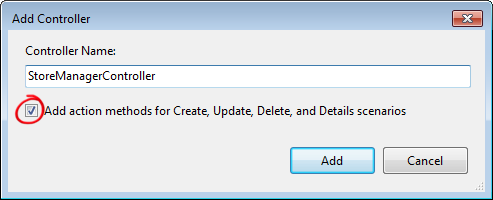
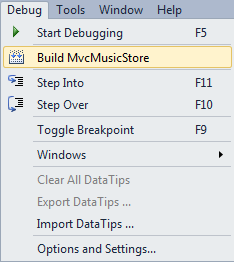
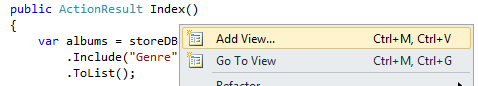
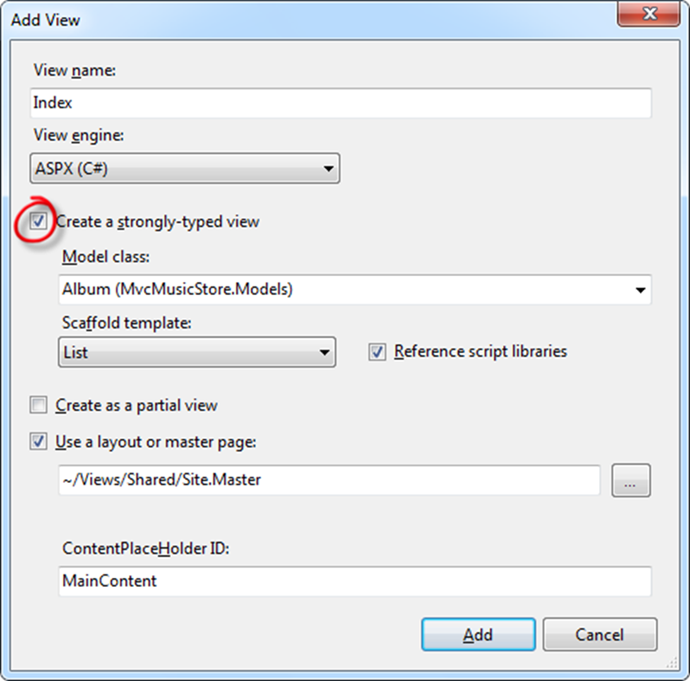
Bài 1: Tạo Store Manager controller và Index view

* 1. Bài này, bạn sẽ học cách tạo một Controller mới có CRUD với sự trợ giúp của Visual, chỉnh sửa Index action để nó trả về danh sách các albums trong cơ sở dữ liệu và tạo Index View để hiển thị danh sách album.
  2. Task 1 – Tạo StoreManagerController
  3. Trong task này, bạn sẽ tạo Controller mới tên **StoreManagerController** .
  4. Mở Microsoft Visual Web Developer 2010 Express: **Start** | **All Programs** | **Microsoft Visual Studio 2010 Express** |**Microsoft Visual Web Developer 2010 Express**.
  5. Trong **File** menu, chọn **Open Project**. Trong hộp thoai Open Project, trỏ đường dẫn tới Source \Begin, chọn **MvcMusicStore.sln** và click **Open**.
  6. Thêm controller: Click chuột phải lên thư mục **Controllers** trong Solution Explorer, chọn **Add 🡪 Controller**. Thay đổi **ControllerName** to **StoreManagerController** và nhớ check vào checkbox **Add action methods for Create, Update, Delete, and Details scenarios** . Click **Add**.
     1. 
     2. Figure 1
     3. Cửa sổ thêm Controller
     4. \* Có thể không cần chọn checkbox **Add action methods for Create, Update, Delete, and Details scenarios** nếu hộp thoại không hiển thị như trên.
  7. Trong lớp Controller vừa mới tạo ra đã có sẵn các phương thưc **Create, Update, Delete, and** **Details**. Từ bây giờ, bạn chỉ cần thêm code sử lý ở sau phần chú thích // TODO
  8. Xóa cái **Details** vì bạn không cần tới nó lúc này.

Task 2 – Chỉnh sửa StoreManager Index

* 1. Trong task này, bạn sẽ chỉnh sửa lại StoreManager Index action để nó trả danh sách album trong cơ sở dữ liệu cho View.
  2. Nếu sử dụng C#, thêm using **MvcMusicStore.Models**:
     1. C#
     2. using System.Web.Mvc;
     3. **using MvcMusicStore.Models;**
  3. Thêm biến **storeDB** thể hiện của **MusicStoreEntities** vào **StoreManagerController**:
     1. C#
     2. public class StoreManagerController : Controller
     3. {
     4. **MusicStoreEntities storeDB = new MusicStoreEntities();**
     5. Visual Basic
     6. PublicClassStoreManagerController
     7. InheritsSystem.Web.Mvc.Controller
     8. **DimstoreDBAsNewMusicStoreEntities**
  4. Trong Index action, ta sử dụng Linq để lấy dữ liệu trong cơ sở dữ liệu.
     1. C#
     2. //
     3. // GET: /StoreManager/
     4. publicActionResult Index()
     5. {
     6. **var albums = storeDB.Albums**
     7. **.Include("Genre").Include("Artist")**
     8. **.ToList();**
     9. **return View(albums);**
     10. }
     11. Visual Basic
     12. '
     13. 'GET: /StoreManager
     14. Function Index() AsActionResult
     15. **Dim albums = storeDB.Albums.Include("Genre").Include("Artist").ToList()**
     16. **Return View(albums)**
     17. EndFunction

Task 3 – Creating the Index View

* 1. Task này, bạn sẽ tạo Index View hiển thị danh sách album mà **StoreManager** Controller trả về.
  2. Trước khi tạo View, bạn nên build the project để **Add View Dialog** biết **Album** class có thể sử dụng . Trên **Debug** menu chọn **Build MvcMusicStore**.
     1. 
     2. Figure 2
     3. Building the project
  3. Click chuột phải vào bên trong hàm **Index** và chọn **Add View** thì có Hộp thoại Add View hiện lên.
     1. 
     2. Figure 3
     3. Adding a View from within the Index method
  4. Trong hộp thoại Add View , sửa tên trong View Name thành **Index**. Check vào **Create a strongly-typed view** checkbox và chọn **Album (MvcMusicStore.Models)** trong **View data class** drop-down. Chọn **List** trong **View content** drop-down. Bạn có thể chọn **View Engine** to **ASPX (C#|VB)**  hoặc **View Engine** khác nếu có, sau đó click **Add**.
     1. 
     2. Figure 4
     3. Thêm Index View

Task 4 – Hiệu chỉnh Index View

* 1. Trong task này, bạn chỉ tạo một trang View hiển thị vài trường dữ liệu đơn giản với sự hỗ trợ của ASP.NET MVC **scaffolding** (định dạng có sẵn).
  2. **Note:** **Saffolding( có thể hiểu là mẫu có sẵn)** được hỗ trợ trong ASP.NET MVC, sẽ tự tạo View với list tất cả các trường dữ liệu trong Album Model. Bạn cũng có thể tụ tạo ra các mẫu khác theo ý mình và cài đạt nó như kiểu mẫu mặc định.
  3. Code HTML được tạo với mẫu có sẵn của ASP .NET MVC
     1. HTML(C#)
     2. <table>
     3. <tr>
     4. <th></th>
     5. <th>
     6. GenreId
     7. </th>
     8. <th>
     9. ArtistId
     10. </th>
     11. <th>
     12. Title
     13. </th>
     14. <th>
     15. Price
     16. </th>
     17. <th>
     18. AlbumArtUrl
     19. </th>
     20. </tr>
     21. <% foreach (var item in Model) { %>
     22. <tr>
     23. <td>
     24. <%: Html.ActionLink("Edit", "Edit", new { id=item.AlbumId }) %> |
     25. <%: Html.ActionLink("Details", "Details", new { id=item.AlbumId }) %> |
     26. <%: Html.ActionLink("Delete", "Delete", new { id=item.AlbumId }) %>
     27. </td>
     28. <td>
     29. <%: item.GenreId %>
     30. </td>
     31. <td>
     32. <%: item.ArtistId %>
     33. </td>
     34. <td>
     35. <%: item.Title %>
     36. </td>
     37. <td>
     38. <%: String.Format("{0:F}", item.Price) %>
     39. </td>
     40. <td>
     41. <%: item.AlbumArtUrl %>
     42. </td>
     43. </tr>
     44. <% } %>
     45. </table>
     46. HTML(Visual Basic)
     47. <table>
     48. <tr>
     49. <th></th>
     50. <th>
     51. GenreId
     52. </th>
     53. <th>
     54. ArtistId
     55. </th>
     56. <th>
     57. Title
     58. </th>
     59. <th>
     60. Price
     61. </th>
     62. <th>
     63. AlbumArtUrl
     64. </th>
     65. </tr>
     66. <% For Each item In Model %>
     67. <tr>
     68. <td>
     69. <%: Html.ActionLink("Edit", "Edit", new with { id=item.AlbumId }) %> |
     70. <%: Html.ActionLink("Details", "Details", new with { id=item.AlbumId }) %> |
     71. <%: Html.ActionLink("Delete", "Delete", new with { id=item.AlbumId }) %>
     72. </td>
     73. <td>
     74. <%: item.GenreId %>
     75. </td>
     76. <td>
     77. <%: item.ArtistId %>
     78. </td>
     79. <td>
     80. <%: item.Title %>
     81. </td>
     82. <td>
     83. <%: String.Format("{0:F}", item.Price) %>
     84. </td>
     85. <td>
     86. <%: item.AlbumArtUrl %>
     87. </td>
     88. </tr>
     89. <% Next %>
     90. </table>
  4. Ở đây chỉ cần hiển thị **Album Title**, **Artist**, và **Genre** nên tao xóa các cột **AlbumId**, **Price** và **Album Art URL** . Đồng thời, ở cột GenreId and ArtistIs sẽ hiển thị tên nên ta thay đổi thành **Artist.Name** và **Genre.Name**. Và xóa đi Details link vì ta không cần.
     1. Thay đổi list trên bằng **<table>** với code bên dưới:
     2. HTML(C#)
     3. **<table>**
     4. **<tr>**
     5. **<th></th>**
     6. **<th>Title</th>**
     7. **<th>Artist</th>**
     8. **<th>Genre</th>**
     9. **</tr>**
     10. **<% foreach (var item in Model) { %>**
     11. **<tr>**
     12. **<td>**
     13. **<%: Html.ActionLink("Edit", "Edit", new { id=item.AlbumId }) %> |**
     14. **<%: Html.ActionLink("Delete", "Delete", new { id=item.AlbumId })%>**
     15. **</td>**
     16. **<td><%: item.Title %></td>**
     17. **<td><%: item.Artist.Name %></td>**
     18. **<td><%: item.Genre.Name %></td>**
     19. **</tr>**
     20. **<% } %>**
     21. **</table>**
     22. HTML(Visual Basic)
     23. **<table>**
     24. **<tr>**
     25. **<th></th>**
     26. **<th>Title</th>**
     27. **<th>Artist</th>**
     28. **<th>Genre</th>**
     29. **</tr>**
     30. **<% For Each item In Model %>**
     31. **<tr>**
     32. **<td>**
     33. **<%: Html.ActionLink("Edit", "Edit", new with { id=item.AlbumId }) %> |**
     34. **<%: Html.ActionLink("Delete", "Delete", new with { id=item.AlbumId })%>**
     35. **</td>**
     36. **<td><%: item.Title %></td>**
     37. **<td><%: item.Artist.Name %></td>**
     38. **<td><%: item.Genre.Name %></td>**
     39. **</tr>**
     40. **<% Next %>**
     41. **</table>**
  5. Thay đổi các mô tả tự phát sinh như Title, Header….

