**An Introduction to URL Routing**

이 튜토리얼에서, URL Routing이라 불리는 ASP.NET MVC 어플리케이션이 중요한 특징에 관해 설명할 것입니다. URL Routing 모듈은 특정 MVC controller actions에 대한 브라우저 요청을 매핑 합니다.

이 튜토리얼 처음에, 표준 route table이 어떻게 controller actions에 요청을 매핑 하는지 배울 것 입니다. 두 번째로, 기본 route table을 어떻게 수정하고 적용하는지 배울 것 입니다.

**Using the Default Route Table**

새로운 ASP.NET MVC 어플리케이션을 만들면, 어플리케이션은 이미 URL Routing을 사용할 준비를 해놓습니다. URL Routing은 두 단계 준비작업이 있습니다.

첫 번째로, URL Routing은 어플리케이션의 웹 구성 파일(Web.config file)에 활성화 됩니다. 구성 파일에는 routing에 상응하는 4가지 영역이 있습니다. system.web.httpModules 영역, system.web.httpHandlers 영역, system.webserver.modules 영역, 그리고 system.webserver.handlers 영역이 해당되는데 이 영역이 없으면 동작을 안 하기에 삭제에 신중해야 합니다.

두 번째로, 중요한 부분으로서, route table은 어플리케이션의 Global.asax 파일에 생성됩니다. 이 파일은 ASP.NET 어플리케이션 생명주기 이벤트와 관련된 이벤트 핸들러로 구성된 특별한 파일입니다. Route table은 어플리케이션 시작 이벤트가 발생할 때 만들어 집니다.

다음 코드는 ASP.NET MVC 어플리케이션에 포함된 기본 Global.asax 입니다.

|  |
| --- |
| using System;using System.Collections.Generic;using System.Linq;using System.Web;using System.Web.Mvc;using System.Web.Routing;namespace MyApp{ public class GlobalApplication : System.Web.HttpApplication { public static void RegisterRoutes(RouteCollection routes) { routes.IgnoreRoute("{resource}.axd/{\*pathInfo}"); routes.MapRoute( "Default", // Route name "{controller}/{action}/{id}", // URL with parameters new { controller = "Home", action = "Index", id = ""}  // Parameter defaults ); } protected void Application\_Start() { RegisterRoutes(RouteTable.Routes); } }} |

ASP.NET MVC 어플리케이션이 처음 시작할 때, Application\_Start() 메소드가 실행됩니다. 이 메소드는, RegisterRoutes() 메소드를 호출하고, RegisterRoutes() 메소드가 route table을 생성합니다.

기본 route table은 Default라는 이름의 single route 로 구성됩니다. Default route은 URL의 첫 번째 구분을 controller name과 매핑 시키고, 두 번째 구분을 URL의 controller action과 매핑 시킵니다. 그리고 세 번째 구분을 id 라는 파라메터로 매핑 시킵니다.

웹 브라우저 주소 바에 다음 URL의 입력을 생각해보세요.

/Home/Index/3

Default route은 다음 파라메터들로 매핑 시킵니다.

Controller = Home

Action = Index

Id = 3

URL /Home/Index/3을 요청할 때, 다음 코드가 실행됩니다.

HomeController.Index(3)

Default route은 3가지 파라메터는 기본값을 가지고 있습니다. Controller가 빠졌다면 controller 파라메터의 기본값인 Home이 적용됩니다. Action이 빠졌다면, action 파라메터의 기본값인 Index가 적용됩니다. 마지막으로 id 파라메터가 빠졌다면, 기본값인 공백 string이 적용됩니다.

아래의 URL을 브라우저 주소 창에 입력하면 Default route가 controller actions에 어떻게 매핑 시킬지 생각해보세요.

/Home

Default route 파라메터가 기본값으로 설정하여, 아래 코드의 HomeController 클래스의 Index() 메소드가 실행될 것입니다.

|  |
| --- |
| using System;using System.Collections.Generic;using System.Linq;using System.Web;using System.Web.Mvc;namespace MyApp.Controllers{ [HandleError] public class HomeController : Controller { public ActionResult Index(string Id) { return View(); } }} |

위 코드에서, HomeController 클래스는 Id라는 단일 파라메터를 받는 Index() 메소드를 포함하고 있습니다. URL /Home은 id 파라메터를 공백으로 Index() 메소드를 호출할 것입니다.

