Creación de rutas

Anteriormente hemos visto que el sistema de routing actúa como *front-controller*, un controlador frontal al que llegan las peticiones para, en función de la información contenida en la tabla de rutas, delegar el proceso de las mismas a controladores más específicos de nuestra aplicación.

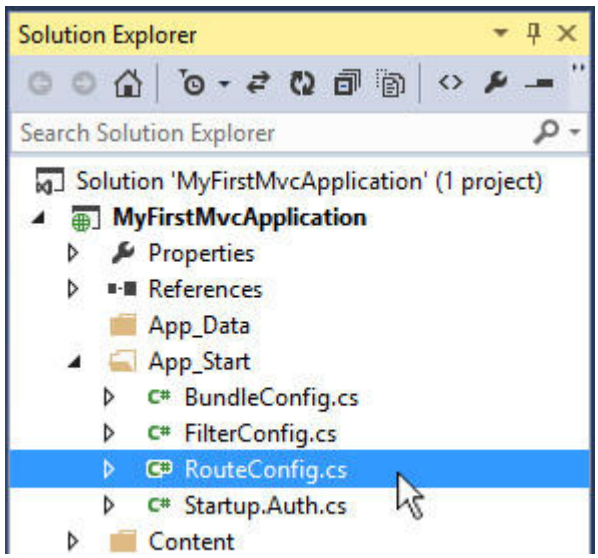
Recordemos que las rutas que habíamos planificado para acceder a las funcionalidades del sistema eran las siguientes:

Patrón de URL	Ejemplos de petición	Acción a realizar
{controller}/{action}/{id}	/blog	Muestra el listado con un resumen de las últimas entradas publicadas, ordenadas cronológicamente.
blog/archive/{year}/{month}	/blog/archive/2005/12 /blog/archive/2006/8	Muestra el resumen de las entradas publicadas en el mes y año indicado en los parámetros de ruta.
blog/{code}	/blog/welcome-to-aspnet-mvc	Muestra el artículo cuyo código coincida con el especificado en la ruta.

Veamos cada una de ellas con más detenimiento.

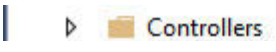
{controller}/{action}/{id}

El patrón de URLs que encontramos en primer lugar coincide con la ruta por defecto, aquella que se incluye en la plantilla de proyectos ASP.NET MVC, por lo que no tendremos que registrarla de nuevo.



Anteriormente hemos comentado que la tabla de rutas se llena durante la inicialización de la aplicación, por lo que, según la convención de ubicación de archivos, encontraremos el código que realiza el registro de rutas en la

carpeta `/App_Start`, y más concretamente en la clase `RouteConfig`. Su método estático `RegisterRoutes()` es invocado desde el evento `Application_Start` (en el archivo *`global.asax.cs`*), y es el encargado de registrar todas las rutas que utilizará nuestra aplicación.



En concreto, la ruta por defecto se define en la siguiente porción de código:

```
routes.MapRoute(
    name: "Default",
    url: "{controller}/{action}/{id}",
    defaults: new { controller = "Home", action = "Index", id =
UrlParameter.Optional }
);
```

Aunque lo veremos detalladamente más adelante, seguro que puedes intuir el significado de algunos de los parámetros utilizados para registrar la ruta:

- Mediante el parámetro `name` indicamos el nombre único que daremos a esta entrada de la tabla de rutas.
- El parámetro `url` indica el patrón de URL al que hace referencia esta ruta.
- Por último, `defaults` especifica el valor por defecto para los parámetros especificados en el patrón de URL anterior **en forma de objeto anónimo**. Aunque pueda parecer algo extraño, en realidad es sólo una forma muy compacta de escribir un diccionario clave-valor, que ya veremos que se utiliza bastante en ASP.NET MVC.

> Observa que en el parámetro {id}, en lugar de asignarle un valor por defecto se está utilizando la constante `UrlParameter.Optional` para indicar su no obligatoriedad.