HTML

<asp:Content ID="Content1" ContentPlaceHolderID="TitleContent" runat="server">

Store Manager - All Albums

</asp:Content>

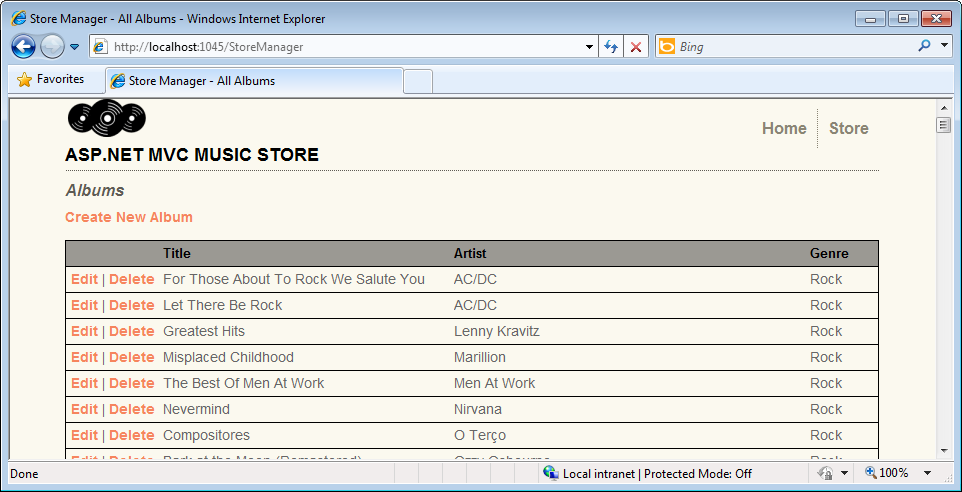
<asp:Content ID="Content2" ContentPlaceHolderID="MainContent" runat="server">

<h2>Albums</h2>

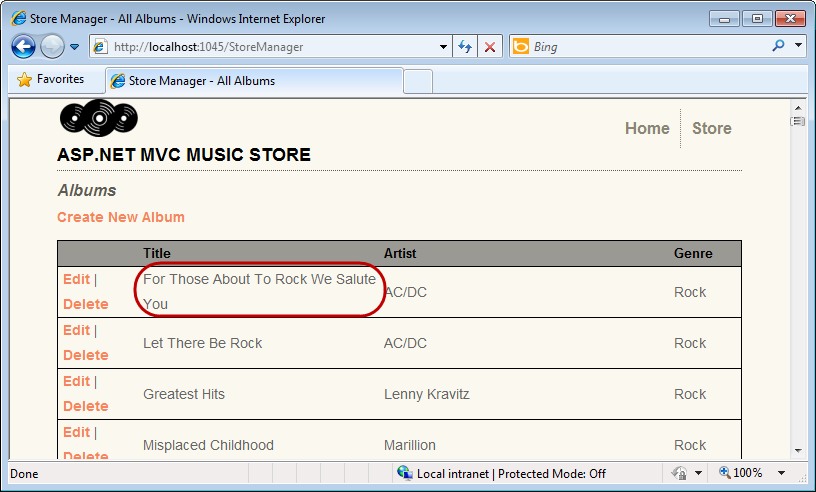
* 1. Thay đổi tên hiển thị đường dẫn thêm Album.
     1. HTML
     2. <asp:Content ID="Content1" ContentPlaceHolderID="TitleContent" runat="server">
     3. Store Manager - All Albums
     4. </asp:Content>
     5. <asp:Content ID="Content2" ContentPlaceHolderID="MainContent" runat="server">
     6. <h2>Albums</h2>
     7. <p>
     8. <%: Html.ActionLink("Create New Album", "Create") %>
     9. </p>

Task 5 – Chạy ứng dụng

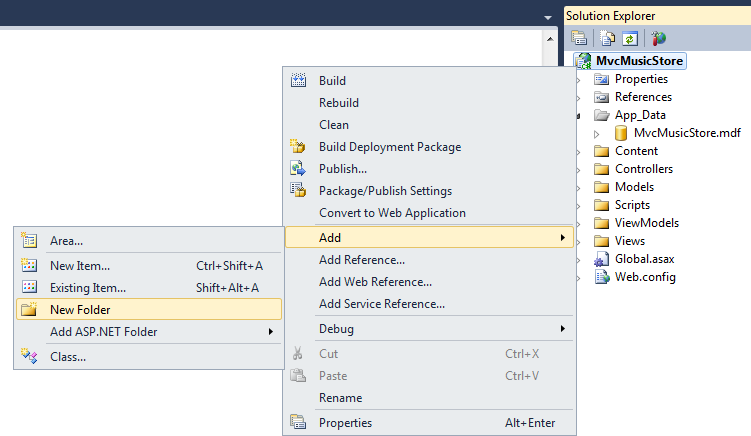
Trong task này, bạn sẽ kiểm tra **StoreManagerIndex** View hiển thị nhứng gì bạn đã làm theo các bước trên.

* 1. Nhấn **F5** để build và run ứng dụng.
  2. Khi chạy project thì mặc định là vào trang Home page. Thêm sau URL **/StoreManager** để xem kết quả. ****
     1. Figure 5 - Trang Index hiển thị danh sách Albums

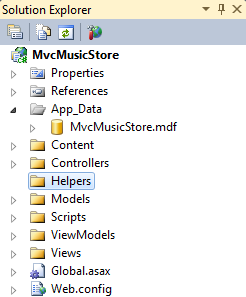
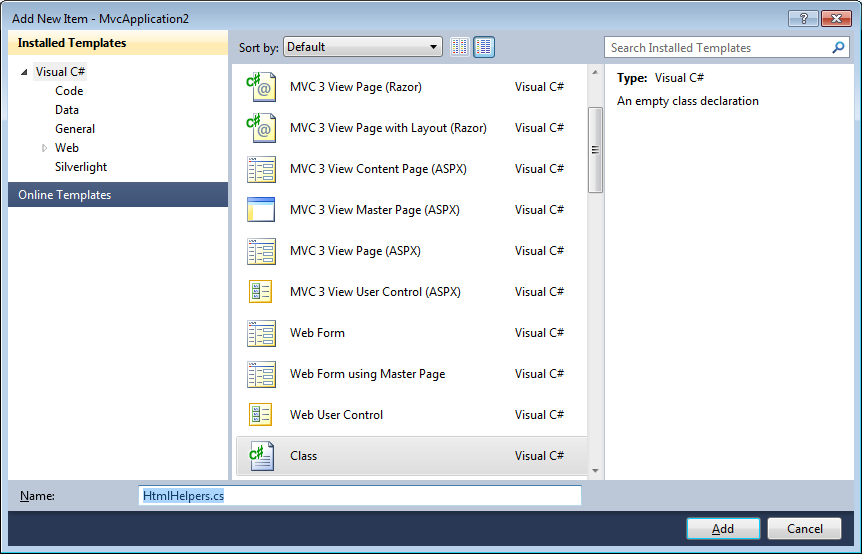
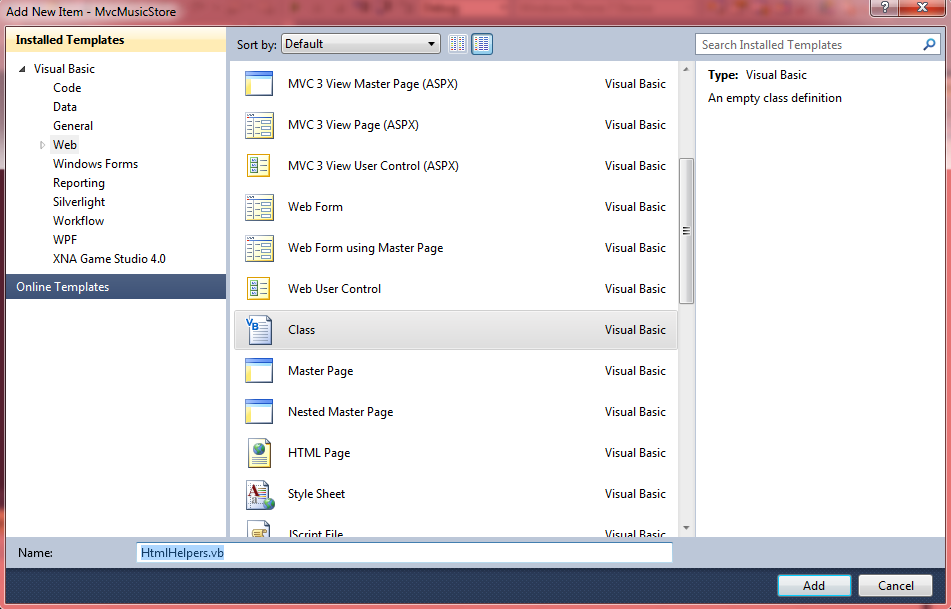
Exercise 2: Thêm HTML Helper

* 1. Trang Index của StoreManager chưa được hoàn chỉnh lắm, nếu như Title and Artist Name quá dài thì nó sẽ làm mấy định dạng table. Trong bài tập này bạn sẽ học cách thêm HTML helper cắt ngắn chuỗi ký tự.
  2. Ở hình bên dưới, bạn sẽ thấy có và vấn đề xảy ra với việc hiển thị dữ liệu không còn đẹp và đúng như ta thiết kế.
  3. 
  4. Figure 6 - Lỗi text quá dài.

Task 1 – Mở rộng HTML Helper

* 1. Trong task này, bạn sẽ thêm phương thức **Truncate** vào trong **Html** trong ASP.NET MVC Views. Để làm được điền này, bạn phải tạo class kế thừa từ **System.Web.Mvc.HtmlHelper.**
  2. **Note:** Bạn có thể tìm hiểu thêm về **System.Web.Mvc.HtmlHelper** tại đây [msdn article.](http://msdn.microsoft.com/en-us/library/bb383977.aspx)
  3. Mở Microsoft Visual Web Developer 2010 Express: **Start** | **All Programs** | **Microsoft Visual Studio 2010 Express** |**Microsoft Visual Web Developer 2010 Express**.
  4. Trên **File** menu, chọn **Open Project**. Trong hộp thoại Open Project, trỏ đường dẫn tới Source\Begin, chọn **MvcMusicStore.sln** sau đó click **Open**.
  5. Thêm thư mục **Helpers** để chứa các phương thức mở rộng: Trong Solution Explorer, click chuột phải **MvcMusicStore** project 🡪 **Add** 🡪**New Folder.**
     1. 
     2. Figure 7

Thêm thư mục Helpers

* 1. Sửa lại tên thư mục mới **Helpers**.
     1. 
     2. Figure 8
     3. Thư mục Helpers
  2. Thêm class **HtmlHelpers** vào thư mục **Helpers** : click chuột phải lên thư mục **Helpers**, chon **Add🡪 Class**. Trong hội thoại **Add New Item**,Thay đổi tên class thành **HtmlHelpers.[cs|vb]** và click **Add**.
     1. 
     2. Figure 9
     3. Thêm class Album – C#
     4. 
     5. Figure 10
     6. Thêm class HtmlHelpers – VB
  3. Thêm namespace **System.Web.Mvc** và các phương thức cho class mới. Vì kế thừa các phương thức từ HtmlHelpers nên các hàm mới phải là hàm static. Code class mới như bên dưới:
     1. C#
     2. using System;
     3. using System.Collections.Generic;
     4. using System.Linq;
     5. using System.Web;
     6. **using System.Web.Mvc;**
     7. **namespace MvcMusicStore.Helpers**
     8. **{**
     9. **public static class HtmlHelpers**
     10. **{**
     11. **public static string Truncate(this HtmlHelper helper, string input, int length)**
     12. **{**
     13. **if (input.Length <= length)**
     14. **{**
     15. **return input;**
     16. **}**
     17. **else**
     18. **{**
     19. **return input.Substring(0, length) + "...";**
     20. **}**
     21. **}**
     22. **}**
     23. **}**
     24. Visual Basic
     25. Public Module HtmlHelpers
     26. <System.Runtime.CompilerServices.Extension()>
     27. Public Function Truncate(ByVal helper As HtmlHelper,
     28. ByVal input As String,
     29. ByVal length As Integer
     30. ) As String
     31. If input.Length <= length Then
     32. Return input
     33. Else
     34. Return input.Substring(0, length) & "..."
     35. End If
     36. End Function
     37. End Module

Task 2 – Đăng kí sử dụng HTML Helper

* 1. Trong Task này, bạn sẽ đăng kí HTML Helper với ứng dụng bằng cách thay đổi file **Web.config** .
  2. Mở file **Web.config** trong Solution Explorer.
  3. Thêm reference **MvcMusicStore.Helpers**  namespace trogn thẻ <**page>**.
     1. XML(C#)
     2. <pages>
     3. <namespaces>
     4. <add namespace="System.Web.Mvc" />
     5. <add namespace="System.Web.Mvc.Ajax" />
     6. <add namespace="System.Web.Mvc.Html" />
     7. <add namespace="System.Web.Routing" />
     8. **<add namespace="MvcMusicStore.Helpers" />**
     9. </namespaces>

</pages>

* + 1. XML(Visual Basic)
    2. <pages>
    3. <namespaces>
    4. <add namespace="System.Web.Mvc" />
    5. <add namespace="System.Web.Mvc.Ajax" />
    6. <add namespace="System.Web.Mvc.Html" />
    7. <add namespace="System.Web.Routing" />
    8. **<add namespace="MvcMusicStore " />**
    9. </namespaces>

