ACTIVIDAD 04 – ASP.NET

PATRÓN MVC

DESARROLLO RÁPIDO DE APLICACIONES

Por Juan José Camacho Hidalgo
UNIVERSIDAD DE ALMERÍA

ASP.NET se trata de un entorno para aplicaciones web propio de Microsoft, que construye sitios dinámicos, aplicaciones web y servicios web XML.

En la solución proporcionada, vamos a crear tres versiones de una aplicación web. En la primera, MyFirstMvcWebApp, vamos a crear un Hello World. Se crea mediante una plantilla proporcionada por Microsoft y nos da la opción de incorporar el patrón MVC.

Se genera un proyecto con carpetas para la vista, el modelo y el controlador. El proyecto además genera una carpeta de App_Data, para colocar los archivos que requieren de lectura/escritura, App_Start, que incluye los archivos de configuración de las tecnologías usadas en las app, Content, para todos los archivos CSS, es decir, el estilo del diseño de la aplicación web, Filters, para atributos personalizados que añaden comportamientos específicos, Scripts, para cualquier archivo JavaScript, References con todos los ensamblados que usa la app, los archivos de configuración en XML del funcionamiento de la app, Global.asax, para gestionar eventos, y packages.config para almacenar la información de los paquetes NuGet.

```
<!DOCTYPE html>

<html>
<head>
    <meta name="viewport" content="width=device-width" />
    <title>Index</title>
</head>
<body>
    <div>
        <h1>Hello World ASP.NET MVC!!!</h1>
    </div>
</body>
</html>
```

En el index definimos todo lo que la aplicación Hello World va a mostrar por pantalla.

```
0 referencias
public class MyFirstController : Controller
{
    //
    // GET: /MyFirst/

    0 referencias
    public ActionResult Index()
    {
        return View();
    }

}
```

El controlador llama a la vista y de momento, no se incorpora ningún modelo.

En MySecondMvcApp se definen una página principal, una gestión de login y una lista de personas. Todo ello manejado como controladores (AccountController, HomeController, PersonaController), que hace uso del modelo de Persona y de AccountModels, es decir, un esquema para la cuenta y otro para las personas. Además inicializa mediante un filtro llamado InitializeSimpleMembershipAttribute el usuario desde la base de datos. Base de datos que usa en una DbContext, basado en el modelo. La aplicación como extra añade una hoja de estilos llamado Site.css.

Por último, MyThirdMvcApp, hace uno de un modelo obtenido por Entity Framework de la base de datos, y añade más funcionalidad a esta aplicación web. Se mantiene la hoja de estilos. Ambas aplicaciones están adapatadas a dispositivos móviles. Como decíamos hace uso en modelos del modelo obtenido de la base de datos, además de AccountModels y Producto, ya que añade productos. Los controladores son AccountController, CategoriaController, HomeController y MantenimientoCategoriasController, haciendo todos ellos uso de las diferentes vistas que muestra. Muchas páginas distribuidas en carpetas por categorías: Home, Account, Categorias, MantenimientoCategoria, donde muestra cada una de las operaciones o accesos disponibles, mediante archivos .cshtml.

```
@{
    ViewBag.Title = "Create";
}

<h2>Create</h2>

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

    <fieldset>
        <legend>Categorias</legend>

        <div class="editor-label">
            @Html.LabelFor(model => model.Nombre)
        </div>
        <div class="editor-field">
            @Html.EditorFor(model => model.Nombre)
            @Html.ValidationMessageFor(model => model.Nombre)
        </div>

        <p>
            <input type="submit" value="Create" />
        </p>
    </fieldset>
}

<div>
    @Html.ActionLink("Back to List", "Index")
</div>

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

Como ejemplo, mostramos el Create.cshtml, para crear productos.