Лабораторная работа № 4 Создание многостраничного приложения на Silverlight	Студент, группа	Голубев А. В.
	Дата выполнения	
	Подпись	
	Дата отчёта	
	Оценка	
	Подпись	

Цель работы: получение общих сведений о технологии Silverlight, получение практических навыков создания web-приложений, использующий Silverlight.

Задачи:

1. Создать web-приложение на ASP.NET
2. Создать многостраничную Silverlight-форму
3. Добавить созданную форму на разные страницы web-приложения на ASP.NET
4. Добавить на Silverlight-форму элементы управления и обработчики событий.
5. Соотнести разные страницы Silverlight-формы с разными страницами web-приложения на ASP.NET

Скриншоты:

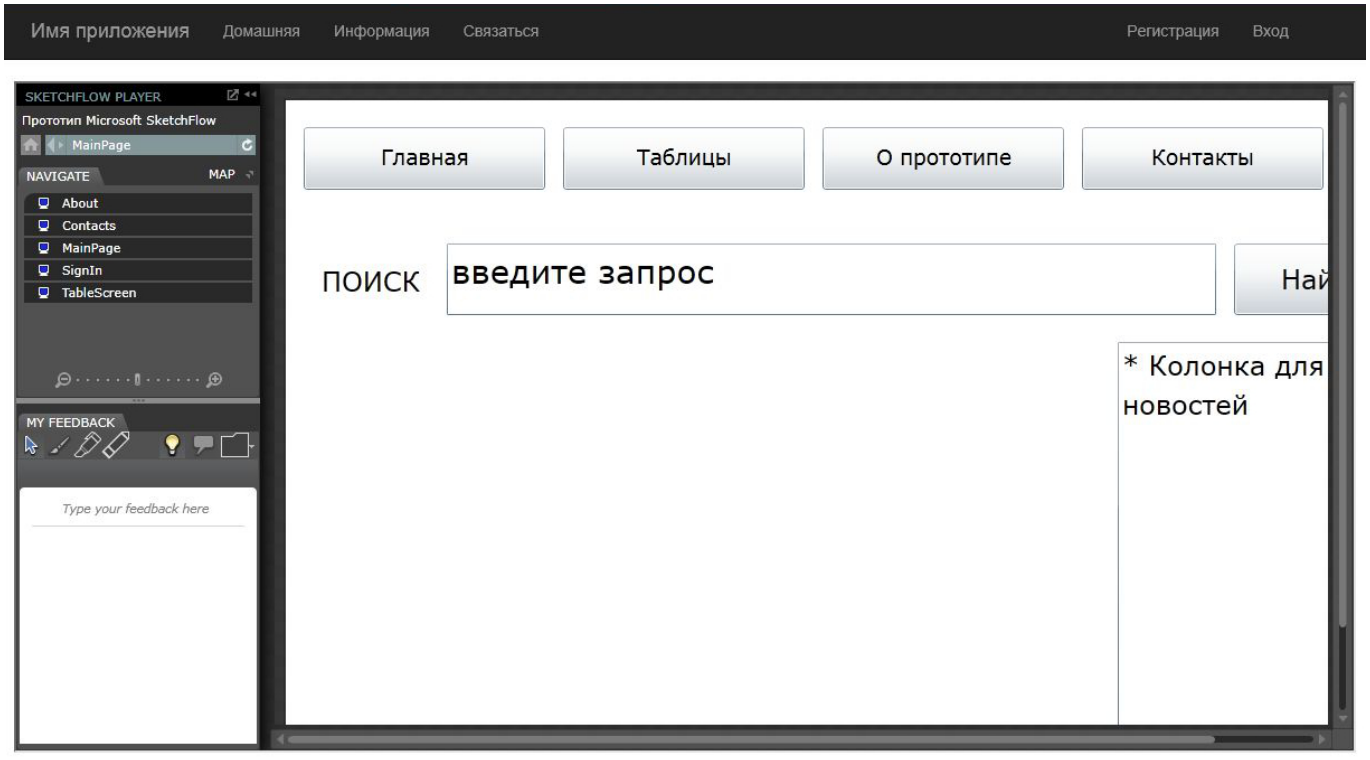

Рисунок 1 — Главная страница с Silverlight-формой

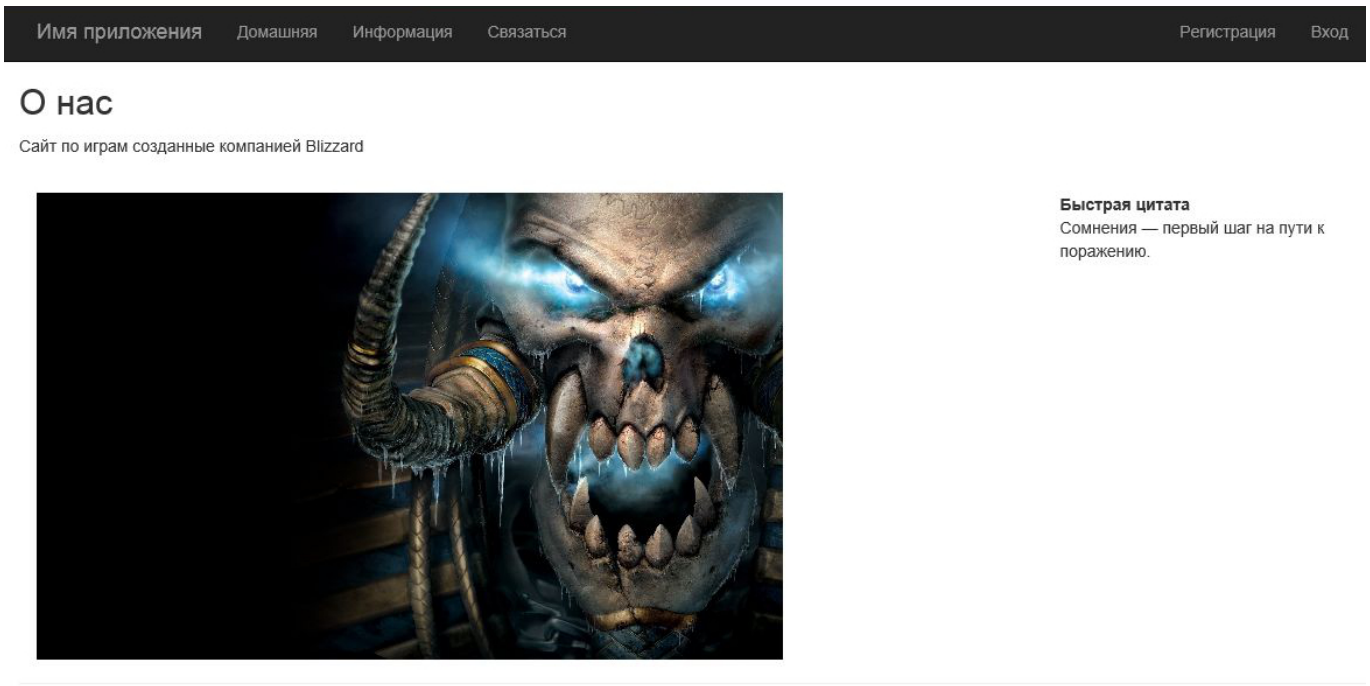

Рисунок 2 — О сайте

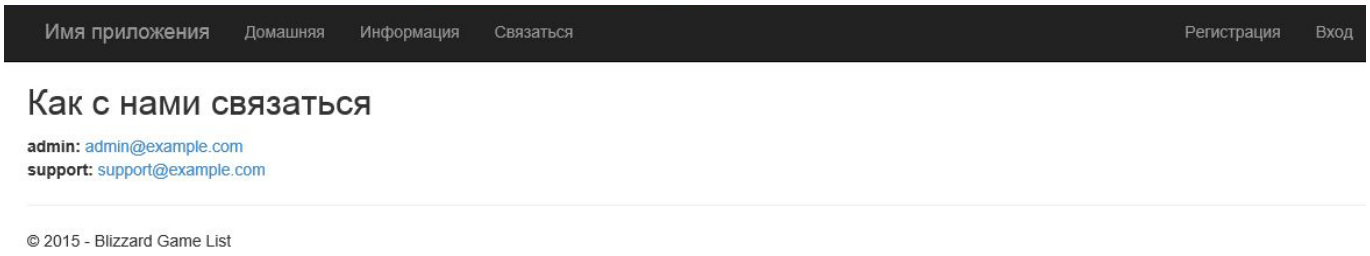

Рисунок 3 — Контактная информация