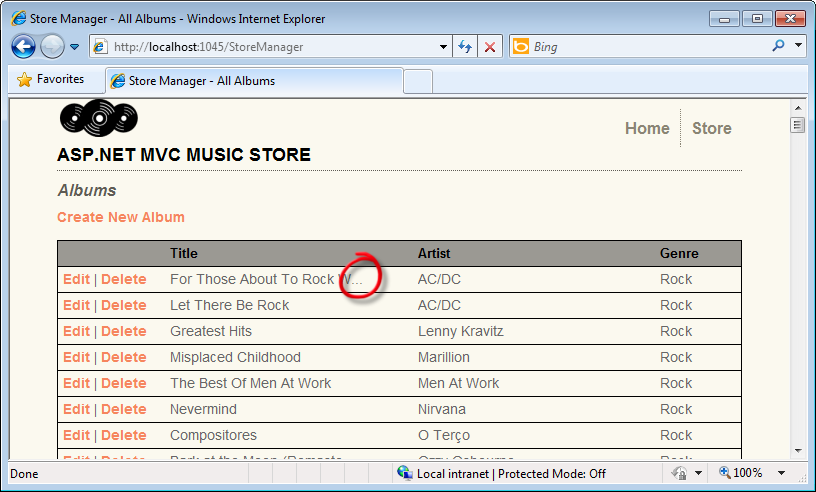
</pages>

Task 3 – Cắt text trong Page

* 1. Trong Task này, bạn sẽ sử dụng **Truncate** trong View.
  2. Mở Index View của StoreManage: Trong Solution Explorer, mở thư mục **Views**, sau đó **StoreManager** và mở file **Index.aspx**.
  3. Chỉnh sửa lại dòng code hiển thị **Title** and **Artist Name** như hình bên dưới:
     1. HTML(C#)
     2. <% foreach (var item in Model) { %>
     4. <tr>
     5. <td>
     6. <%: Html.ActionLink("Edit", "Edit", new { id=item.AlbumId }) %> |
     7. <%: Html.ActionLink("Delete", "Delete", new { id=item.AlbumId })%>
     8. </td>
     9. **<td><%: Html.Truncate(item.Title, 25) %></td>**
     10. **<td><%: Html.Truncate(item.Artist.Name, 25) %></td>**
     11. <td><%: item.Genre.Name %></td>
     12. </tr>
     14. <% } %>
     15. HTML(Visual Basic)
     16. <% For Each item In Model %>
     18. <tr>
     19. <td>
     20. <%: Html.ActionLink("Edit", "Edit", new { id=item.AlbumId }) %> |
     21. <%: Html.ActionLink("Delete", "Delete", new { id=item.AlbumId })%>
     22. </td>
     23. **<td><%: Html.Truncate(item.Title, 25) %></td>**
     24. **<td><%: Html.Truncate(item.Artist.Name, 25) %></td>**
     25. <td><%: item.Genre.Name %></td>
     26. </tr>
     28. <% Next %>

Task 4 – Chạy ứng dụng

Trong task này, bạn sẽ kiểm tra thử hàm **truncate** trong HTML Helper đã hiệu chỉnh lại Title and Artist Name trong **StoreManagerIndex** View.

* 1. Nhấn **F5** để chạy ứng dụng.
  2. Ứng dụng bắt đầu với trang Home mặc định. Thêm vào cuối URL **/StoreManager** để đến trang **StoreManagerIndex** ,xem và so sánh với kết quả trước.
     1. 
     2. Figure 11
     3. Titles và Artist Names đã được cắt ngắn

Exercise 3: Tạo Edit View

Trong bài tập này, bạn sẽ học cách tạo một form cho phép người dùng có thể thay đổi nội dung của Album. Để vào Edit View thì đường dẫn URL trên trình duyệt sẽ là **/StoreManager/Edit/id** (trong đó **id** sẽ là khóa của Album), đây sẽ là HTTP-GET.

Phương thức Edit trong Controller sẽ lấy dữ liệu Album trong cơ sở dữ liệu, theo mặc định thì nó sẽ lấy và hiển thị hết tất cả các trường trong bảng Abbum, điều đó đôi lúc không như ý muốn. Để khắc phục điều đó bạn cần phải thêm **StoreManagerViewModel** (nó chứa các trường của Album như Artists and Genres). Sau đó bạn sẽ tạo trang Edit cho người dùng với các textbox, dropdown list… (**form)** .

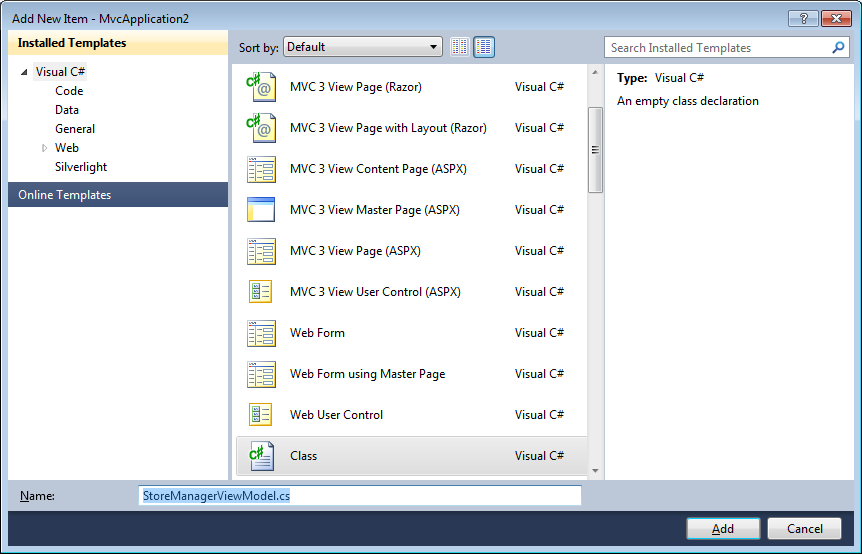
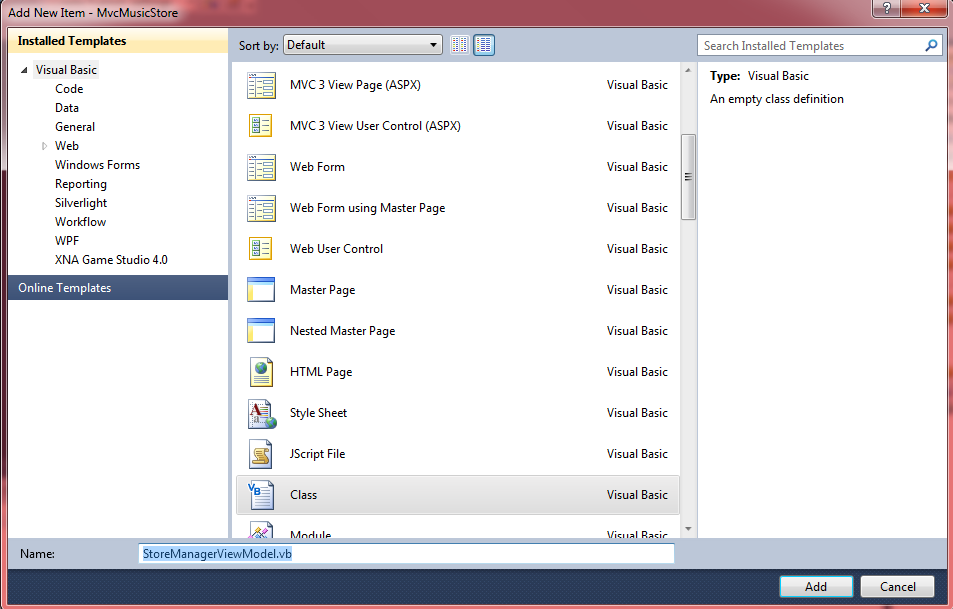
Khi người dùng thay đổi xong thông tin, click button **Save**, lúc này trình duyệt vẫn gọi URL **/StoreManager/Edit/id** nhưng ASP.NET MVC sẽ tự nhận biết đây là HTTP-POST và thực thi phương thức Edit khác trong Controller (phương thức dành cho **[HttpPost]**).

Task 1 – Tạo StoreManagerViewModel

Để thiết kế tốt form Edit, bạn cần hiểu một về Album mà bạn sẽ xây dựng form cho nó:

* 1. Album hiển thị là thông tin hiện tại của nó có trong cơ sở dự liệu
  2. Genres là danh sách tất cả Genre trong cơ sở dữ liệu và hiển thị dưới dạnh dropdown list
  3. Artists là danh sách tất cả Artist trong cơ sở dữ liệu và hiển thị dưới dạnh dropdown list

Trong task này, bạn sẽ tạo mội class mới StoreManagerViewModel giúp ta quản lí toàn bộ dữ liệu bên dưới. Class này sẽ được dùng ở cả Edit and Create.

* 1. Mở Microsoft Visual Web Developer 2010 Express: **Start** | **All Programs** | **Microsoft Visual Studio 2010 Express** |**Microsoft Visual Web Developer 2010 Express**.
  2. Trên menu **File**, chọn **Open Project**. Trong hộp thoại Open Project, trỏ đường dẫn tới Source\Begin, chọn **MvcMusicStore.sln** và click **Open**.
  3. Tạo class **StoreManagerViewModel** : click chuột phải lên thư mục **ViewModels**, chọn **Add** và thêm **Class** mới.
  4. Thay đổi tên class thành **StoreManagerViewModel.[cs|vb]** và click **Add.**
     1. 
     2. Figure 12
     3. Tạo class StoreManagerViewModel – C#
     4. 
     5. Figure 13
     6. Tạo class StoreManagerViewModel - VB
  5. Nếu bạn sử dụng C#, thêm vào các namespace sau:
     1. C#
     2. using System;
     3. using System.Collections.Generic;
     4. using System.Linq;
     5. using System.Web;
     6. **using System.Web.Mvc;**
     7. **using MvcMusicStore.Models;**
  6. Thêm các thuộc **Album**, **Artists** và **Genres**.
     1. C#
     2. public class StoreManagerViewModel
     3. {
     4. **public Album Album { get; set; }**
     5. **public SelectList Artists { get; set; }**
     6. **public SelectList Genres { get; set; }**
     7. }
     8. Visual Basic
     9. Public Class StoreManagerViewModel
     10. **Public Property Album As Album**
     11. **Public Property Artists As SelectList**
     12. **Public Property Genres As SelectList**
     13. EndClass

Task 2 – Thêm HTTP-GET với action Edit

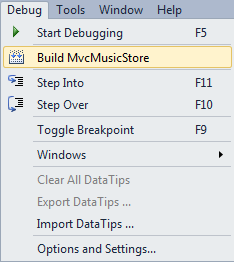
Trong task này,bạn sẽ thêm HTTP-GET vào phương thức action Edit để lấy dữ liệu của Album từ cơ sở dữ liệu, cũng như tất cả các danh sách Genres và Artists. Nó sẽ cho dữ liệu vào StoreManagerViewModel đã định nghĩa ở trên và sẽ hiển thị ở ttang View theo định dạng có sẵn.

