

WEB SYSTEMS 202

Marinda Taljaard
Office 09 02 29
marinda.taljaard@mandela.ac.za

ASP.NET MVC Authentication

ADMIN

- Un-moderated test marks available
 - Scripts not available yet
- Portfolio
 - Due Tuesday 16 October, 12:00 midday
 - Submit on server (not Moodle)
 - MVC projects are big – do not start the copy process 11:55....
 - Submit details to follow later
- Module evaluation – 3 questions – please complete in Moodle

AUTHORIZATION AND AUTHENTICATION

- Complete access to anonymous users
 - Default situation
- Restricted access to some pages
 - Authentication required on some pages
- Restricted access to all pages
 - All users must be authenticated
- Restricted access at different levels
 - Only authenticated users can access “general” pages
 - Only admin users can access “admin” pages



SPECIFIC ACCESS TO ACTION METHODS

- In Controller file
 - Decorate the controller (or individual) methods with attributes specifying access:
 - [AllowAnonymous]
 - [Authorize]

```
public class SecurityController : Controller
{
    // GET: Security
    [AllowAnonymous]
    public ActionResult Login()
    {
        return View();
    }
    [AllowAnonymous]
    public ActionResult CheckUser()
    {
        //In this check we will do the forms authentication
    }
}
```

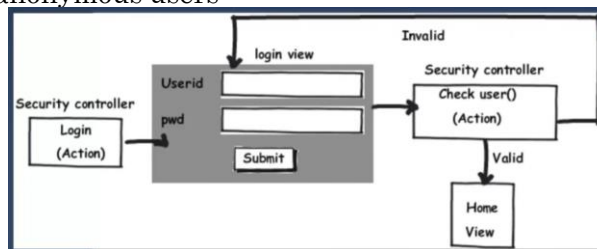
```
[Authorize]
public class MoviesController : Controller
{
    private MovieDbContext db = new MovieDbContext();

    // GET: Movies
    public ActionResult Index()
    {
        return View(db.Movies.ToList());
    }

    // GET: Movies/Details/5
    public ActionResult Details(int? id)
    {
        if (id == null)
        {
            return RedirectToAction("Index");
        }
    }
}
```

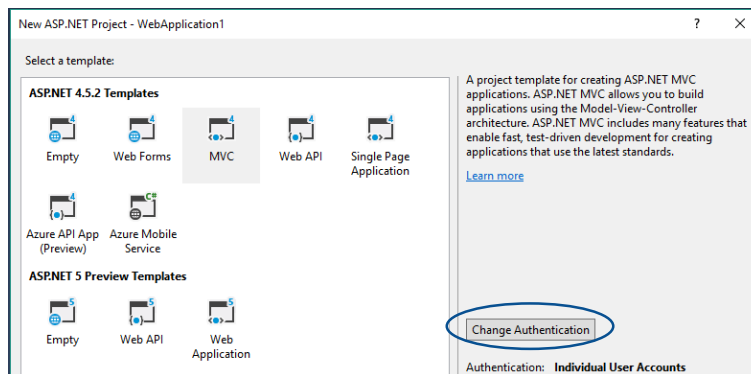
USER AUTHENTICATION

- List of registered users (database table?)
- Add Controller to handle authentication
- Login View to collect user credentials
- Register View to add more users to the table
- Consider levels of Authentication
 - E.g. Specify that only login page can be viewed by anonymous users