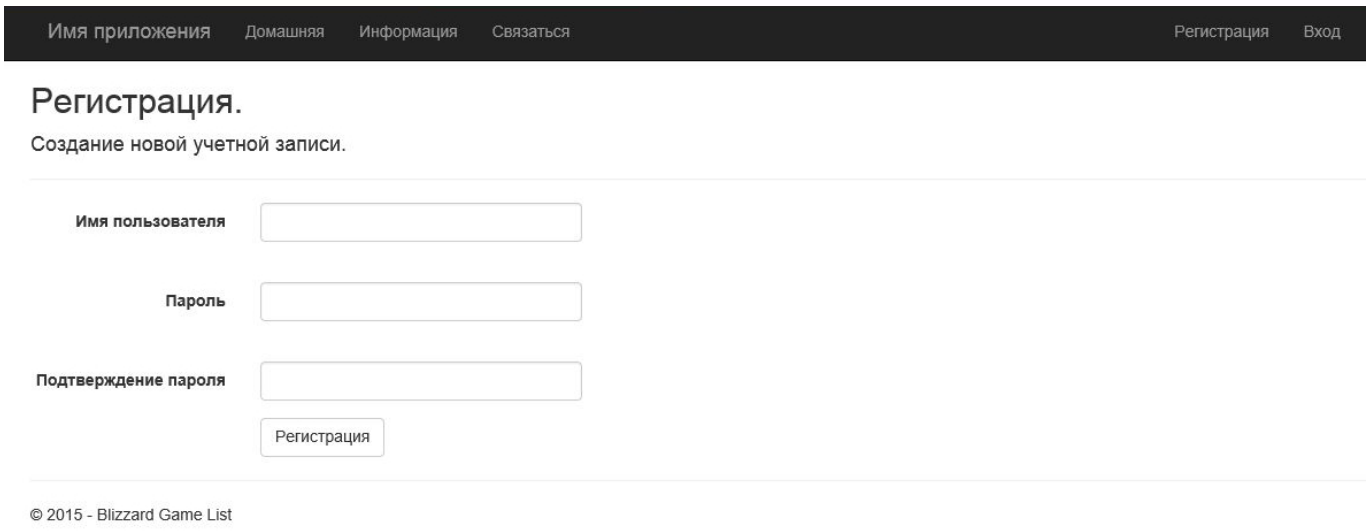

Рисунок 4 — Регистрация

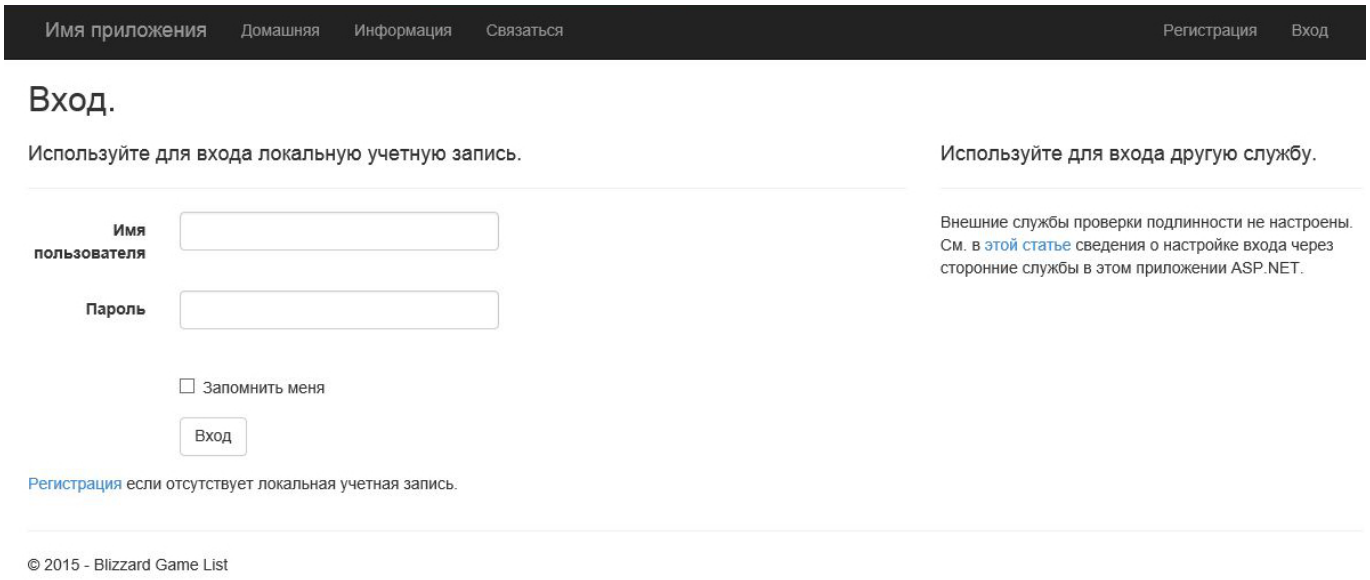

Рисунок 5 — Вход

Исходный код:

Страница About

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

public partial class About : Page
{
    static int image_number = 1;
    static int phrase_number = 1;
    static int counter = 0;
    static Random rnd = new Random();
    string[] phrases = {
        "После всего ада, через который мы прошли, я знаю одно: мы можем положиться "+
        "друг на друга и достигнуть цели, или погибнуть, если такова цена... Потому что есть вещи, "+
        "за которые стоит бороться!",
        "Сомнения — первый шаг на пути к поражению.",
        "Сначала стреляй, потом сомневайся.",
        "Ты мне нравишься. Тебя я убью последним.",
        "Единственное, что ты должен чувствовать, стреляя в кого-то— — это отдача.",