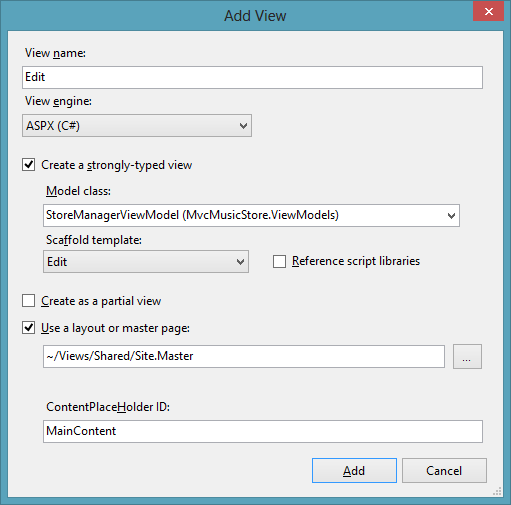
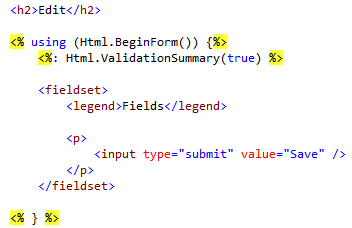
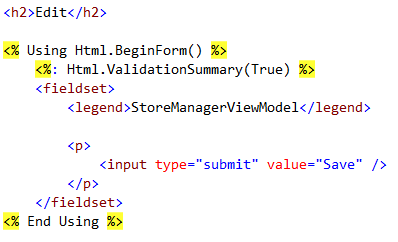
* 1. Để mở class StoreManagerController: Mở thư mục Controllers và nhấp đôi chuột vào StoreManagerController.[cs|vb].
  2. Nếu dùng C#, nhớ thêm using **MvcMusicStore.ViewModels**.
     1. C#
     2. using MvcMusicStore.Models;
     3. **using MvcMusicStore.ViewModels;**
  3. Thay đổi **HTTP-GET** action **Edit** như code bên dưới.
     1. C#
     2. //
     3. // GET: /StoreManager/Edit/5
     5. public ActionResult Edit(int id)
     6. {
     7. **Album album = storeDB.Albums.Single(a => a.AlbumId == id);**
     8. **var viewModel = new StoreManagerViewModel()**
     9. **{**
     10. **Album = album,**
     11. **Genres = new SelectList(storeDB.Genres.ToList(), "GenreId", "Name", album.GenreId),**
     12. **Artists = new SelectList(storeDB.Artists.ToList(), "ArtistId", "Name", album.ArtistId)**
     13. **};**
     14. **return View(viewModel);**
     15. }
     16. Visual Basic
     17. '
     18. 'GET: /StoreManager/Edit/5
     19. **Function Edit(ByVal id As Integer) As ActionResult**
     20. **Dim album As Album = storeDB.Albums.Single(Function(a) a.AlbumId = id)**
     21. **Dim viewModel = New StoreManagerViewModel With**
     22. **{.Album = album,**
     23. **.Genres = New SelectList(storeDB.Genres.ToList(), "GenreId", "Name", album.GenreId),**
     24. **.Artists = New SelectList(storeDB.Artists.ToList(), "ArtistId", "Name", album.ArtistId)}**
     25. **Return View(viewModel)**
     26. End Function

Task 3 – Tạo Edit View

Tong task này, bạn sẽ tạo một mẫu cho Edit View và sẽ hiển thị các thuộc tính của album.

Trước khi tạo View template mới, bạn nên build project để **Add View Dialog** biết các lớp ta xây dựng đã sẵn sàng sử dụng.



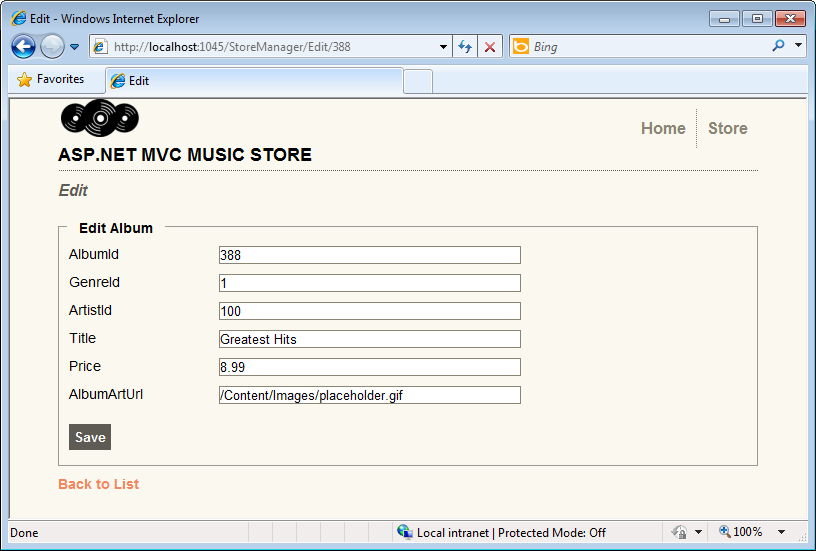
* + 1. Figure 14
    2. Build project
  1. Tạo Edit View: Click chuột phải vào bên trong **Edit** action và chọn **Add View**.
  2. Trong hộp thoại Add View,sửa tên lại thành **Edit**. Check **Create a strongly-typed view** checkbox và chọn **StoreManagerViewModel(MvcMusicStore.ViewModels)** trên **View data class** drop-down. Chọn **Edit** trên **View content** drop-down. Các option khác để mặc định và click **Add**.
     1. 
     2. Figure 15
     3. Thêm Edit View
  3. Edit View được tạo không hiển thị gì từ cơ sở dữ liệu vì StoreManagerViewModel không có gì cả, chưa gắn dữ liệu vào nó.
     1. 
     2. Figure 16
     3. Edit View – C#
     4. 
     5. Figure 17
     6. Edit View - VB

Task 4 – Chỉnh sửa Edit View

* 1. Trong task này, bạn sẽ thay đổi View template mặc định để hiển thị thuộc tính của Album.
  2. Thêm **Html.EditorFor()** để hiển thị HTML editor mặc định của Album.
     1. HTML(C#)
     2. **<legend>Edit Album</legend>**
     3. **<%: Html.EditorFor(model => model.Album,**
     4. **new { Artists = Model.Artists, Genres = Model.Genres })%>**
     6. <p>
     7. <input type="submit" value="Save" />
     8. </p>
     9. HTML(Visual Basic)
     10. **<legend>Edit Album</legend>**
     11. **<%: Html.EditorFor(Function(model) model.Album,**
     12. **New With {Key .Artists = Model.Artists, Key .Genres = Model.Genres })%>**
     14. <p>
     15. <input type="submit" value="Save" />
     16. </p>
     17. **Note:Html.EditorFor()** giúp bạn tạo form mặc định, form này cho phép sửa các thuộc tính có trong **Album**. Tuy nhiên, hai thuộc tính **Artists** và **Genres** sẽ không là dropdown list.

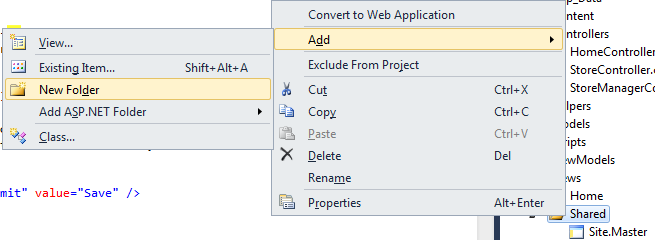
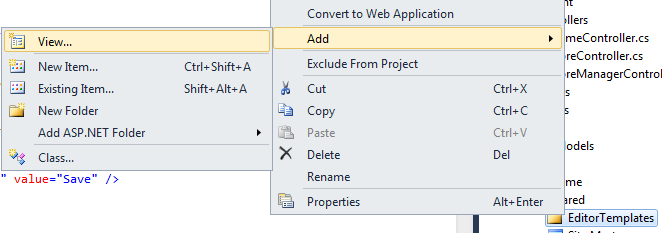
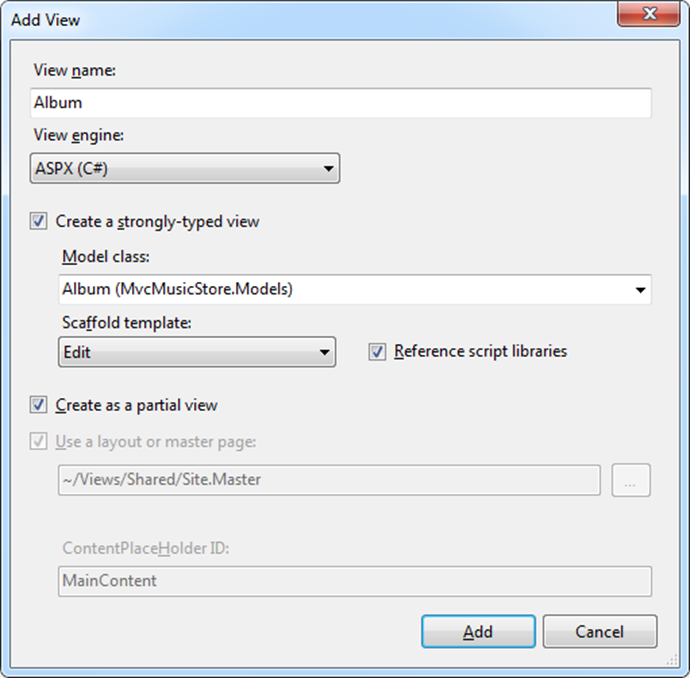
Task 5 – Chạy ứng dụng

Trong task này, bạn sẽ kiểm tra xem trang View **StoreManagerEdit** mặc định.

* 1. Nhấn **F5** để run chạy ứng dụng.
  2. Ứng dụng bắt đầu với trang Home. Thêm vào sau URL **/StoreManager/Edit/388** kiểm tra kết quả.
     1. ****
     2. Figure 18
     3. Album’s Edit view

Task 6 – Tạo một Album Editor Template

Trong task này, bạn sẽ tạo Shared Album Editor template để đảm bảo cho trang Edit View có thẩm mĩ.

* 1. Tạo thư mục bên trong **/Views/Shared** và đặt tên là **EditorTemplates**.
     1. 
     2. Figure 19
     3. Adding a new folder
  2. Click chuột phải lên thư mục **EditorTemplates**, chọn **Add** 🡪 **View**.
     1. 
     2. Figure 20
     3. Thêm View template
  3. **EditorTemplates** này sẽ là Partial View, có nghĩa là **EditorView** này có thể lồng vào trang View khác. Trong hộp thoại Add View, thay đổi tên thành **Album**. Check **Create a partial view (.ascx)** checkbox và **Create a strongly-typed view**checkbox. Chọn **Album (MvcMusicStore.Models)** trên danh sách **Model class** và chọn **Edit** trên danh sách **Scaffold template**. Các tùy chọn khác để mặc định và click **Add**.
     1. 
     2. Figure 21
     3. Thêm Partial View

Task 7 – Thêm danh sách drop-downs trong Album Editor Template

Trong task này, bạn sẽ thêm danh sách drop-downs cho Artists và Genres vào **Album Editor** như đã nói task trước.

* 1. Thay đổi **Album.ascx Partial View** như code bên dưới:
     1. HTML(C#)
     2. **<%@ Control Language="C#" Inherits="System.Web.Mvc.ViewUserControl<MvcMusicStore.Models.Album>" %>**
     3. **<p>**
     4. **<%: Html.LabelFor(model => model.Title) %>**
     5. **<%: Html.TextBoxFor(model => model.Title) %>**
     6. **<%: Html.ValidationMessageFor(model => model.Title) %>**
     7. **</p>**
     8. **<p>**
     9. **<%: Html.LabelFor(model => model.Price) %>**
     10. **<%: Html.TextBoxFor(model => model.Price) %>**
     11. **<%: Html.ValidationMessageFor(model => model.Price) %>**
     12. **</p>**
     13. **<p>**
     14. **<%: Html.LabelFor(model => model.AlbumArtUrl) %>**
     15. **<%: Html.TextBoxFor(model => model.AlbumArtUrl) %>**
     16. **<%: Html.ValidationMessageFor(model => model.AlbumArtUrl) %>**
     17. **</p>**
     18. **<p>**
     19. **<%: Html.LabelFor(model => model.Artist) %>**
     20. **<%: Html.DropDownList("ArtistId", (SelectList) ViewData["Artists"]) %>**
     21. **</p>**
     22. **<p>**
     23. **<%: Html.LabelFor(model => model.Genre) %>**
     24. **<%: Html.DropDownList("GenreId", (SelectList) ViewData["Genres"]) %>**
     25. **</p>**
     26. HTML(Visual Basic)
     27. <%@ Page Title="" Language="VB" MasterPageFile="~/Views/Shared/Site.Master"
     28. Inherits="System.Web.Mvc.ViewPage(Of MvcMusicStore.Album)" %>
     29. <p>
     30. <%:Html.LabelFor(Function(model) model.Title)%>
     31. <%:Html.TextBoxFor(Function(model) model.Title)%>
     32. <%:Html.ValidationMessageFor(Function(model) model.Title)%>
     33. </p>
     34. <p>
     35. <%:Html.LabelFor(Function(model) model.Price)%>
     36. <%:Html.TextBoxFor(Function(model) model.Price)%>
     37. <%:Html.ValidationMessageFor(Function(model) model.Price)%>
     38. </p>
     39. <p>
     40. <%:Html.LabelFor(Function(model) model.AlbumArtUrl)%>
     41. <%:Html.TextBoxFor(Function(model) model.AlbumArtUrl)%>
     42. <%:Html.ValidationMessageFor(Function(model) model.AlbumArtUrl)%>
     43. </p>
     44. <p>
     45. <%:Html.LabelFor(Function(model) model.Artist)%>
     46. <%:Html.DropDownList("ArtistId", CType(ViewData("Artists"), SelectList))%>
     47. </p>
     48. <p>
     49. <%:Html.LabelFor(Function(model) model.Genre)%>
     50. <%:Html.DropDownList("GenreId", CType(ViewData("Genres"), SelectList))%>
     51. </p>
     52. **Note:** **Html.DropDownList** giúp ta thêm drop-downs để chọn Artists and Genres. Các tham số của **Html.DropDownList** là:
     53. 1. Tên hiển thị (**“ArtistId”**)
     54. 2. **SelectList** sẽ chứa các giá trị drop-down, các giá trị này lấy từ **ViewData**:
     55. **C#**
     56. **<%: Html.EditorFor(model => model.Album,**
     57. **new { Artists = Model.Artists, Genres = Model.Genres })%>**