USER AUTHENTICATION

- Windows Authentication
- Individual User Accounts
- Work and School Accounts



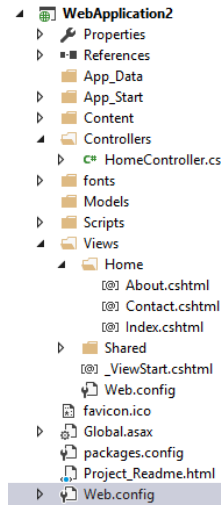
USER AUTHENTICATION

- Selecting Windows authentication
 - Info in web.config

```

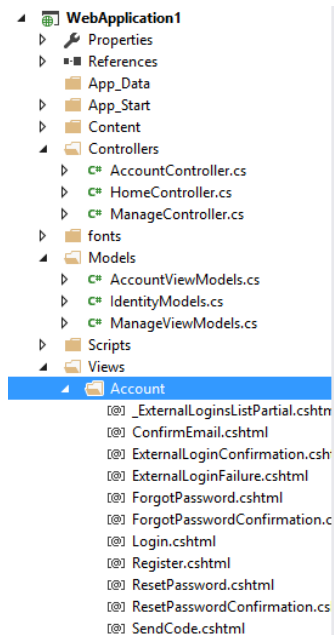
Web.config -x Your ASP.NET application
1 <?xml version="1.0" encoding="utf-8"?>
2 <!--
3 For more information on how to configure your ASP.NET applicat
4 http://go.microsoft.com/fwlink/?linkid=301879
5 -->
6 <configuration>
7   <appSettings>
8     <add key="webpages:Version" value="3.0.0.0"/>
9     <add key="webpages:Enabled" value="false"/>
10    <add key="ClientValidationEnabled" value="true"/>
11    <add key="UnobtrusiveJavaScriptEnabled" value="true"/>
12  </appSettings>
13  <system.web>
14    <compilation debug="true" targetFramework="4.5.2"/>
15    <httpRuntime targetFramework="4.5.2"/>
16    <authentication mode="Windows"/>
17    <authorization>
18      <deny users="?" />
19    </authorization>
20  </system.web>

```



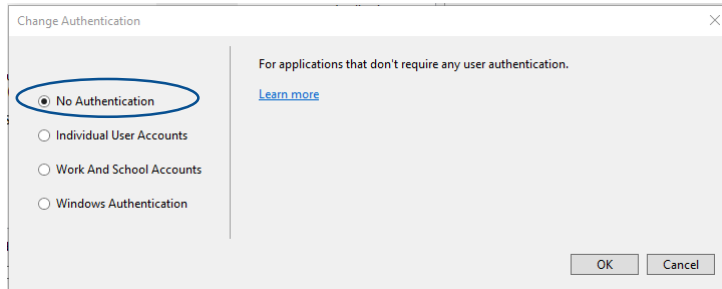
USER AUTHENTICATION

- Selecting Individual user accounts – creates:
 - AccountContoller
 - AccountviewModels
 - Many views linked to AccountController
 - Info in web.config
- Uses Membership and Identity
 - Which we not covering in this module



CUSTOM USER AUTHENTICATION

- Select No Authentication
 - Will use Forms Authentication – update web.config
 - Add required model, controllers and views



```
<authentication mode="Forms">
  <forms loginUrl="~/Security/Login" defaultUrl="~/Home/Index"></forms>
</authentication>
```

USER AUTHENTICATION

- Forms authentication
 - Hardcode details in controller?
 - Only for testing purposes
 - Use details from database table (model)?
- Configure the application to use Forms Authentication (web.config)
- Create a login page
- Whenever a user tries to access the restricted area, push him to Login page
- When a user tries to login, verify his credentials
 - Valid username and password

FORMS AUTHENTICATION

- Create Model for user information
 - Add table to database, or
 - Add class with properties
 - Build application again
- Validation can be specified in class

```

12 1 reference
13 public class User
14 {
15     0 references
16     public int Id { get; set; }
17
18     [Required(ErrorMessage = "Username required.", AllowEmptyStrings = false)]
19     1 reference
20     public string Username { get; set; }
21
22     [Required(ErrorMessage = "Password required.", AllowEmptyStrings = false)]
23     [DataType(DataType.Password)]
24     1 reference
25     public string Password { get; set; }
  
```

FORMS AUTHENTICATION

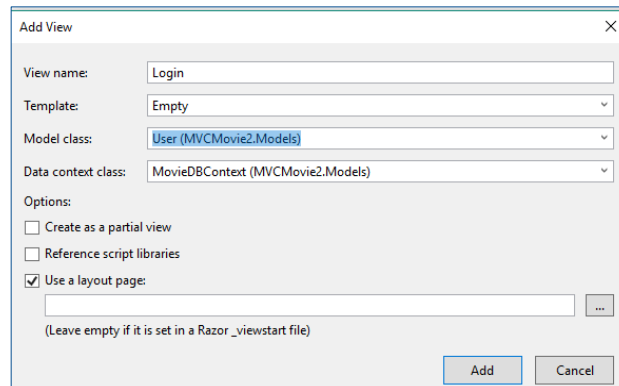
- Create Controller to handle login/authentication process

```

4  using System.Web;
5  using System.Data.Entity;
6  using System.Web.Mvc;
7  using System.Web.Security;
8  using MVCMovie2.Models;
9
10 namespace MVCMovie2.Controllers
11 {
12     0 references
13     public class SecurityController : Controller
14     {
15         // GET: Security
16         [AllowAnonymous]
17         0 references
18         public ActionResult Login()
19         {
20             return View();
21         }
22         [AllowAnonymous]
23         0 references
24         public ActionResult CheckUser()
25         {
26             //In this check we will do the forms authentication
  