        "Огонь снимает с тела кожу, а с души — грех. Огонь выжигает всю грязь, а мама с детства учила меня чистоте.",
        "Ревность — удел слабых духом. Она заполняет сознание распутными мыслями, не оставляя места для остального.",
        "Да мне цены нет! Даже страховку оформить не могу!",
        "Скромность украшает, но оставляет голодным.",
        "«Мы строим — вы отдыхаете». Именно под таким девизом работает союз эльфовстроителей—. Вот и отдыхай, и не "+
        "мешай нам строить!",
        "Ты же командир! Не делай умное лицо!",
        "И солнце ярче блещет,<brИ> веселей пейзаж,<brКогда> в желудке плещется<br>C2H5OH!",
        "— Скажика—, дядя, ведь не даром?<br—> Ну конечно, не даром!",
        "Не могу стоять, пока другие работают... Пойду полежу.",
        "— Рожденный ползать упасть не может!",
        "— Кто к нам с мечем придет, тех проще застрелить!",
        "— Первое условие бессмертия — смерть.",
        "— Быстро и дешево выполню любую халтуру!",
        "— Если смертные достают, то достают до смерти.",
        "— Все не так плохо, как вам кажется. Все намного, намного хуже.",
        "— Я не злопамятный, я запишу.",
        "— И никакие мы не сектанты, мы просто группа ошалелых вооружённых фанатиков!",
        "— Гоблин в доспехах — консервы, завтрак дракона."
    };
    int image_count = 31;
    protected void Page_Load(object sender, EventArgs e)
    {
        Image1.ImageUrl = "images/image_" + image_number + ".jpg";
        Label1.Text = "<bБыстрая> цитата</b><br>" + phrases[phrase_number];
    }
    protected void Timer1_Tick1(object sender, EventArgs e)
    {
        image_number = rnd.Next(1, image_count);
        if ( counter % 2 == 0 ) {
            phrase_number = rnd.Next(1, phrases.Length);
        }
        counter++;
    }
}
```

Страница Login