**Visual Basic**

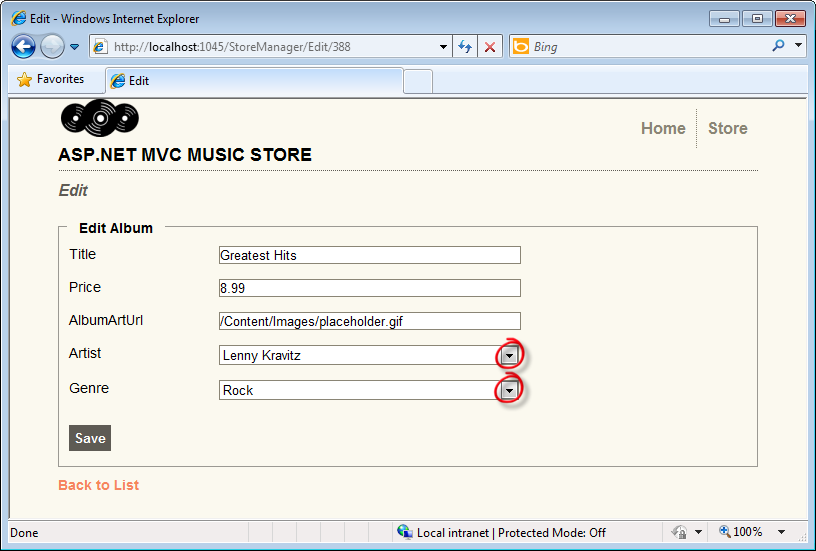
**<%:Html.EditorFor(Function(model) model.Album,**

**New With {Key .Artists = Model.Artists, Key .Genres = Model.Genres})%>**

* + 1. **Html.EditorFor()** giúp bạn gọi hiển thị form của bạn với các tham **Artists** và **Genre** được đưa vào ViewData trong Editor Form.

Task 8 – Chạy ứng dụng

Trong task này, bạn sẽ kiểm tra trang **StoreManagerEdit** hiển thị drop-downs cho Artist and Genre.

* 1. Nhấn **F5** để chạy ứng dụng.
  2. Ứng dụng bắt đầu với trang Home mặc định. Thêm vào cuối URL **/StoreManager/Edit/388** để xem kết quả được hiển thị.
     1. ****
     2. Figure 22
     3. Browsing Album’s Edit view, this time with dropdowns

Task 9 – Thêm HTTP-POST vào Edit action

* 1. Bây giờ thì Edit View đã khá đẹp, bạn cần thêm HTTP-POST vào Edit Action lưu lại những gì đã thay đổi trên Album.
  2. Mở **StoreManagerController** trong thư mục **Controllers**.
  3. Thay đổi **HTTP-POST Edit** action như code bên dưới:
     1. C#
     2. //
     3. // POST: /StoreManager/Edit/5
     4. [HttpPost]
     5. public ActionResult Edit(int id, FormCollection collection)
     6. {
     7. **var album = storeDB.Albums.Single(a => a.AlbumId == id);**
     8. **try**
     9. **{**
     10. **// Save Album**
     11. **UpdateModel(album, "Album");**
     12. **storeDB.SaveChanges();**
     14. **return RedirectToAction("Index");**
     15. **}**
     16. **catch**
     17. **{**
     18. **// Error occurred - so redisplay the form**
     19. **var viewModel = new StoreManagerViewModel()**
     20. **{**
     21. **Album = album,**
     22. **Genres = new SelectList(storeDB.Genres.ToList(), "GenreId", "Name", album.GenreId),**
     23. **Artists = new SelectList(storeDB.Artists.ToList(), "ArtistId", "Name", album.ArtistId)**
     24. **};**
     25. **return View(viewModel);**
     26. **}**

}

* + 1. Visual Basic

'

'POST: /StoreManager/Edit/5

<HttpPost()>

Function Edit(ByVal id As Integer,

ByVal collection As FormCollection

) As ActionResult

* + 1. **Dim album = storeDB.Albums.Single(Function(a) a.AlbumId = id)**
    2. **Try**
    3. **'Save Album**
    4. **UpdateModel(album, "Album")**
    5. **storeDB.SaveChanges()**
    6. **Return RedirectToAction("Index")**
    7. **Catch**
    8. **'Error occurred - so redisplay the form**
    9. **Dim viewModel = New StoreManagerViewModel With**
    10. **{.Album = album,**
    11. **.Genres = New SelectList(storeDB.Genres.ToList(), "GenreId", "Name", album.GenreId),**
    12. **.Artists = New SelectList(storeDB.Artists.ToList(), "ArtistId", "Name", album.ArtistId)}**
    13. **Return View(viewModel)**
    14. **End Try**

End Function

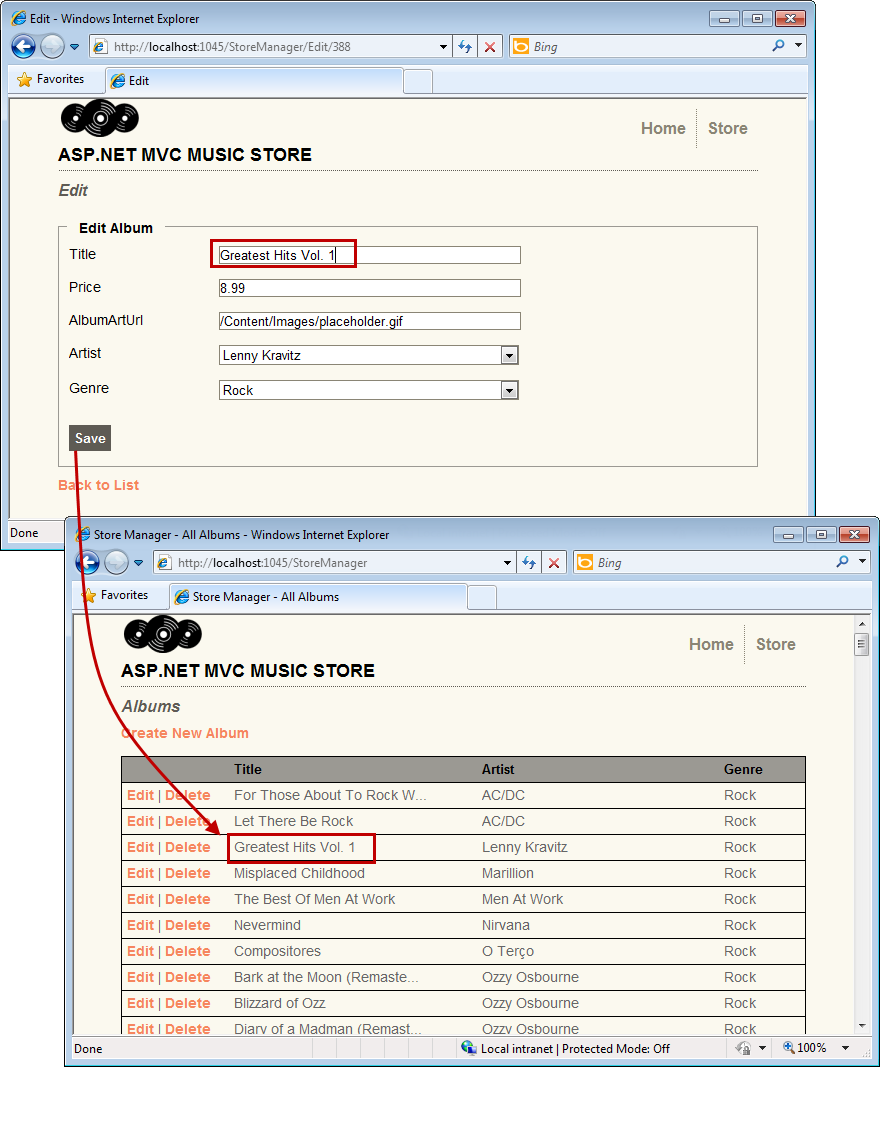
* + 1. **Note:**Phương thức này sẽ thực thi khi người dùng click vào nút **Save** trên trang View và tất cả các giá trị của HTTP-POST form sẽ lưu và cơ sở dữ liệu.
    2. Phương thức HTTP – POST trên làm 3 nhiệm vụ chính:
    3. 1. Load dữ liệu của Album có trong cơ sở dữ liệu với tham số là **id**

2. Cập nhật lại các giá trị từ Client gửi về, và lưu xuống cơ sở dữ liệu.

3.Hiển thị kết quả cho người dùng xem

Task 10 – Chạy ứng dụng

Trong task này, bạn sẽ kiểm tra trang **StoreManagerEdit** có thực sự chỉnh sửa được thong tin và lưu xuống cơ sở dữ liệu thành công.

* 1. Nhấn **F5** để khởi chạy ứng dụng.
  2. Ứng dụng bắt đầu với trang Home mặc định. Thêm vào cuối URL **/StoreManager/Edit/388**. Thay đổi thong tin🡪 save🡪 xem kết quả
     1. 
     2. Figure 23
     3. Update Album

Exercise 4: Thêm Create View

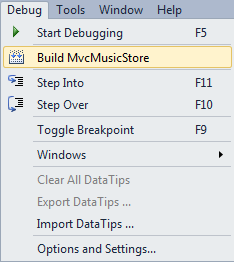
* 1. Trong bài tập này, bạn sẽ học cách thêm view template cho Album trong ứng dụng của chúng ta.
  2. Cũng như Edit action, bạn sẽ tạo hai phương thức cho Create action đó là HTTP-GET và HTTP-POST:
  3. HTTP-GET sẽ thiển thị form Create.
  4. HTTP\_POST sẽ thực thi khi nhấn nút **Save**.

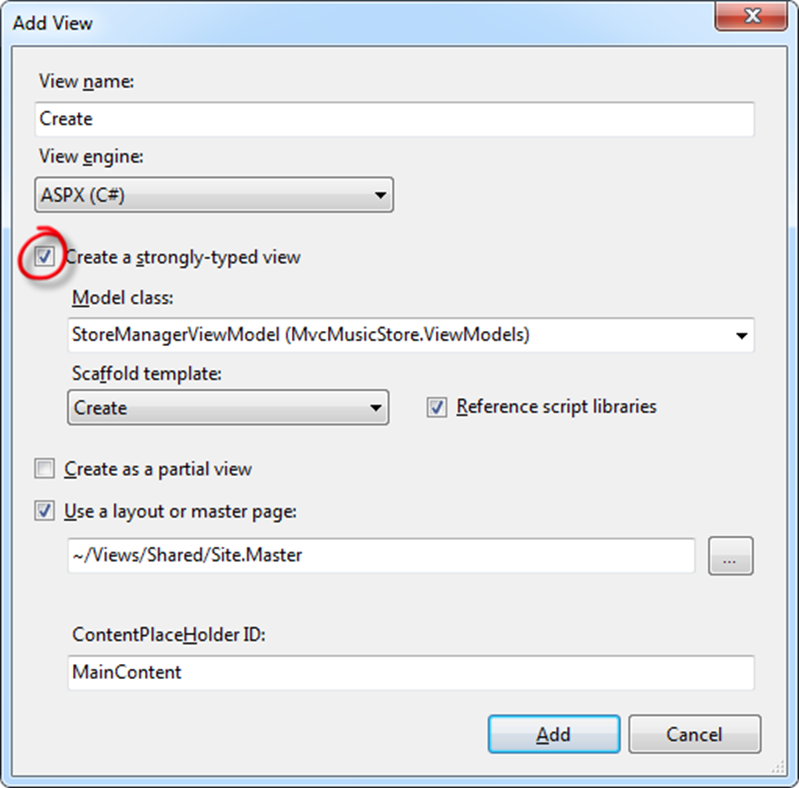
Task 1 – Thêm HTTP-GET Create action

* 1. Trong task này, bạn sẽ thêm HTTP-GET cho Create action, bạn cần lấy danh sáng Genres và Artists trong cơ sở dữ liệu và dùng StoreManagerViewModel để lưu trữ chúng.
  2. Mở Microsoft Visual Web Developer 2010 Express: **Start** | **All Programs** | **Microsoft Visual Studio 2010 Express** |**Microsoft Visual Web Developer 2010 Express**.
  3. Trên **File** menu, chọn **Open Project**. Trong hộp thoại Open Project, trỏ đường dẫn tới Source\Begin, chọn **MvcMusicStore.sln** và click **Open**.
  4. Mở class **StoreManagerController**: expand thư mục **Controllers** và nhấp đôi vào **StoreManagerController.[cs|vb]**.
  5. Thay đổi **Create** action theo code bên dưới:
     1. C#
     2. //
     3. // GET: /StoreManager/Create
     4. public ActionResult Create()
     5. {
     6. **var viewModel = new StoreManagerViewModel()**
     7. **{**
     8. **Album = new Album(),**
     9. **Genres = new SelectList(storeDB.Genres.ToList(), "GenreId", "Name"),**
     10. **Artists = new SelectList(storeDB.Artists.ToList(), "ArtistId", "Name")**
     11. **};**
     13. **return View(viewModel);**
     14. }
     15. Visual Basic
     16. '
     17. 'GET: /StoreManager/Create
     18. Function Create() As ActionResult
     19. **Dim viewModel = New StoreManagerViewModel With**
     20. **{.Album = New Album,**
     21. **.Genres = New SelectList(storeDB.Genres.ToList(), "GenreId", "Name"),**
     22. **.Artists = New SelectList(storeDB.Artists.ToList(), "ArtistId", "Name")}**
     23. **Return View(viewModel)**
     24. End Function

