType.Text:
        case SPFieldType.Note:
        case SPFieldType.Choice:
            if (value is KeyValuePair<int, string>)
            {
                fieldValue = ((KeyValuePair<int, string>)value).Value;
            }
            else
            {
                fieldValue = value.ToString();
            }
            break;

        case SPFieldType.URL:
            if (value is KeyValuePair<string, string>)
            {
                KeyValuePair<string, string> tuple = (KeyValuePair<string, string>)value;
                SPFieldUrlValue urlValue = new SPFieldUrlValue(tuple.Value);
                urlValue.Description = tuple.Key;
                fieldValue = urlValue;
            }
            else if (value is SPFieldUrlValue)
            {
                fieldValue = value;
            }
            else

```

```

        {
            fieldValue = value.ToString();
        }
        break;

    case SPFieldType.Boolean:
        fieldValue = Convert.ToBoolean(value);
        break;

    case SPFieldType.Lookup:
        if (value is KeyValuePair<int, string>)
        {
            KeyValuePair<int, string> kvp = (KeyValuePair<int, string>)value;

            //TODO: Change this to an SPFieldLookupValue object.
            //fieldValue = ((KeyValuePair<int, string>)value).Key;
            fieldValue = new SPFieldLookupValue(kvp.Key, kvp.Value);
        }
        else if (value is List<KeyValuePair<int, string>>)
        {
            string val = string.Empty;
            SPFieldLookupValueCollection values = new SPFieldLookupValueCollection();

            foreach (KeyValuePair<int, string> pair in ((List<KeyValuePair<int, string>>)
value))
            {
                values.Add(new SPFieldLookupValue(pair.Key, pair.Value));
            }
            fieldValue = values;
        }
        break;

    default:
        fieldValue = value;
        break;
    }

    listItem[columnName] = fieldValue;
}
}
}

```