```
using Microsoft.AspNet.Identity;
using Microsoft.Owin.Security;
using System;
using System.Web;
using System.Web.UI;
using Лабы;

public partial class Account_Login : Page
{
        protected void Page_Load(object sender, EventArgs e)
```

```
11          {
12              RegisterHyperLink.NavigateUrl = "Register";
13              OpenAuthLogin.ReturnUrl = Request.QueryString["ReturnUrl"];
14              var returnUrl = HttpUtility.UrlEncode(Request.QueryString["ReturnUrl"]);
15              if (!String.IsNullOrEmpty(returnUrl))
16              {
17                  RegisterHyperLink.NavigateUrl += "?ReturnUrl=" + returnUrl;
18              }
19          }
20
21          protected void LogIn(object sender, EventArgs e)
22          {
23              if (IsValid)
24              {
25                  // Validate the user password
26                  var manager = new UserManager();
27                  ApplicationUser user = manager.Find(UserName.Text, Password.Text);
28                  if (user != null)
29                  {
30                      IdentityHelper.SignIn(manager, user, RememberMe.Checked);
31                      IdentityHelper.RedirectToReturnUrl(Request.QueryString["ReturnUrl"], Response);
32                  }
33                  else
34                  {
35                      FailureText.Text = "Invalid username or password.";
36                      ErrorMessage.Visible = true;
37                  }
38              }
39          }
40  }
```

Страница Register

```
1   using Microsoft.AspNet.Identity;
2   using System;
3   using System.Linq;
4   using System.Web.UI;
5   using Лабы;
6
7   public partial class Account_Register : Page
8   {
9       protected void CreateUser_Click(object sender, EventArgs e)
10      {
11          var manager = new UserManager();
12          var user = new ApplicationUser() { UserName = UserName.Text };
13          IdentityResult result = manager.Create(user, Password.Text);
14          if (result.Succeeded)
15          {
16              IdentityHelper.SignIn(manager, user, isPersistent: false);
17              IdentityHelper.RedirectToReturnUrl(Request.QueryString["ReturnUrl"], Response);
18          }
19          else
20          {
21              ErrorMessage.Text = result.Errors.FirstOrDefault();
22          }
23      }
24  }
```

Вспомогательный код для регистрации

```
1   using Microsoft.AspNet.Identity;
2   using System;
3   using System.Collections.Generic;
4   using System.Linq;
5   using Лабы;
6
7   public partial class Account_Manage : System.Web.UI.Page
8   {
9       protected string SuccessMessage
10      {
11          get;
12          private set;
```