Task 2 – Thêm Create View

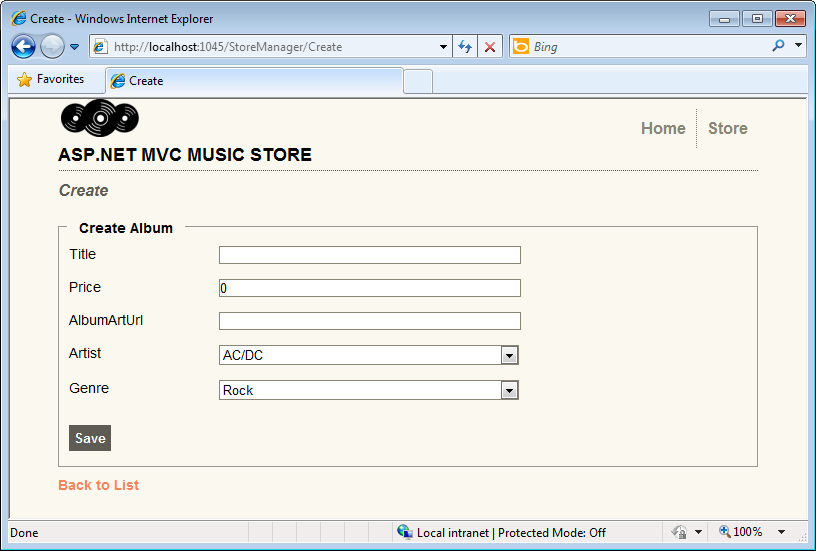
* 1. Trong task này, bạn sẽ thêm Create View template hiển thị như Edit View nhưng các trường không có dữ liệu.
  2. Trước khi tạo View template, bạn nên build project để dảm bảo rằng hộp thoại **Add View Dialog** biết tất cả các class cần sử dụng.



* + 1. Figure 24
    2. Building the project
  1. Click chuột phải trong **Create** action và chọn **Add View**.
  2. Trong hộp thoại Add View, sửa tên trong View Name thành **Create**. Check **Create a strongly-typed view** checkbox và chọn **StoreManagerViewModel (MvcMusicStore.ViewModels)** trên **View data class** drop-down. Chọn **Create** trên **View content** drop-down. Các giá trị còn lại để mặc định và click **Add**.
     1. 
     2. Figure 25
     3. Thêm Edit View
  3. Cập nhật code sử dụng **Html.EditorFor()** như bên dưới:
     1. HTML(C#)
     2. <% using (Html.BeginForm()) {%>
     3. <%: Html.ValidationSummary(true) %>
     4. <fieldset>
     5. **<legend>Create Album</legend>**
     6. **<%: Html.EditorFor(model => model.Album,**
     7. **new { Artists = Model.Artists, Genres = Model.Genres })%>**
     8. **<p>**
     9. **<input type="submit" value="Save" />**
     10. **</p>**
     11. </fieldset>
     12. <% } %>
     13. HTML(Visual Basic)
     14. <% Using Html.BeginForm() %>
     15. <%: Html.ValidationSummary(True) %>
     16. <fieldset>
     17. **<legend>Create Album</legend>**
     18. **<%: Html.EditorFor(Function(model) model.Album,**
     19. **New With {Key .Artists = Model.Artists, Key .Genres = Model.Genres})%>**
     20. **<p>**
     21. **<input type="submit" value="Save" />**
     22. **</p>**
     23. </fieldset>
     24. <% End Using %>

Task 3 – Chạy ứng dụng

Trong task này, bạn sẽ kiểm tra trang **StoreManagerCreate** View hiển thị form tạo Album.

* 1. Nhấn **F5** để chạy ứng dụng.
  2. Ứng dụng bắt đầu với trang Home mặc định. Thêm vào cuối URL **/StoreManager/Create**. ****
     1. Figure 26
     2. Create View with an empty form

Task 4 – Thêm HTTP-POST Create Action

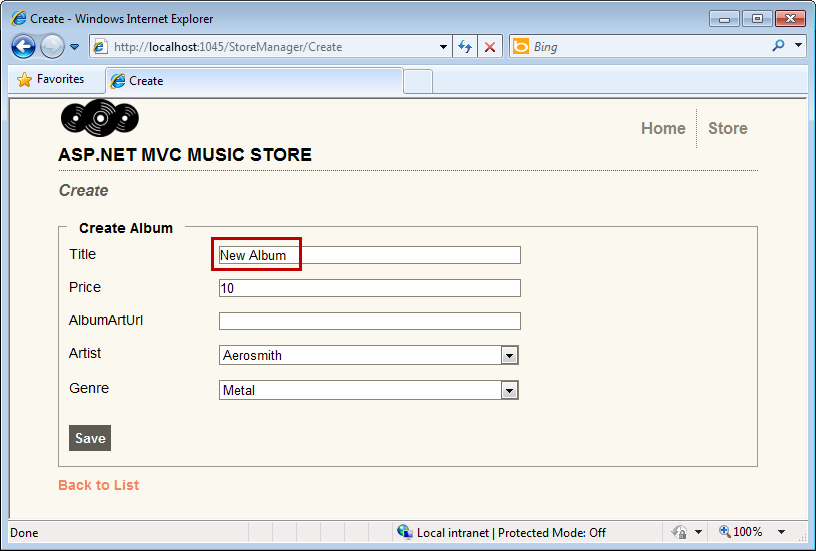
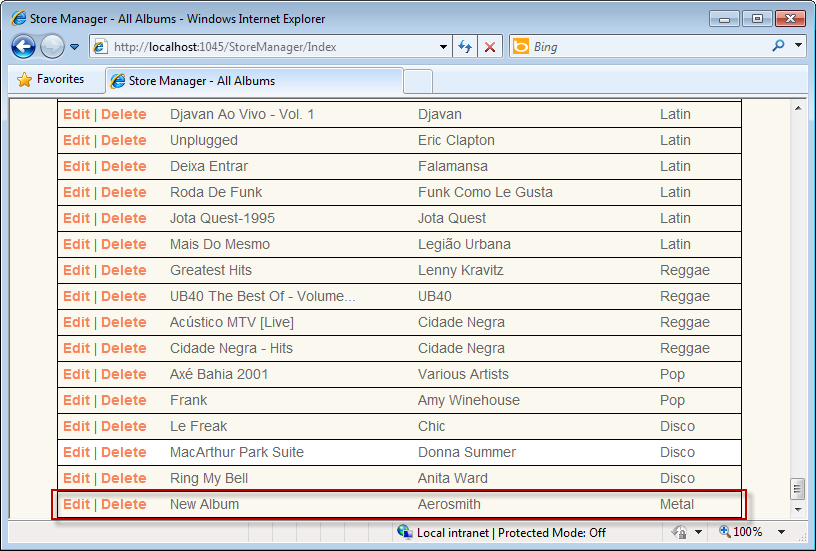
* 1. Trong task này, bạn sẽ thêm HTTP-POST Create action để tạo và lưu lại Album mới khi người dừng nhấn nút **Save** trên form Create.
  2. Mở class **StoreManagerController**: Expand thư mục **Controllers** và nhấp đôi chuột vào **StoreManagerController.[cs|vb]**.
  3. Thay đổi **HTTP-POST Create** action như code bên dưới:
     1. C#
     2. //
     3. // POST: /StoreManager/Create
     4. [HttpPost]
     5. **public ActionResult Create(Album album)**
     6. **{**
     7. **try**
     8. **{**
     9. **if (ModelState.IsValid)**
     10. **{**
     11. **// Save Album**
     12. **storeDB.AddToAlbums(album);**
     13. **storeDB.SaveChanges();**
     14. **return Redirect("Index");**
     15. **}**
     16. **}**
     17. **catch (Exception ex)**
     18. **{**
     19. **ModelState.AddModelError(String.Empty, ex);**
     20. **}**
     21. **// Invalid - redisplay with errors**
     22. **var viewModel = new StoreManagerViewModel()**
     23. **{**
     24. **Album = album,**
     25. **Genres = new SelectList(storeDB.Genres.ToList(), "GenreId", "Name", album.GenreId),**
     26. **Artists = new SelectList(storeDB.Artists.ToList(), "ArtistId", "Name", album.ArtistId)**
     27. **};**
     28. **return View(viewModel);**
     29. **}**
     30. (Code Snippet – ASP.NET MVC 3 Helpers and Forms and Validation – Ex4 StoreManagerController HTTP- POST Create action – VB)
     31. Visual Basic
     32. '
     33. 'POST: /StoreManager/Create
     34. <HttpPost()>
     35. **Public Function Create(ByVal album As Album) As ActionResult**
     36. **Try**
     37. **If ModelState.IsValid Then**
     38. **'Save Album**
     39. **storeDB.AddToAlbums(album)**
     40. **storeDB.SaveChanges()**
     41. **Return Redirect("Index")**
     42. **End If**
     43. **Catch ex As Exception**
     44. **ModelState.AddModelError(String.Empty, ex)**
     45. **End Try**
     46. **'Invalid - redisplay with errors**
     47. **Dim viewModel = New StoreManagerViewModel With**
     48. **{.Album = album,**
     49. **.Genres = New SelectList(storeDB.Genres.ToList(), "GenreId", "Name", album.GenreId),**
     50. **.Artists = New SelectList(storeDB.Artists.ToList(), "ArtistId", "Name", album.ArtistId)}**
     51. **Return View(viewModel)**
     52. **End Function**

**Note:** Điểm khác biệt giữa Edit action và Create action là Edit action thì lấy giữa liệu có trong cơ sở dữ liệu rồi dùng UpdateModel để cập nhâp giá trị cho Album, còn Create action tự tạo Album mới và gán các giá trị từ form Create.

Sau khi kiểm tra giữ liệu thỏa yêu cầu thì Album mới sẽ được lưu vào cơ sở dữ liệu và chuyển ta về trang StoreManager Index.

Task 5 – Chạy ứng dụng

Trong task này, bạn sẽ chạy và kiểm tra trang **StoreManagerCreate** .

