Tipo : Guía de laboratorio
Capítulo : Seguridad de aplicaciones web
Duración : 45 minutos

I. OBJETIVO

Implementación de autenticación y creación la página de Login.

II. REQUISITOS

Los siguientes elementos de software son necesarios para la realización del laboratorio:

- Windows 10 (como mínimo Windows 8)
- Visual Studio 2017 (como mínimo Visual Studio 2015)

III. EJECUCIÓN DEL LABORATORIO

1. Abrir la solución del módulo anterior.

2. Instalar los siguientes paquetes nuget en el proyecto **Cibertec.Mvc:**
   - Install-Package Microsoft.Owin.Host.SystemWeb
   - Install-Package Microsoft.Owin.Security.Cookies

3. Creamos una nueva clase en la raíz del proyecto "**Cibertec.Mvc**" con el nombre "**Startup**" con el siguiente código:

```
using Microsoft.Owin;
using Owin;
using Microsoft.Owin.Security.Cookies;

[assembly: OwinStartup(typeof(Cibertec.Mvc.Startup))]

namespace Cibertec.Mvc
{
    public class Startup
    {
        public void Configuration(IAppBuilder app)
        {
            app.UseCookieAuthentication(new CookieAuthenticationOptions
            {
                AuthenticationType = "ApplicationCookie",
                LoginPath = new PathString("/Account/Login")
            });
        }
    }
}
```

4. En el fichero "**web.config**" agregamos la siguiente line de código en la llave "**AppSettings**":

```
<add key="owin:AppStartup" value="Cibertec.Mvc.Startup" />
```

Quedando como se muestra en la siguiente imagen:

```
<appSettings>
  <add key="webpages:Version" value="3.0.0.0" />
  <add key="webpages:Enabled" value="false" />
  <add key="ClientValidationEnabled" value="true" />
  <add key="UnobtrusiveJavaScriptEnabled" value="true" />
  <add key="owin:AppStartup" value="Cibertec.Mvc.Startup" />
</appSettings>
```

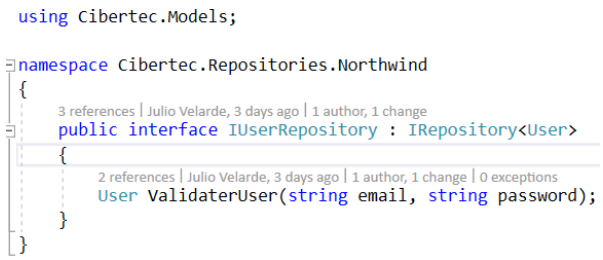
5. Creamos un nuevo ViewModel en la carpeta "**Models**" con el nombre "**UserViewModel**" que tendrá el siguiente código:

```
using System.ComponentModel.DataAnnotations;

namespace Cibertec.Mvc.Models
{
    public class UserViewModel
    {
        [Required]
        [DataType(DataType.EmailAddress)]
        public string Email { get; set; }

        [Required]
        [DataType(DataType.Password)]
        public string Password { get; set; }
        public string ReturnUrl { get; set; }
    }
}
```

6. Para poder validar que el usuario exista en nuestra base de datos debemos de modificar nuestro repositorio:
   (1) En la interfaz "**IUserRepository**" agregamos la siguiente línea:

```
User ValidaterUser(string email, string password);
```

Validar con esta imagen:



   (2) Procedemos con la implementación en el fichero "**UserRepository**" con el siguiente código:

```
public User ValidaterUser(string email, string password)
        {
            using (var connection = new
SqlConnection(_connectionString))
            {
                var parameters = new DynamicParameters();
                parameters.Add("@email", email);
                parameters.Add("@password", password);

                return
connection.QueryFirstOrDefault<User>("dbo.ValidateUser",
                    parameters,
                    commandType:
System.Data.CommandType.StoredProcedure);
```

```
            }
        }
```

