http://www.cnblogs.com/artech/archive/2012/03/19/aspnet-routing-01.html

表现为请求地址与目标Controller和Action的动态映射的URL路由系统并不是专属于ASP.NET MVC，而是直接建立在ASP.NET 中。ASP.NET通过URL路由系统实现了请求地址与物理文件的分离。[源代码地址从[这里](http://files.cnblogs.com/artech/S201.rar)（S201）下载]

**一、URL与物理文件的分离**

对于一个 ASP.NET Web Form应用来说，任何一个请求都对应着某个具体的物理文件。部署在Web服务器上的物理文件可以是静态的（比如图片和静态HTML文件等），也可以是动态的（比如.asxp文件）。对于静态文件的请求，ASP.NET直接返回文件的整个内容；而针对动态文件的请求则会触发相关代码的执行，并最终返回执行后的结果。但是这种将URL与物理文件紧密绑定在一起的方式并不是一种好的解决方案，它带来的局限性主要体现在如下几个方面：

* **灵活性：**由于URL是对物理文件路径的反映，意味着如果物理文件的路径发生了改变（比如改变了文件的目录结构或者文件名），原来基于该文件链接将变得无效。
* **可读性：**在很多情况下，URL不仅仅需要能够访问正确的网络资源，还需要具有很好的可读性，最好的URL应该让我们一眼就能看出针对它访问的目标资源是什么。请求地址与物理文件紧密绑定让我们完全失去了定义高可读性URL的机会。
* **SEO优化：**对于网站开发来说，为了迎合搜索引擎检索的规则，我们需要对URL进行有效的设计使之能易于被主流的引擎检索收录。如果URL完全与物理地址关联，这无异于失去了SEO优化的能力。

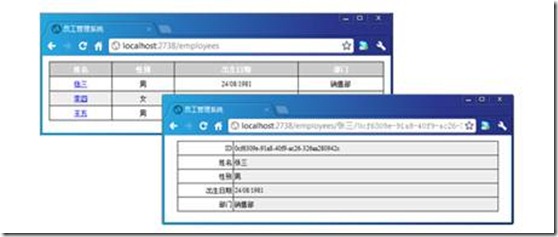
出于针对URL与物理文件绑定机制带来的上述局限，我们需要一种更加灵活的机制实现针对物理文件的请求地址与文件本身的路径的分离，通过一种动态映射的机制实现URL与物理文件的关联。

说到这里，可能很多人会想到URL重写。为了使Web应用可以独立地涉及用于访问应用资源的URL，微软为IIS 7编写了一个URL重写模块。这是一个基于规则的URL重写引擎，用于在URL被Web服务器处理之前改变请求的URL。对于动态Web应用程序，它可以为用户和搜索引擎提供友好的URL，URL重写和重定向是基于HTTP头和服务器变量的，并可以对站点内容进行访问控制。

URL重写在IIS级别解决了URL与物理地址的分离，它通过一个基于本地（Native）代码的模块注册到IIS进行HTTP请求处理的管道上，所以可以应用于所以寄宿于IIS中的Web应用。而URL路由系统则是ASP.NET的一部分，是通过托管代码实现的。为了让读者对ASP.NET的URL路由具有一个感官的认识，我们来演示一个简单的实例。

**二、 实例演示：通过URL路由实现请求地址与.aspx页面的映射**

接下来我们将创建一个简单的ASP.NET Web Forms应用，并采用一个独立于.aspx文件路径的URL来访问对应的Web页面，而两者之间的映射通过URL路由来实现。我们是一个关于员工管理的场景，我们将创建一个页面来显示员工的列表和某个员工的详细信息，页面呈现出来效果如下图所示。

[](http://images.cnblogs.com/cnblogs_com/artech/201203/201203190742342730.jpg)