```
13      }
14
15      protected bool CanRemoveExternalLogins
16      {
17          get;
18          private set;
19      }
20
21      private bool HasPassword(UserManager manager)
22      {
23          var user = manager.FindById(User.Identity.GetUserId());
24          return (user != null && user.PasswordHash != null);
25      }
26
27      protected void Page_Load()
28      {
29          if (!IsPostBack)
30          {
31              // Определите разделы для отображения
32              UserManager manager = new UserManager();
33              if (HasPassword(manager))
34              {
35                  changePasswordHolder.Visible = true;
36              }
37              else
38              {
39                  setPassword.Visible = true;
40                  changePasswordHolder.Visible = false;
41              }
42              CanRemoveExternalLogins = manager.GetLogins(User.Identity.GetUserId()).Count() > 1;
43
44              // Отобразить сообщение об успехе
45              var message = Request.QueryString["m"];
46              if (message != null)
47              {
48                  // Извлечь строку запроса из действия
49                  Form.Action = ResolveUrl("~/Account/Manage");
50
51                  SuccessMessage =
52                      message == "ChangePwdSuccess" ? "Ваш пароль изменен."
53                      : message == "SetPwdSuccess" ? "Пароль задан."
54                      : message == "RemoveLoginSuccess" ? "Учетная запись удалена."
55                      : String.Empty;
56                  successMessage.Visible = !String.IsNullOrEmpty(SuccessMessage);
57              }
58          }
59      }
60
61      protected void ChangePassword_Click(object sender, EventArgs e)
62      {
63          if (IsValid)
64          {
65              UserManager manager = new UserManager();
66              IdentityResult result = manager.ChangePassword(User.Identity.GetUserId(), CurrentPassword.Text
        , NewPassword.Text);
67              if (result.Succeeded)
68              {
69                  Response.Redirect("~/Account/Manage?m=ChangePwdSuccess");
70              }
71              else
72              {
73                  AddErrors(result);
74              }
75          }
76      }
77
78      protected void SetPassword_Click(object sender, EventArgs e)
79      {
80          if (IsValid)
81          {
82              // Создание информации о локальном имени входа и связывание локальной учетной записи с пользователем
83              UserManager manager = new UserManager();
84              IdentityResult result = manager.AddPassword(User.Identity.GetUserId(), password.Text);
85              if (result.Succeeded)
86              {
87                  Response.Redirect("~/Account/Manage?m=SetPwdSuccess");
```

```
            }
            else
            {
                AddErrors(result);
            }
        }
    }

    public IEnumerable<UserLoginInfo> GetLogins()
    {
        UserManager manager = new UserManager();
        var accounts = manager.GetLogins(User.Identity.GetUserId());
        CanRemoveExternalLogins = accounts.Count() > 1 || HasPassword(manager);
        return accounts;
    }

    public void RemoveLogin(string loginProvider, string providerKey)
    {
        UserManager manager = new UserManager();
        var result = manager.RemoveLogin(User.Identity.GetUserId(), new UserLoginInfo(loginProvider, providerKey));
        var msg = result.Succeeded
            ? "?m=RemoveLoginSuccess"
            : String.Empty;
        Response.Redirect("~/Account/Manage" + msg);
    }

    private void AddErrors(IdentityResult result)
    {
        foreach (var error in result.Errors)
        {
            ModelState.AddModelError("", error);
        }
    }
}
```

```
using Microsoft.Owin.Security;
using Microsoft.AspNet.Identity;
using System;
using System.Collections.Generic;
using System.Globalization;
using System.Linq;
using System.Web;
using Лабы;

public partial class OpenAuthProviders : System.Web.UI.UserControl
{
    protected void Page_Load(object sender, EventArgs e)
    {
        if (IsPostBack)
        {
            var provider = Request.Form["provider"];
            if (provider == null)
            {
                return;
            }
            // Запрос перенаправления к внешнему поставщику входа
            string redirectUrl = ResolveUrl(String.Format(CultureInfo.InvariantCulture,
                "~/Account/RegisterExternalLogin?{0}={1}&returnUrl={2}", IdentityHelper.ProviderNameKey, provider, ReturnUrl));
            var properties = new AuthenticationProperties() { RedirectUri = redirectUrl };
            // Add xsrf verification when linking accounts
            if (Context.User.Identity.IsAuthenticated)
            {
                properties.Dictionary[IdentityHelper.XsrfKey] = Context.User.Identity.GetUserId();
            }
            Context.GetOwinContext().Authentication.Challenge(properties, provider);
            Response.StatusCode = 401;
            Response.End();
        }
    }

    public string ReturnUrl { get; set; }

    public IEnumerable<string> GetProviderNames()
    {
```

```
40          return Context.GetOwinContext().Authentication.GetExternalAuthenticationTypes().Select(t => t.
        AuthenticationType);
41      }
42  }
```