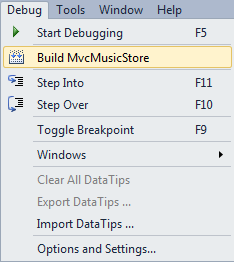
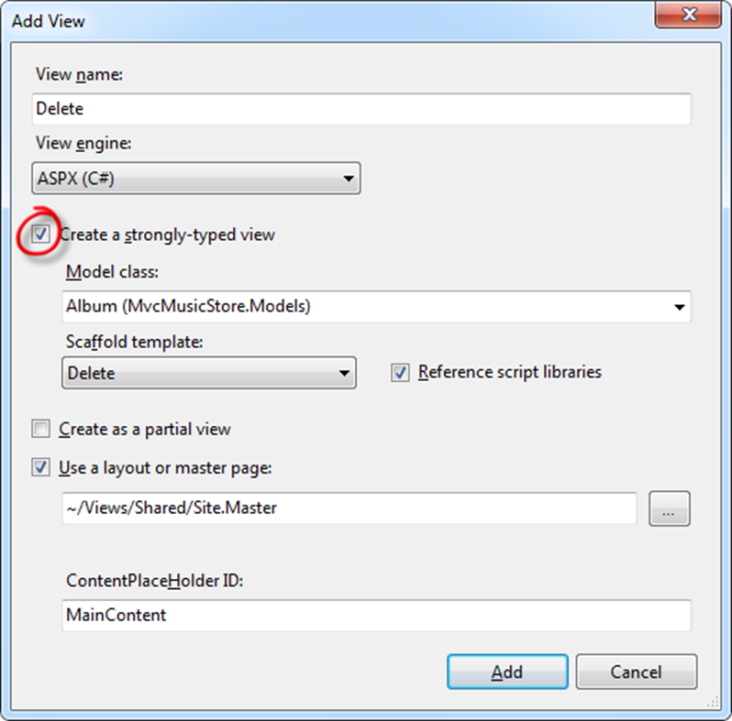
* 1. Nhấn **F5** để chạy.
  2. Ứng dụng bắt đầu với trang Home mặc định. Thêm vào cuối URL **/StoreManager/Create**. Điền đầy đủ thông tin mới cho Album:
     1. ****
     2. Figure 27
     3. Creating an Album
  3. Sau khi hoàn thành thì trình duyệt chạy trang **StoreManager Index**.
     1. ****
     2. Figure 28
     3. New Album created

Exercise 5: Thao tác với xóa Album

* 1. Cũng giống như trước để xóa dữ liệu ta sử dụng 2 phương thức HTTP- GET và HTTP-POST trong **StoreManagerController**:
  2. HTTP-GET hiển thị thông tin sẽ xóa
  3. HTTP-POST xóa dữ liệu khi nhấn nút Submit

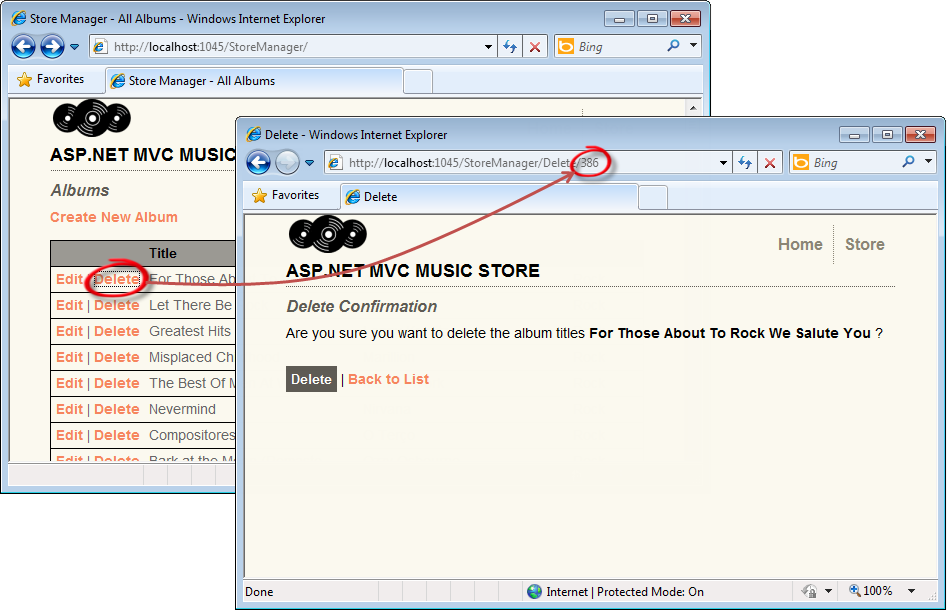
Task 1 – Thêm HTTP-GET DeleteAction

Trong task này, bạn sẽ thêm HTTP-GET Delete action để lấy thông tin của album cần xóa.

* 1. Mở Microsoft Visual Web Developer 2010 Express: **Start** | **All Programs** | **Microsoft Visual Studio 2010 Express** |**Microsoft Visual Web Developer 2010 Express**.
  2. Trong **File** menu, chọn Open **Project**. Trong hộp thoại Open Project, trỏ tới Source\\Begin, select **MvcMusicStore.sln** và click **Open**.
  3. Mở class **StoreManagerController**: Expand thư mục **Controllers** và nhấp đôi vào **StoreManagerController.[cs|vb]**.
  4. Delete action giống như Store Details action: nó lấy **album** từ cơ sở dữ liệu thông qua tham số **id** và hiển thị thông tin lên trang View, Thay đổi HTTP-GET **Delete** action theo code sau:
     1. C#
     2. //
     3. // GET: /StoreManager/Delete/5
     4. **public ActionResult Delete(int id)**
     5. **{**
     6. **var album = storeDB.Albums.Single(a => a.AlbumId == id);**
     7. **return View(album);**
     8. **}**
     9. Visual Basic
     10. '
     11. 'GET: /StoreManager/Delete/5
     12. **Function Delete(ByVal id As Integer) As ActionResult**
     13. **Dim album = storeDB.Albums.Single(Function(a) a.AlbumId = id)**
     14. **Return View(album)**
     15. **End Function**
  5. Trước khi tạo View template, bạn nên build the project để đảm bảo **Add View Dialog** biết tất cả các class cần dùng.
     1. 
     2. Figure 29
     3. Building the project
  6. Click chuột phải trong **Delete** action, chọn **Add View**.
  7. Trong hộp thoại Add View, Thay đổi tên trong View name thành **Delete**. Check **Create a strongly-typed view** checkbox và chọn **Album (MvcMusicStore.Models)** trên **View data class**drop-down. Chọn **Delete** trên **View content** drop-down. Các giá trị còn lại để mặc định và click **Add**.
     1. 
     2. Figure 30
     3. Thêm Delete View
  8. Thay đồi code trang View như sau:
     1. HTML(C#)
     2. <asp:Content ID="Content2" ContentPlaceHolderID="MainContent" runat="server">
     3. **<h2>Delete Confirmation</h2>**
     4. **<p>**
     5. **Are you sure you want to delete the album titles**
     6. **<strong><%: Model.Title %></strong> ?**
     7. **</p>**
     8. **<div>**
     9. **<% using (Html.BeginForm()) { %>**
     10. **<input type="submit" value="Delete" /> |**
     11. **<%: Html.ActionLink("Back to List", "Index") %>**
     12. **<% } %>**
     13. **</div>**
     14. </asp:Content>
     15. HTML(Visual Basic)
     16. <asp:Content ID="Content2" ContentPlaceHolderID="MainContent" runat="server">
     17. **<h2>Delete Confirmation</h2>**
     18. **<p>**
     19. **Are you sure you want to delete the album titles**
     20. **<strong><%: Model.Title %></strong> ?**
     21. **</p>**
     22. **<div>**
     23. **<% using Html.BeginForm()%>**
     24. **<input type="submit" value="Delete" /> |**
     25. **<%: Html.ActionLink("Back to List", "Index") %>**
     26. **<% End Using %>**
     27. **</div>**
     28. </asp:Content>

Task 2 – Chạy ứng dụng

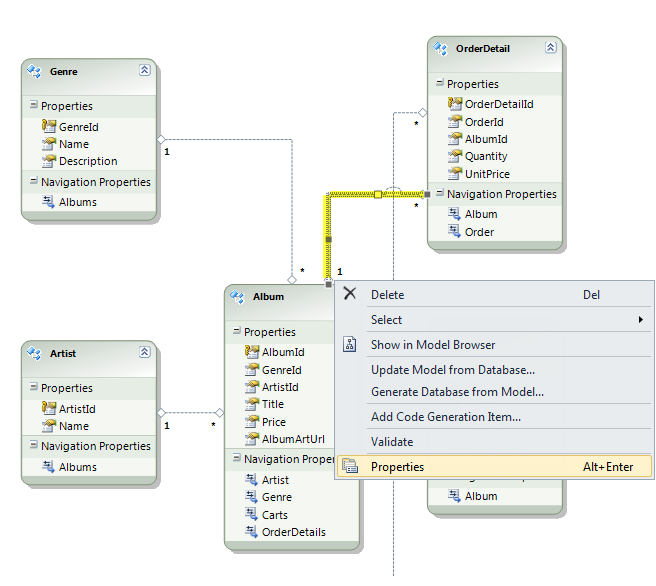
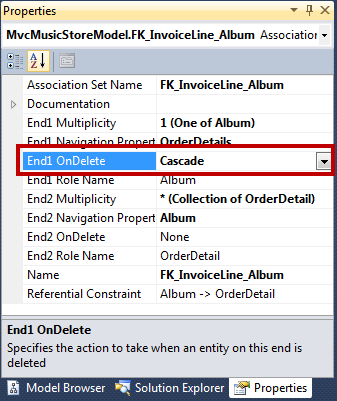
Trong task này, bạn sẽ kiểm tra trang **StoreManagerDelete** View và thử Submit để xóa dữ liệu

* 1. Nhấn **F5** để chạy ứng dụng.
  2. Ứng dụng bắt đầu với trang Home mặc định. Thêm vào cuối URL **/StoreManager**. Chọn album bất kì và click **Delete**.
     1. 
     2. Figure 31
     3. Deleting an Album

Task 3 – Thêm HTTP-POST Delete Action

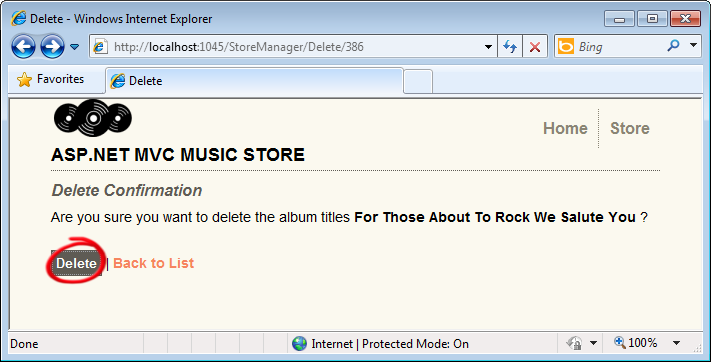
* 1. Trong task này, bạn sẽ thêm HTTP-POST Delete action để xử lí khi người dùng nhấn nút **Delete**.
  2. Mở class **StoreManagerController**:Expand thư mục **Controllers** và nhấp đôi chuột vào **StoreManagerController.[cs|vb]**.
  3. Thay đổi **HTTP-POST Delete** action như code sau:
     1. C#
     2. //
     3. // POST: /StoreManager/Delete/5
     4. [HttpPost]
     5. public ActionResult Delete(int id, FormCollection collection)
     6. {
     7. **var album = storeDB.Albums**
     8. **.Include("OrderDetails").Include("Carts")**
     9. **.Single(a => a.AlbumId == id);**
     10. **storeDB.DeleteObject(album);**
     11. **storeDB.SaveChanges();**
     12. **return RedirectToAction("Index");**
     13. }
     14. (Code Snippet – ASP.NET MVC 3 Helpers and Forms and Validation – Handling Deletion HTTP-POST Delete action – VB)
     15. Visual Basic
     16. '
     17. 'POST: /StoreManager/Delete/5
     18. <HttpPost()>
     19. Function Delete(ByVal id As Integer,
     20. ByVal collection As FormCollection
     21. ) As ActionResult
     22. **Dim album = storeDB.Albums.Include("OrderDetails").Include("Carts").Single(**
     23. **Function(a) a.AlbumId = id)**
     24. **storeDB.DeleteObject(album)**
     25. **storeDB.SaveChanges()**
     26. **Return RedirectToAction("Index")**
     27. End Function

Task 4 – Xóa dây chuyền

* 1. Vì một số ràng buộc trong cơ sở dữ liệu nên khi xóa **Album** có thể phát sinh một số lỗi như album đó có trong **OrderDetails** . Để giải quyết, bạn nên cho phép cơ sở dữ liệu xóa tất cả các dữ liệu liên quan tới Album. Trong task này, bạn sẽ kích hoạt tính năng này - xóa dây chuyền (deletion on cascade).
  2. **Note:** Trong trường hợp bạn xóa Album bất kì đã được đặt hàng từ một khách hàng. Bây giờ bạn quan tâm là action Delete làm việc có như ý muốn của ta nữa không.
  3. Trên Solution Explorer, expand thư mục **Models** và nhấp đôi chuột vào **StoreDB.edmx**. Việc này sẽ mở Entity Data Model designer.
  4. Mở sơ đồ **Models/StoreDB.edmx** . Click chuột phải lên quan hệ giữa **Album và** **OrderDetail** sau đó chọn **Properties**.
     1. 
     2. Figure 32
     3. Chỉnh sửa lại mối quan hệ trên cơ sở dữ liệu
  5. Trong hộp thoại **Properties**, trên **End1 OnDelete** chọn là **Cascade**.
     1. 
     2. Figure 33
     3. OnDelete