ASP.NET MVC 프레임워크는 controller actions을 호출 하기 때문에, URL /Home은 아래 코드의 HomeController 클래스에 있는 Index() 메소드와 매치 시킵니다.

|  |
| --- |
| using System;using System.Collections.Generic;using System.Linq;using System.Web;using System.Web.Mvc;namespace MyApp.Controllers{ [HandleError] public class HomeController : Controller { public ActionResult Index() { return View(); } }} |

위의 Index() 메소드는 어떠한 파라메터도 받지 않습니다. URL /Home이 호출될 때, Index() 메소드를 부릅니다. URL /Home/Index/3 이 입력된다면 아이디를 무시하고 이 메소드가 불려질 것입니다.

URL /Home은 항상 다음 코드의 HomeController 클래스 내의 Index() 메소드로 처리될 것입니다.

|  |
| --- |
| using System;using System.Collections.Generic;using System.Linq;using System.Web;using System.Web.Mvc;namespace MyApp.Controllers{ [HandleError] public class HomeController : Controller { public ActionResult Index(int? id) { return View(); } }} |

위 코드에서, Index() 메소드는 하나의 정수형 파라메터를 가지고 있습니다. 파라메터가 nullable 파라메터이기 때문에, Index() 메소드는 에러 발생 없이 호출될 것입니다.

마지막으로, 다음 코드에서 URL /Home으로 인하여 Index() 메소드가 불려지면, Id Parameter가 nullable 파라메터가 아니라는 에러가 발생됩니다.

|  |
| --- |
| using System;using System.Collections.Generic;using System.Linq;using System.Web;using System.Web.Mvc;namespace MyApp.Controllers{ [HandleError] public class HomeController : Controller { public ActionResult Index(int id) { return View(); } }} |



한편으로, URL /Home/Index/3은 위 코드에서 잘 작동합니다. 3은 파라메터 값으로 전달되어 Index() 메소드가 호출 될 것입니다.

**Creating a Custom Route**

ASP.NET MVC 어플리케이션을 위해, 기본 route table은 잘 작동 할 것입니다. 하지만, 특별한 routing에 대한 요구에 대하여 custom route을 만들 수도 있습니다.

예를 들어 블로그 어플리케이션을 만든다고 가정합시다. 아래와 같은 요청을 처리 해봅시다.

/Archive/12-25-2009

사용자가 이와 같은 요청을 했을 때, 12/25/2009 에 응답하는 블로그를 원합니다. 이런 유형의 요청을 다루기 위해서, custom route을 만들 필요가 있습니다.

아래의 Global.asax 파일은 /Archive/entry date 와 같은 요청을 다루는 Blog라는 이름의 새로운 custom route을 가지고 있습니다.

|  |
| --- |
| using System.Web.Mvc;using System.Web.Routing;namespace MyApp{ public class GlobalApplication : System.Web.HttpApplication { public static void RegisterRoutes(RouteCollection routes) { routes.IgnoreRoute("{resource}.axd/{\*pathInfo}"); routes.MapRoute( "blog", "Archive/{entryDate}", new {controller = "Archive", action = "Entry"} ); routes.MapRoute( "Default", // Route name "{controller}/{action}/{id}", // URL with parameters new { controller = "Home", action = "Index", id = ""}  // Parameter defaults ); } protected void Application\_Start() { RegisterRoutes(RouteTable.Routes); } }} |

Route table에 추가한 route의 순서는 매우 중요합니다. Blog라는 새로운 custom route은 Default route 전에 추가 해야 합니다. 순서가 바뀐다면, Default route은 항상 custom route 대신 호출될 것입니다.

Custom Blog route은 다음과 같은 /Archive 로 시작하는 요청을 처리 할 것입니다.

/Archive/12-25/2009

/Archive/10-6-2004

/Archive/apple

Custom route은 Archive controller와 Entry() action에 매핑 될 것입니다. Entry() 메소드가 호출되면, entryDate 파라메터로 전달될 것입니다.

Blog custom route을 다음 코드의 controller 로 사용할 수 있습니다.

|  |
| --- |
| using System;using System.Web.Mvc;namespace MyApp.Controllers{ public class ArchiveController : Controller { public string Entry(DateTime entryDate) { return "You requested the entry on " + entryDate.ToString(); } }} |

위 코드에서 Entry() 메소드는 DateTime 타입의 파라메터를 받는다는 것을 확인하세요. ASP.NET MVC 프레임워크는 자동적으로 URL의 날짜를 DateTime 타입으로 변환해줍니다. URL에 날짜 입력 값이 DateTime으로 변환될 수 없다면, 에러가 발생됩니다.