```
1   using Microsoft.AspNet.Identity;
2   using Microsoft.Owin.Security;
3   using System;
4   using System.Web;
5   using Лабы;
6
7   public partial class Account_RegisterExternalLogin : System.Web.UI.Page
8   {
9       protected string ProviderName
10      {
11          get { return (string)ViewState["ProviderName"] ?? String.Empty; }
12          private set { ViewState["ProviderName"] = value; }
13      }
14
15      protected string ProviderAccountKey
16      {
17          get { return (string)ViewState["ProviderAccountKey"] ?? String.Empty; }
18          private set { ViewState["ProviderAccountKey"] = value; }
19      }
20
21      protected void Page_Load()
22      {
23          // Обработка результата от поставщика проверки подлинности в запросе
24          ProviderName = IdentityHelper.GetProviderNameFromRequest(Request);
25          if (String.IsNullOrEmpty(ProviderName))
26          {
27              Response.Redirect("~/Account/Login");
28          }
29          if (!IsPostBack)
30          {
31              var manager = new UserManager();
32              var loginInfo = Context.GetOwinContext().Authentication.GetExternalLoginInfo();
33              if (loginInfo == null)
34              {
35                  Response.Redirect("~/Account/Login");
36              }
37              var user = manager.Find(loginInfo.Login);
38              if (user != null)
39              {
40                  IdentityHelper.SignIn(manager, user, isPersistent: false);
41                  IdentityHelper.RedirectToReturnUrl(Request.QueryString["ReturnUrl"], Response);
42              }
43              else if (User.Identity.IsAuthenticated)
44              {
45                  // Apply Xsrf check when linking
46                  var verifiedloginInfo = Context.GetOwinContext().Authentication.GetExternalLoginInfo(
        IdentityHelper.XsrfKey, User.Identity.GetUserId());
47                  if (verifiedloginInfo == null)
48                  {
49                      Response.Redirect("~/Account/Login");
50                  }
51
52                  var result = manager.AddLogin(User.Identity.GetUserId(), verifiedloginInfo.Login);
53                  if (result.Succeeded)
54                  {
55                      IdentityHelper.RedirectToReturnUrl(Request.QueryString["ReturnUrl"], Response);
56                  }
57                  else
58                  {
59                      AddErrors(result);
60                      return;
61                  }
62              }
63              else
64              {
65                  userName.Text = loginInfo.DefaultUserName;
66              }
67          }
68      }
69
70      protected void LogIn_Click(object sender, EventArgs e)
```

```
71      {
72          CreateAndLoginUser();
73      }
74
75      private void CreateAndLoginUser()
76      {
77          if (!IsValid)
78          {
79              return;
80          }
81          var manager = new UserManager();
82          var user = new ApplicationUser() { UserName = userName.Text };
83          IdentityResult result = manager.Create(user);
84          if (result.Succeeded)
85          {
86              var loginInfo = Context.GetOwinContext().Authentication.GetExternalLoginInfo();
87              if (loginInfo == null)
88              {
89                  Response.Redirect("~/Account/Login");
90                  return;
91              }
92              result = manager.AddLogin(user.Id, loginInfo.Login);
93              if (result.Succeeded)
94              {
95                  IdentityHelper.SignIn(manager, user, isPersistent: false);
96                  IdentityHelper.RedirectToReturnUrl(Request.QueryString["ReturnUrl"], Response);
97                  return;
98              }
99              AddErrors(result);
100         }
101     }
102
103     private void AddErrors(IdentityResult result)
104     {
105         foreach (var error in result.Errors)
106         {
107             ModelState.AddModelError("", error);
108         }
109     }
110 }
```