```

FORMS AUTHENTICATION

- Login method linked to Login View
 - Use an empty view linked to user model, OR
 - Use the Create template – which will provide some textboxes linked to your model



Add View

View name: Login

Template: Empty

Model class: User (MVCMovie2.Models)

Data context class: MovieDbContext (MVCMovie2.Models)

Options:

☐ Create as a partial view

☐ Reference script libraries

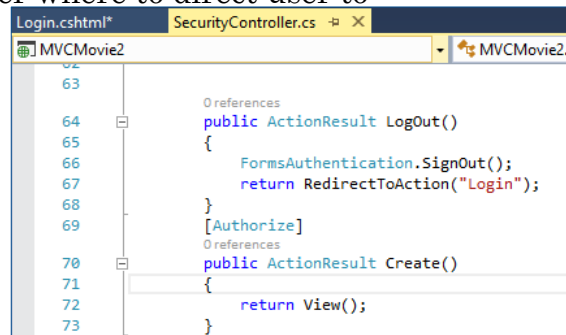
☒ Use a layout page:

(Leave empty if it is set in a Razor _viewstart file)

Add Cancel

FORMS AUTHENTICATION

- Create other applicable methods (and views), e.g.
 - LogOut
 - Create/Register - for adding more users
- Consider whether [Authorize] attribute should be used
- Consider where to direct user to



```

63
64
65
66
67
68
69
70
71
72
73

0 references
public ActionResult LogOut()
{
    FormsAuthentication.SignOut();
    return RedirectToAction("Login");
}

[Authorize]
0 references
public ActionResult Create()
{
    return View();
}

```

LOGIN VIEW

```

SecurityController.cs | Login.cshtml
4 ViewBag.Title = "Login";
5 Layout = "~/Views/Shared/_Layout.cshtml";
6
7
8 <h2>Login</h2>
9
10
11 @using (Html.BeginForm())
12 {
13     @Html.AntiForgeryToken()
14
15     <div class="form-horizontal">
16         <h4>MUser</h4>
17         <hr />
18         @Html.ValidationSummary(true, "", new { @class = "text-danger" })
19         <div class="form-group">
20             @Html.LabelFor(model => model.Username, htmlAttributes: new { @class = "control-label col-md-2" })
21             <div class="col-md-10">
22                 @Html.EditorFor(model => model.Username, new { htmlAttributes = new { @class = "form-control" } })
23                 @Html.ValidationMessageFor(model => model.Username, "", new { @class = "text-danger" })
24             </div>
25         </div>
26
27         <div class="form-group">
28             @Html.LabelFor(model => model.Password, htmlAttributes: new { @class = "control-label col-md-2" })
29             <div class="col-md-10">
30                 @Html.EditorFor(model => model.Password, new { htmlAttributes = new { @class = "form-control" } })
31                 @Html.ValidationMessageFor(model => model.Password, "", new { @class = "text-danger" })
32             </div>
33         </div>
34
35         <div class="form-group">
36             <div class="col-md-offset-2 col-md-10">
37                 <input type="submit" value="Login" class="btn btn-default" />
38             </div>
39         </div>
40     </div>
41 }

```

FORMS AUTHENTICATION – PROCESS (1)

- Hardcode proper username and password in controller
 - To be removed later...

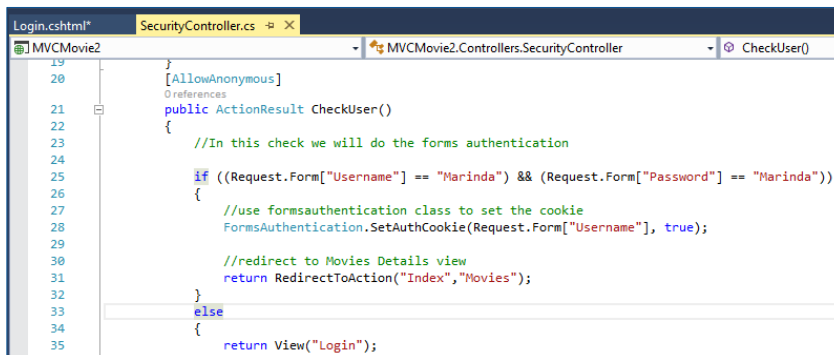
```

Login.cshtml | SecurityController.cs
MVCMovie2 | MVCMovie2.Controllers.SecurityController | CheckUser()
19 }
20 [AllowAnonymous]
21 public ActionResult CheckUser()
22 {
23     //In this check we will do the forms authentication
24
25     if ((Request.Form["Username"] == "Marinda") && (Request.Form["Password"] == "Marinda"))
26     {
27         //use formsauthentication class to set the cookie
28         FormsAuthentication.SetAuthCookie(Request.Form["Username"], true);
29
30         //redirect to Movies Details view
31         return RedirectToAction("Index", "Movies");
32     }
33     else
34     {
35         return View("Login");
36     }
37 }

