**Displaying a Table of Database Data**

이 튜토리얼에서, 데이터베이스 레코드를 표현하는 두 가지 메소드를 설명할 것입니다. HTML 테이블에서 데이터베이스 레코드의 집합을 두 가지 형식으로 설명할 것입니다. 첫 번째로, view에서 바로 데이터베이스 레코드를 접근하는 것과, 다음으로

**Create the Model Classes**

Movies 데이터베이스 테이블의 레코드들을 화면에 나타낼 것입니다. Movies 데이터베이스 테이블은 다음과 같은 칼럼으로 구성되어 있습니다.

|  |  |  |
| --- | --- | --- |
| **Column Name** | **Data Type** | **Allow Nulls** |
| Id | Int | False |
| Title | Nvarchar(200) | False |
| Director | NVarchar(50) | False |
| DateReleased | DateTime | False |

Movies 데이터베이스 테이블을 나타내기 위해, 데이터베이스 액세스 기술로 LINQ to SQL을 사용할 것입니다. 즉, LINQ to SQL을 사용하여 MVC model을 구현할 것입니다.

LINQ to SQL 클래스를 만드는 가장 빠른 방법은 Visual Studio Object Relational Designer을 사용하는 것입니다. Model 폴더에서 우 클릭을 한 후, LINQ to SQL 클래스 템플릿을 추가하십시오.



Movie LINQ to SQL 클래스를 만들고 나면, Object Relational Designer가 나타날 것입니다. 서버 탐색기로부터 데이터베이스 테이블을 Object Relational Designer 로 드래그하십시오.



기본적으로, Object Relational Designer은 데이터베이스 테이블 이름을 단수형으로 단수 형으로 만듭니다. 예를 들어, Customers 라는 데이터베이스 테이블이 있을 때, 드래그 하여 추가하면 Object Relational Designer은 Customer라 이름을 만들 것입니다.

Object Relational Designer에서 테이블 이름 변경은 가능합니다.

**Using LINQ to SQL in a Controller Action**

LINQ to SQL 클래스를 만든 것은, 데이터베이스로부터 데이터를 가져올 클래스가 생겼다는 것입니다. 아래의 코드의 controller 클래스는 Movies 데이터베이스 테이블로부터 movie 레코드를 가져오기 위해 LINQ to SQL 클래스를 사용합니다.

|  |
| --- |
| using System.Linq;using System.Web.Mvc;using MvcApplication1.Models;namespace MvcApplication1.Controllers{ [HandleError] public class HomeController : Controller { public ActionResult Index() { var dataContext = new MovieDataContext(); var movies = from m in dataContext.Movies select m; return View(movies); } }} |

Index() action은 데이터베이스 테이블로부터 movies을 가져오기 위해 LINQ to SQL MovieDataContext 클래스를 사용합니다. Movies 목록은 ViewData.Model 속성의 값으로 Index view로 전달됩니다.

**Formatting within a View**

가장 단순하지만, 최고의 방법은 아닌, 데이터베이스 레코드 집합을 형식화 하는 메소드(method of formatting)는 view에서 바로 데이터를 보여줍니다. 예를 들어, 아래 코드의 Index view은 HTML 테이블 내에서 movie 데이터베이스 레코드를 렌더링 합니다.

|  |
| --- |
| <%@ Page Language="C#" MasterPageFile="~/Views/Shared/Site.Master" AutoEventWireup="true" CodeBehind="Index.aspx.cs" Inherits="MvcApplication1.Views.Home.Index" %><%@ Import Namespace="MvcApplication1.Models" %><asp:Content ID="indexContent" ContentPlaceHolderID="MainContent" runat="server"><table><tr> <th>Id</th><th>Title</th><th>Release Date</th></tr><% foreach (Movie m in (IEnumerable)ViewData.Model){ %><tr> <td><%= m.Id %></td> <td><%= Html.Encode(m.Title) %></td> <td><%= m.DateReleased %></td></tr><% } %></table></asp:Content> |

View은 각각의 movie 레코드를 반복하는 foreach 루프 구문으로 구성되어 있습니다. 각각의 movie 속성 값은 테이블의 각 셀마다 표시됩니다. 테이블 행은 HTML <tr> 태그로 만들고 테이블 셀은 HTML <td> 태그로 만듭니다.

Html.Encode() helper 메소드는 값이 표시되기 전에 각 속성 값을 인코딩 합니다. 사용자 입력 값을 받고, 그 입력 값을 가져와 화면에 표시할 때마다, 자바스크립트 인젝션 공격을 막는 것은 매우 중요합니다.