我们将关注点放到上图所示的两个页面的URL上。用于显示员工列表的页面地址为<http://localhost:2738/employees>。当用户点击某个显示为姓名的连接后，用于显示所选员工详细信息的页面被呈现出现，其页面地址的URL模式为http://localhost:2738/employees/{姓名}/{ID}。对于后者，最终用户一眼可以从URL中看出通过该地址获取的是哪个员工的信息。有人可能会问，为什么我们要在URL同时包含员工的姓名和ID呢？这是因为ID（本例采用GUID）的可读性不如员工姓名，但是员工姓名不具有唯一性，在这里我们使用的ID是为了逻辑处理的需要而提供的唯一标识，而姓名则是出于可读性的需要。

我们将员工的所有 信息（ID、姓名、性别、出生日期和所在部门）定义在如下所示的Employee类型中。我们照例定义了如下一个EmployeeRepository类型表示维护员工列表的领域模型。维护的员工列表通过静态字段employees 表示。EmployeeRepository的GetEmployees方法根据指定的ID返回指包含相应员工的列表，如果指定的ID为“\*”，则返回所有员工列表

1: public class Employee

2: {

3: public string Id { get; private set; }

4: public string Name { get; private set; }

5: public string Gender { get; private set; }

6: public DateTime BirthDate { get; private set; }

7: public string Department { get; private set; }

8:

9: public Employee(string id, string name, string gender, DateTime birthDate, string department)

10: {

11: this.Id = id;

12: this.Name = name;

13: this.Gender = gender;

14: this.BirthDate = birthDate;

15: this.Department = department;

16: }

17: }

18: public class EmployeeRepository

19: {

20: private static IList<Employee> employees;

21: static EmployeeRepository()

22: {

23: employees = new List<Employee>();

24: employees.Add(new Employee(Guid.NewGuid().ToString(), "张三", "男",new DateTime(1981, 8, 24), "销售部"));

25: employees.Add(new Employee(Guid.NewGuid().ToString(), "李四", "女",new DateTime(1982, 7, 10), "人事部"));

26: employees.Add(new Employee(Guid.NewGuid().ToString(), "王五", "男",new DateTime(1981, 9, 21), "人事部"));

27: }

28: public IEnumerable<Employee> GetEmployees(string id = "")

29: {

30: return employees.Where(e => e.Id == id || string.IsNullOrEmpty(id) || id=="\*");

31: }

32: }

对于如上图所示的两个页面实际上对应着同一个.aspx文件，即作为Web应用默认页面的Default.aspx。要通过一个独立于物理路径的URL来访问该.aspx页面，我们就需要采用URL路由机制来实现两者之间的映射。为此我们在添加的Global.asax文件中编写了如下几行代码。如下面的代码片断所示，在Application\_Start方法中我们通过System.Web.Routing.RouteTable的Routes属性得到了表示路由对象列表的System.Web.Routing.RouteCollection对象，并调用该列表对象的MapPageRoute方法将Default.aspx页面（~/Default.aspx）与一个URL模板（employees/{name}/{id）进行了映射。

1: public class Global : System.Web.HttpApplication

2: {

3: protected void Application\_Start(object sender, EventArgs e)

4: {

5: var defaults = new RouteValueDictionary{{"name","\*"},{"id","\*"}};

6: RouteTable.Routes.MapPageRoute("", "employees/{name}/{id}", "~/Default.aspx", true,defaults);

7: }

8: }

作为MapPageRoute方法最后一个参数的RouteValueDictionary对象用于指定定义在路由模板中相应变量（{name}和{id}）的默认值。对于指定了默认值的路由对象，在当前请求地址的后续部分缺失的情况下，它会采用提供的默认值对该地址进行填充之后再进行模式的匹配。在如上所示的代码片断中，我们将{name}和{id}两变量的默认值均指定为“\*”。对于针对URI为http://localhost:2738/employees的请求，我们注册的路由对象会将其格式成http://localhost:2738/employees/\*/\*，后者无疑是和定义的URL模式变现出来的模式是匹配的。