```


FORMS AUTHENTICATION – PROCESS (2)

- Set the Authentication cookie for this user (if required)
- Redirect to appropriate page
- Can make use of session state variable to keep track of username over the different pages



```

19  }
20  [AllowAnonymous]
21  public ActionResult CheckUser()
22  {
23      //In this check we will do the forms authentication
24
25      if ((Request.Form["Username"] == "Marinda") && (Request.Form["Password"] == "Marinda"))
26      {
27          //use formsauthentication class to set the cookie
28          FormsAuthentication.SetAuthCookie(Request.Form["Username"], true);
29
30          //redirect to Movies Details view
31          return RedirectToAction("Index","Movies");
32      }
33      else
34      {
35          return View("Login");

```

FORMS AUTHENTICATION – PROCESS (3)

- Add/Update Model and Database
 - ADO.net code in Model or in Controller?
- Update Relevant Model / Controller
 - Query the database – Does this user exist?
 - E.g. (basic idea; there will be other ways too...)

```

bool flag = false;
string CS;
CS = ConfigurationManager.ConnectionStrings["MovieDBContext"].ConnectionString;
SqlConnection connection = new SqlConnection(CS);
string cmd = "Select count(*) from [Users] where [Username]='" + username + "' and [Password]='" + password + "'";
connection.Open();
SqlCommand command = new SqlCommand(cmd, connection);
flag = Convert.ToBoolean(command.ExecuteScalar());
connection.Close();
return flag;

```

- If the user exists – set cookie
- Redirect to relevant page



FORMS AUTHENTICATION - HASHING

- Create/Register:
 - Before the password is written to the database, it must be hashed
 - Use FormsAuthentication class
 - HashPasswordForStoringInConfigFile method
 - Remember to check whether the user already exists, before trying to insert into the table
- Login:
 - Hash the information from the form field before the query is run against the database



REMEMBER THE FINISHING TOUCHES

- Remove unused “template” components
- Update your menu bar when you have new options
 - Definitely in the _layout View
 - Possibly some aspects in other Views (e.g. headings)
- Update any “template – general” information



FOR EXAMPLE

Some of the views...

The image displays three screenshots of the MVC Movie application's user interface, each with a dark navigation bar at the top containing links for 'MVC Movie', 'Login', 'Movies', 'Create User', and 'Logout'.

Login View: Features the heading 'Login' and the instruction 'Enter your user credentials'. It includes input fields for 'Username' and 'Password', a 'Login' button, and a footer with '© 2017 - MVC Movies'.

Create View: Features the heading 'Create' and the sub-heading 'User'. It includes input fields for 'Username', 'Password', 'Firstname', 'Surname', and 'Email', a 'Create' button, and a footer with '© 2017 - MVC Movies'.

Index View: Features the heading 'Index' and a link 'Create New'. It contains a table with the following data:

Title	ReleaseDate	Genre	Price	
Star Wars	2011/01/01 12:00:00 AM	Science Fiction	20.00	Edit Details Delete
When Harry met Sally	1989/01/11 12:00:00 AM	Comedy	15.00	Edit Details Delete


The footer of the Index view reads '© 2017 - MVC Movies'.

TROUBLESHOOTING

Database first method

- Adding tables later can be problematic
- Might then need different DbContext - connection string to database

AUTHENTICATION WITH ROLES

- Restricted access at different levels
 - Only authenticated users can access “general” pages
 - Only admin users can access “admin” pages
 - Update model and database
 - Add roles (Admin, user/general)
 - Specify appropriate authorization
 - [Authorize (Roles = “Admin”)]
- 

RESOURCES

- Forms authentication
 - <https://www.aspsnippets.com/Articles/Forms-Authentication-using-FormsAuthentication-Ticket-Cookie-example-in-ASPNet-MVC.aspx>
 - <https://www.codeproject.com/articles/578374/aplusbeginner-splustutorialplusonplustocomplust>
 - Session State:
 - <http://www.beansoftware.com/ASP.NET-Tutorials/Write-Read-Delete-Session.aspx>
- 