어플리케이션을 실행 시키면, 아래와 같이 Index view가 렌더링 되어 출력됩니다.



그런데, HTML 테이블 형식이 존재하지 않습니다. 그래서 CSS을 사용하여 HTML 테이블의 외관을 향상 시킬 필요가 있습니다.

마스터 페이지의 <head>태그 사이에 다음과 같은 코드를 추가시킵니다.

|  |
| --- |
| <style type="text/css">table{ border-collapse:collapse;}table td, table th{ border: solid 1px black; padding:10px;}</style> |

마스터 페이지에 CSS 코드를 추가한 후, Index view은 다음과 같이 페이지가 보여질 것입니다.



**Formatting within a Partial**

view에서 모든 형식 코드가 동작하기도 하지만, 템플릿으로 특정 부분만 사용할 수도 있습니다. 예를 들어, 다음 코드는 특정 movie 데이터베이스 레코드를 위한 형식입니다. (사용자 정의 컨트롤)

|  |
| --- |
| <%@ Control Language="C#" AutoEventWireup="true" CodeBehind="MovieTemplate.ascx.cs" Inherits="MvcApplication1.Views.Movies.MovieTemplate" %><tr> <td> <%=ViewData.Model.Id%></td> <td> <%=Html.Encode(ViewData.Model.Title)%></td> <td> <%=ViewData.Model.DateReleased.ToString("D")%></td></tr> |

위 코드에서 템플릿은 HTML 테이블의 행으로서 각각 movie 레코드의 형태를 갖추고 있습니다. ViewData.Model 속성은 특정 부분 내에서, 모든 데이터베이스 레코드가 아닌 하나의 데이터베이스 레코드를 나타냅니다.

Movie 클래스의 개체에서 ViewData.Model 속성을 캐스팅하기 위해, 특정 부분을 위한 코드 비하인드 클래스를 만들어야 합니다. 아래 코드에서, 코드 비하인드 클래스는 타입 파라메터로 Movie을 가지고 있는 제네릭 클래스를 상속 받는 MovieTemplate 클래스를 정의하고 있습니다.

|  |
| --- |
| using MvcApplication1.Models;namespace MvcApplication1.Views.Movies{ public partial class MovieTemplate : System.Web.Mvc.ViewUserControl<Movie> { }} |

MovieTemplate 클래스는 ViewUserControl<movie> 클래스를 상속받습니다. MovieTemplate 클래스가 이 클래스를 상속받기 때문에, ViewData.Model 속성은 자동적으로 Movie을 캐스팅합니다.

아래 코드에서 Index view은 view에서 MovieTemplate을 어떻게 사용하는지 설명하고 있습니다. Html.RenderPartial() 메소드는 MovieTemplate을 렌더링 하기 위해 사용합니다.

|  |
| --- |
| <%@ Page Title="" Language="C#" MasterPageFile="~/Views/Shared/Site.Master" AutoEventWireup="true" CodeBehind="Index.aspx.cs" Inherits="MvcApplication1.Views.Movies.Index" %><%@ Import Namespace="MvcApplication1.Models" %><asp:Content ID="Content1" ContentPlaceHolderID="MainContent" runat="server"><table><tr> <th>Id</th><th>Title</th><th>Release Date</th></tr> <% foreach (Movie m in (IEnumerable)ViewData.Model) { Html.RenderPartial("MovieTemplate", m);  } %></table></asp:Content> |

위 코드에서, Html.RenderPartial() 메소드는 각 Movie 레코드를 나타내기 위해 MovieTemplate을 렌더링 하는데 사용됩니다. 두 파라메터는 RenderPartial() 메소드로 전달합니다. 첫 번째 파라메터는 렌더링 할 템플릿의 이름입니다. 기본적으로, 템플릿은 Views\ControllerName 폴더 또는 Vies\Shared 폴더에 위치되어야 합니다.

두 번째 파라메터는 템플릿에 전달되는 view data 입니다. 템플릿에서, ViewData.Model 속성은 특정 Movie 레코드를 나타냅니다.

하나 중요한 것으로, Html.RenderPartial() 메소드는 다른 Helper 메소드와는 달리, string을 리턴 하지 않는다는 것입니다. 이것은 아래처럼 Renderpartial() 메소드를 호출하지 못한다는 것입니다.

|  |
| --- |
| <%= Html.RenderPartial("MovieTemplate", m) %> |

아래와 같이 사용해야 합니다.

|  |
| --- |
| <% Html.RenderPartial("MovieTemplate", m); %> |

RenderPartial() 메소드는 어떤 일, 행동을 의미하는 것이지, 그것을 표현하는 것은 아니다.