(3) Creamos el procedimiento almacenado para validar los datos.

```
CREATE PROCEDURE [dbo].[ValidateUser]
    @email varchar(100),
    @password varchar(20)
AS
BEGIN
    SELECT [Id]
      ,[Email]
      ,[FirstName]
      ,[LastName]
      ,[Roles]
    FROM [dbo].[User]
    WHERE [Email]=@email AND [Password]=@password
END
```

7. Procedemos a crear un nuevo controlador con el nombre "**AccountController**" y agregamos el siguiente código:

```
using Cibertec.Mvc.Models;
using System.Web;
using System.Web.Mvc;
using Cibertec.UnitOfWork;
using log4net;
using System.Security.Claims;

namespace Cibertec.Mvc.Controllers
{
    public class AccountController : BaseController
    {
        public AccountController(ILog log, IUnitOfWork unit) : base(log,
unit)
        {
        }

        [AllowAnonymous]
        public ActionResult Login(string returnUrl)
        {
            return View(new UserViewModel { ReturnUrl = returnUrl });
        }

        [HttpPost]
        [ValidateAntiForgeryToken]
        [AllowAnonymous]
        public ActionResult Login(UserViewModel user)
        {
            if (!ModelState.IsValid) return View(user);
            var validUser = _unit.Users.ValidaterUser(user.Email,
user.Password);
            if (validUser == null)
            {
                ModelState.AddModelError("Error", "Invalid email or
password");
                return View(user);
            }

            var identity = new ClaimsIdentity(new[] {
                new Claim(ClaimTypes.Email, validUser.Email),
                new Claim(ClaimTypes.Role, validUser.Roles),
```

```
                new Claim(ClaimTypes.Name, $"{validUser.FirstName}
{validUser.LastName}"),
                new Claim(ClaimTypes.NameIdentifier,
validUser.Id.ToString())
            }, "ApplicationCookie");

            var context = Request.GetOwinContext();
            var authManager = context.Authentication;

            authManager.SignIn(identity);
            return RedirectToLocal(user.ReturnUrl);
        }

        public ActionResult Logout()
        {
            var context = Request.GetOwinContext();
            var authManager = context.Authentication;

            authManager.SignOut("ApplicationCookie");
            return RedirectToAction("Login", "Account");
        }

        private ActionResult RedirectToLocal(string returnUrl)
        {
            if (Url.IsLocalUrl(returnUrl))
            {
                return Redirect(returnUrl);
            }
            return RedirectToAction("Index", "Home");
        }
    }
}
```

8. Editamos nuestro "**Global.asax**" con el siguiente código:

```
using System.Security.Claims;
using System.Web.Helpers;
using System.Web.Mvc;
using System.Web.Optimization;
using System.Web.Routing;

namespace Cibertec.Mvc
{
    public class MvcApplication : System.Web.HttpApplication
    {
        protected void Application_Start()
        {
            DIConfig.ConfigureInjector();
            AreaRegistration.RegisterAllAreas();
            FilterConfig.RegisterGlobalFilters(GlobalFilters.Filters);
            RouteConfig.RegisterRoutes(RouteTable.Routes);
            BundleConfig.RegisterBundles(BundleTable.Bundles);

            AntiForgeryConfig.UniqueClaimTypeIdentifier =
ClaimTypes.NameIdentifier;
            log4net.Config.XmlConfigurator.Configure();
        }
    }
}
```

9. Procedemos a modificar el "**BaseController**" con el siguiente código:

```
using Cibertec.UnitOfWork;
using log4net;
using System.Web.Mvc;

namespace Cibertec.Mvc.Controllers
{
    [Authorize]
    public class BaseController : Controller
    {
        protected readonly IUnitOfWork _unit;
        protected readonly ILog _log;
        public BaseController(ILog log, IUnitOfWork unit)
        {
            _log = log;
            _unit = unit;
        }
    }
}
```