Así, las peticiones a la URL `/Blog`, dado que encajan en el patrón indicado, serán procesadas por el controlador "Blog", que es el valor del parámetro `{controller}` de la URL. La acción a ejecutar, dado que no se especifica en la URL de la petición, se tomará el valor "Index", indicado por defecto para el parámetro `{action}`.

blog/archive/{year}/{month}

Como se puede observar, este tipo de direcciones no encajarían en el patrón de ruta anterior, por lo que será necesario crear una ruta específica para ello, insertando la siguiente porción en el código de inicialización:

```
routes.MapRoute(
    name: "archive",
    url: "blog/archive/{year}/{month}",
    defaults: new { controller = "Blog", action = "Archive" }
);
```

Fíjate que el patrón de URL no incluye los parámetros obligatorios `{controller}` y `{action}` porque no queremos introducir ningún factor variable en estas peticiones. Por esta razón, en el objeto anónimo definido en el tercer parámetro se establecen sus valores por defecto a "blog" y "archive", respectivamente, que corresponden con el nombre del controlador y la acción que procesará la petición. Observa también que en este mismo objeto no hemos asignado valores por defecto a los parámetros `{year}` y `{month}`, pues nos interesa que sean obligatorios.

> En este caso es indiferente la posición de esta regla dentro de la tabla de rutas, sin embargo, **hay que prestar mucha atención al hecho de que se procesan de forma secuencial** y el primer patrón que encaje con la URL del recurso solicitado será el que defina cómo va a ser procesada la petición.

Según la convención, las peticiones que se correspondan con este patrón serán procesadas por el método de acción `Archive()` que se encontrará en la clase `BlogController`.

blog/{codigo}

En este momento, si nuestro sistema recibiera una petición hacia la URL `/blog/hello-world`, ésta intentaría ser procesada por la ruta por defecto al encajar en el patrón {controller}/{action}/{id}. Sin embargo, generaría un error al interpretar que el proceso será delegado al controlador "blog", y a la acción "hello-world", lo cual es obviamente incorrecto.

Por tanto, en primer lugar, llegamos a la conclusión de que necesitamos crear una regla en la tabla de rutas para procesar las peticiones de este tipo. Además, en este caso **será necesario registrar la ruta en primer lugar**, para que se compruebe antes que las demás:

```
routes.MapRoute(
    name: "post",
    url: "blog/{code}",
```

```
    defaults: new { controller = "Blog", action = "ViewPost" }
);
```

Según la convención, las peticiones cuya URL se corresponda con este patrón, serán procesadas por el método de acción `ViewPost()` que se encontrará en la clase `BlogController`.

Resumen del registro de rutas

El código del archivo `App_Start/RouteConfig.cs`, incluyendo las definiciones que ya venían definidas por defecto, es el siguiente:

```
public class RouteConfig
{
   public static void RegisterRoutes(RouteCollection routes)
   {
      routes.IgnoreRoute("{resource}.axd/{*pathInfo}");

      routes.MapRoute(
         name: "post",
         url: "blog/{code}",
         defaults: new { controller = "Blog", action =
"ViewPost" }
      );

      routes.MapRoute(
         name: "archive",
         url: "blog/archive/{year}/{month}",
         defaults: new { controller = "Blog", action = "Archive"
}
      );

      routes.MapRoute(
         name: "Default",
         url: "{controller}/{action}/{id}",
         defaults: new { controller = "Home", action = "Index",
id = UrlParameter.Optional }
      );
   }
}
```

A continuación, siguiendo la convención, implementaremos el controlador "Blog" en la clase `BlogController`, con los métodos de acción que necesitamos para llevar a cabo las tareas propuestas: `Index()`, `Archive()` y `ViewPost()`.

> Habrás observado que en el código por defecto de la clase `RouteConfig`, aparece una llamada al método `IgnoreRoute()`. Como se puede intuir, se trata de una ruta especial para ignorar peticiones dirigidas a URLs concretas. Lo estudiaremos más adelante.