在Default.aspx页面中，我们分别采用GridView和DetailsView来显示所有员工列表和某个列表的详细信息，下面的代码片断表示该页面主体部分的HTML。值得一提的是：GridView模板中显示为员工姓名的HyperLinkField的连接采用了上面我们定义在URL模板（employees/{name}/{id））中的模式。

1: <form id="form1" runat="server">

2: <div id="page">

3: <asp:GridView ID="GridViewEmployees" runat="server" AutoGenerateColumns="false" Width="100%">

4: <Columns>

5: <asp:HyperLinkField HeaderText="姓名" DataTextField="Name" DataNavigateUrlFields="Name,Id" DataNavigateUrlFormatString="~/employees/{0}/{1}" />

6: <asp:BoundField DataField="Gender" HeaderText="性别" />

7: <asp:BoundField DataField="BirthDate" HeaderText="出生日期" DataFormatString="{0:dd/MM/yyyy}" />

8: <asp:BoundField DataField="Department" HeaderText="部门" />

9: </Columns>

10: </asp:GridView>

11: <asp:DetailsView ID="DetailsViewEmployee" runat="server" AutoGenerateRows="false" Width="100%">

12: <Fields>

13: <asp:BoundField DataField="ID" HeaderText= "ID" />

14: <asp:BoundField DataField="Name" HeaderText= "姓名" />

15: <asp:BoundField DataField="Gender" HeaderText="性别" />

16: <asp:BoundField DataField="BirthDate" HeaderText="出生日期" DataFormatString="{0:dd/MM/yyyy}" />

17: <asp:BoundField DataField="Department" HeaderText="部门" />

18: </Fields>

19: </asp:DetailsView>

20: </div>

21: </form>

Default.aspx页面的整个后台代码定义如下。由于所有员工列表和单一员工的详细信息均体现在该页面中，所以我们需要根据其请求地址来判断应该呈现怎样的数据，而这可以通过RouteData属性表示的路由数据来实现。Page具有一个类型为System.Web.Routing.RouteData的RouteData表示通过注册的与当前请求匹配的路由对象对请求地址进行解析生成的路由数据。RouteData的Values属性是一个存储路由变量的字典，其Key为变量名称。在如下所示的代码片断中，我们得到表示员工ID的路由变量（RouteData.Values["id"]），如果它是默认值则表示当前请求是针对员工列表的，反之则是这对指定的某个具体员工的。

1: public partial class Default : Page

2: {

3: private EmployeeRepository repository;

4: public EmployeeRepository Repository

5: {

6: get { return null == repository ? repository = new EmployeeRepository() : repository; }

7: }

8: protected void Page\_Load(object sender, EventArgs e)

9: {

10: if (this.IsPostBack)

11: {

12: return;

13: }

14: string employeeId = this.RouteData.Values["id"] as string;

15: if (employeeId == "\*" || string.IsNullOrEmpty(employeeId))

16: {

17: this.GridViewEmployees.DataSource = this.Repository.GetEmployees();

18: this.GridViewEmployees.DataBind();

19: this.DetailsViewEmployee.Visible = false;

20: }

21: else

22: {

23: var employees = this.Repository.GetEmployees(employeeId);

24: this.DetailsViewEmployee.DataSource = employees;

25: this.DetailsViewEmployee.DataBind();

26: this.GridViewEmployees.Visible = false;

27: }

28: }

29: }

[ASP.NET的路由系统：URL与物理文件的分离](http://www.cnblogs.com/artech/archive/2012/03/19/aspnet-routing-01.html)   
[ASP.NET的路由系统：路由映射](http://www.cnblogs.com/artech/archive/2012/03/20/aspnet-routing-02.html)   
[ASP.NET的路由系统：根据路由规则生成URL](http://www.cnblogs.com/artech/archive/2012/03/21/aspnet-routing-03.html)