10. En la carpeta "**Views\Account**" creamos la vista "**Login.cshtml**" con el siguiente código:

```
@model Cibertec.Mvc.Models.UserViewModel

@{
    ViewBag.Title = "Login";
}

<h2>Login</h2>

@using (Html.BeginForm())
{
    @Html.AntiForgeryToken()

    <div class="form-horizontal">
        <h4>UserViewModel</h4>
        <hr />
        @Html.ValidationSummary(true, "", new { @class = "text-danger" })
        <div class="form-group">
            @Html.LabelFor(model => model.Email, htmlAttributes: new {
@class = "control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.Email, new { htmlAttributes
= new { @class = "form-control" } })
                @Html.ValidationMessageFor(model => model.Email, "", new {
@class = "text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.Password, htmlAttributes: new {
@class = "control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.Password, new {
htmlAttributes = new { @class = "form-control" } })
                @Html.ValidationMessageFor(model => model.Password, "",
new { @class = "text-danger" })
            </div>
        </div>

        <div class="form-group">
            <div class="col-md-offset-2 col-md-10">
```

```
                <input type="submit" value="Login" class="btn btn-success"
/>
            </div>
        </div>
    </div>
}

@section Scripts {
    @Scripts.Render("~/bundles/jqueryval")
}
```

11. Para poder visualizar si estamos autenticados o no, creamos la vista parcial "**_LoginPartial.cshtml**" en la carpeta "**Views\Shared**" con el siguiente código:

```
@if (Request.IsAuthenticated)
{
    using (Html.BeginForm("Logout", "Account", FormMethod.Post, new { id =
"logoutForm", @class = "navbar-right" }))
    {
        @Html.AntiForgeryToken()

        <ul class="nav navbar-nav navbar-right">
            <li>
                @Html.ActionLink($"Hello {User.Identity.Name}", "Index",
"Home")
            </li>
            <li><a
href="javascript:document.getElementById('logoutForm').submit()">Log
off</a></li>
        </ul>
    }
}
else
{
    <ul class="nav navbar-nav navbar-right">
        <li>@Html.ActionLink("Log in", "Login", "Account", routeValues:
null, htmlAttributes: new { id = "loginLink" })</li>
    </ul>
}
```

12. Como paso final realizamos la edición del "**_Layout.cshtml**" como se muestra en la siguiente imagen:

```
<div class="navbar navbar-inverse navbar-fixed-top">
    <div class="container">
        <div class="navbar-header">
            <button type="button" class="navbar-toggle" data-toggle="collapse" data-target=".navbar-collapse">
                <span class="icon-bar"></span>
                <span class="icon-bar"></span>
                <span class="icon-bar"></span>
            </button>
            @Html.ActionLink("Application name", "Index", "Home", new { area = "" }, new { @class = "navbar-brand" })
        </div>
        <div class="navbar-collapse collapse">
            <ul class="nav navbar-nav">
                <li>@Html.ActionLink("Home", "Index", "Home")</li>
                <li>@Html.ActionLink("Customer", "Index", "Customer")</li>
            </ul>
            @Html.Partial("_LoginPartial")
        </div>
    </div>
</div>
```

13. Ejecutamos la aplicación y validar que la autenticación sea funcional.

IV. EVALUACIÓN

1. ¿Cuáles son las carpetas principales en una aplicación MVC?

   Las carpetas principales son: Controllers, Views y Models

2. ¿Qué pasa si al nombre del controlador no se le coloca el prefijo Controller?

   Por convención todos los archivos que van a ser controladores deben tener el prefijo Controller al final del nombre para que puedan ser reconocidos por Asp.NET como tales.

3. ¿Cuál es el tipo de datos de los métodos del controller?

   El tipo base de todos los controladores es ActionResult.

4. ¿Qué es una vista Layout?

   Es la vista maestra, en ella se establece la estructura base (html) de todas las vistas que tendrá la aplicación web. Algunas vistas pueden no utilizar la vista base.

5. Los modelos, ¿pueden ser librería de clases?

   Sí, los modelos son un concepto que involucra entidades, acceso a datos, reglas de negocio y pueden ser implementadas en librería de clases (dll).

6. ¿Qué son las vistas parciales?

   Al igual que las vistas normales, estas permiten ser reutilizadas en ciertas partes de la aplicación.