Task 5 – Chạy ứng dụng

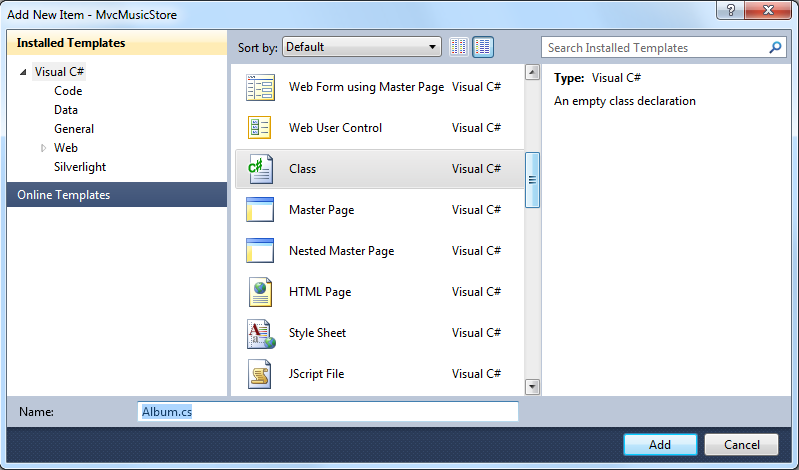
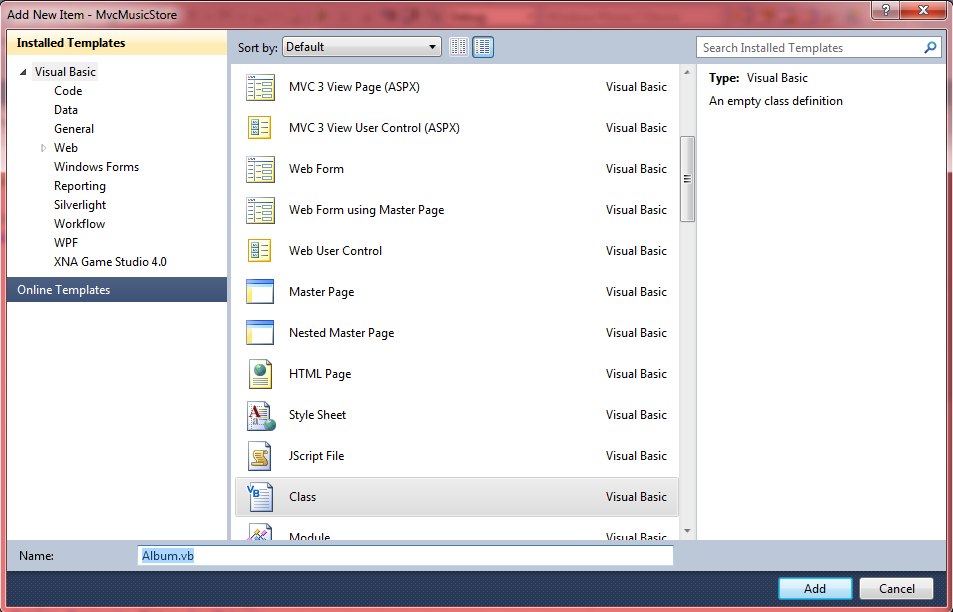
Trong task này, bạn sẽ kiểm tra xem trang **StoreManagerDelete** View xóa Album như thế nào, đã như bạn thiết kế chưa.

* 1. Nhấn **F5** để chạy ứng dụng.
  2. Ứng dụng bắt đầu với trang Home mặc định. Thêm vào cuối URL **/StoreManager**. Xóa thử một vài Album:
     1. 
     2. Figure 34
     3. Xóa Album
  3. Kiểm tra ở trang **Index.aspx** xem thử Album có thực sự xóa chưa.

Exercise 6: Thêm Validation

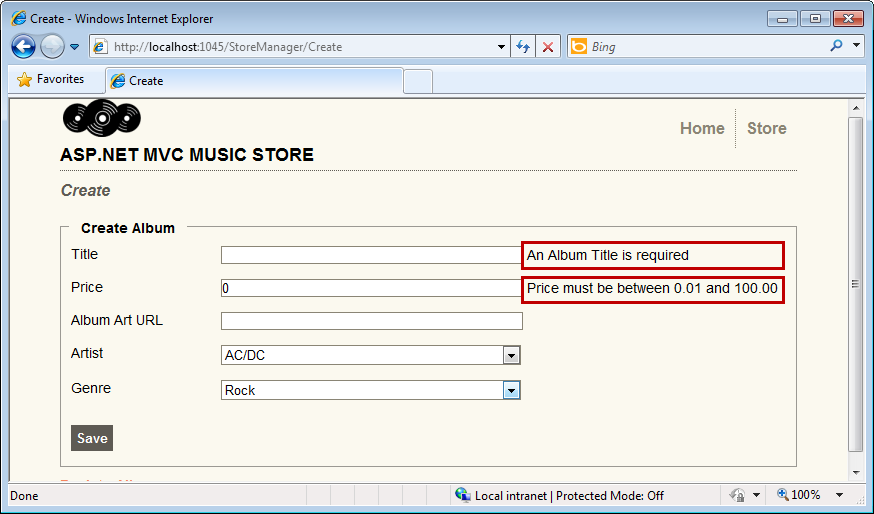
* 1. Hiện tại, form Create and Edit của bạn có xử lý một số validation mà hệ thống tự phát sinh như là: không được bỏ trống, yêu cầu nhập số, giá… để dữ liệu không bị lỗi.
  2. Bạn có thể can thiệp và quản lý hoàn toàn các validation này bằng cách thêm Data Annotations vào class model. Data Annotations cho phép bạn mô tả, định nghĩa các ràng buộc mà bạn muốn lên các thuộc tính của model, và ASP.NET MVC sẽ tự động kiểm tra và phát sinh cảnh báo cho người dùng.

Task 1 – Thêm Data Annotations

* 1. Trong task này, bạn sẽ thêm Data Annotations cho Album Model.
  2. Mở Microsoft Visual Web Developer 2010 Express: **Start** | **All Programs** | **Microsoft Visual Studio 2010 Express** |**Microsoft Visual Web Developer 2010 Express**.
  3. Trên menu **File**, chọn **Open Project**. Trong hộp thoại Open Project, trỏ tới **Source\\Begin**, chọn **MvcMusicStore.sln** và click **Open**.
  4. Click chuột phải lên thư mục **Models** trên Solution Explorer, chọn **And🡪New Item**.
  5. Trong hộp thoại **Add New Item**, chọn **Class** template trong **Visual [C#|Basic] ->Web** template list. Thay đổi tên **Album.[cs|vb]** và click **Add**.
     1. 
     2. Figure 35
     3. Thêm Album class – C#
     4. 
     5. Figure 36
     6. Thêm Album class - VB
  6. Thay đổi code **Album.[cs|vb]** như bên dưới:
     1. C#
     2. using System;
     3. using System.Collections.Generic;
     4. using System.Linq;
     5. using System.Web;
     6. **using System.ComponentModel;**
     7. **using System.ComponentModel.DataAnnotations;**
     8. **using System.Web.Mvc;**
     9. **namespace MvcMusicStore.Models**
     10. **{**
     11. **[MetadataType(typeof(AlbumMetaData))]**
     12. **public partial class Album**
     13. **{**
     14. **// Validation rules for the Album class**
     15. **[Bind(Exclude = "AlbumId")]**
     16. **public class AlbumMetaData**
     17. **{**
     18. **[ScaffoldColumn(false)]**
     19. **public object AlbumId { get; set; }**
     20. **[DisplayName("Genre")]**
     21. **public object GenreId { get; set; }**
     22. **[DisplayName("Artist")]**
     23. **public object ArtistId { get; set; }**
     24. **[Required(ErrorMessage = "An Album Title is required")]**
     25. **[DisplayFormat(ConvertEmptyStringToNull = false)]**
     26. **[StringLength(160)]**
     27. **public object Title { get; set; }**
     28. **[DisplayName("Album Art URL")]**
     29. **[StringLength(1024)]**
     30. **public object AlbumArtUrl { get; set; }**
     31. **[Required(ErrorMessage = "Price is required")]**
     32. **[Range(0.01, 100.00, ErrorMessage = "Price must be between 0.01 and 100.00")]**
     33. **public object Price { get; set; }**
     34. **}**
     35. **}**
     36. **}**
     37. Visual Basic
     38. **Imports System.ComponentModel**
     39. **Imports System.ComponentModel.DataAnnotations**
     40. **<MetadataType(GetType(Album.AlbumMetaData))>**
     41. **Partial Public Class Album**
     42. **'Validation rules for the Album class**
     43. **<Bind(Exclude:="AlbumId")>**
     44. **Public Class AlbumMetaData**
     45. **<ScaffoldColumn(False)>**
     46. **Public Property AlbumId As Object**
     47. **<DisplayName("Genre")>**
     48. **Public Property GenreId As Object**
     49. **<DisplayName("Artist")>**
     50. **Public Property ArtistId As Object**
     51. **<Required(ErrorMessage:="An Album Title is required"),**
     52. **DisplayFormat(ConvertEmptyStringToNull:=False),**
     53. **StringLength(160)>**
     54. **Public Property Title As Object**
     55. **<DisplayName("Album Art URL"), StringLength(1024)>**
     56. **Public Property AlbumArtUrl As Object**
     57. **<Required(ErrorMessage:="Price is required"),**
     58. **Range(0.01, 100.0, ErrorMessage:="Price must be between 0.01 and 100.00")>**
     59. **Public Property Price As Object**
     60. **End Class**
     61. **End Class**
     62. **Note:**.Vài thuộc tính của Data Annotation mà bạn sử dụng cho Album model:
     63. • Required – Yêu cầu phải có dữ liệu, không được bỏ trống
     64. • DisplayName – Tên hiển thị trên form
     65. • DisplayFormat – Định dạng các kiểu hiển thị đặc biệt như password
     66. • StringLength – Yêu cầu độ dài chuỗi
     67. • Range – Giới hạn giá trị số
     68. • Bind – Liên kiết tới miền giá trị khác
     69. • ScaffoldColumn – Cho phép ẩn các trường

Task 2 – Chạy ứng dụng

Trong task này, bạn sẽ kiểm tra trang validate ở trang Create and Edit.

* 1. Nhấn **F5** để chạy ứng dung.
  2. Ứng dụng bắt đầu với trang Home mặc định. Thêm vào cuối URL **/StoreManager/Create**. Thêm giá trị cho Album (cố tình sai nha)
  3. Click **Save**, và bạn sẽ nhận các thông báo validate.
     1. ****
     2. Figure 37
     3. Validate ở trang Creat

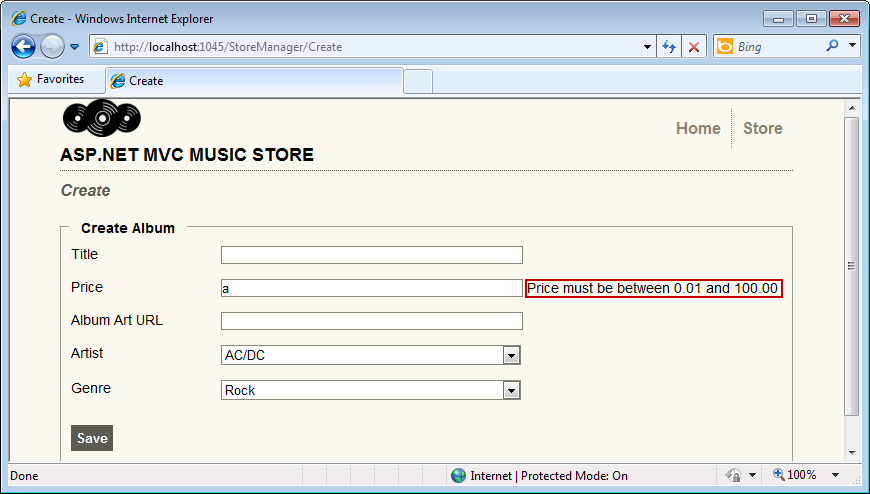
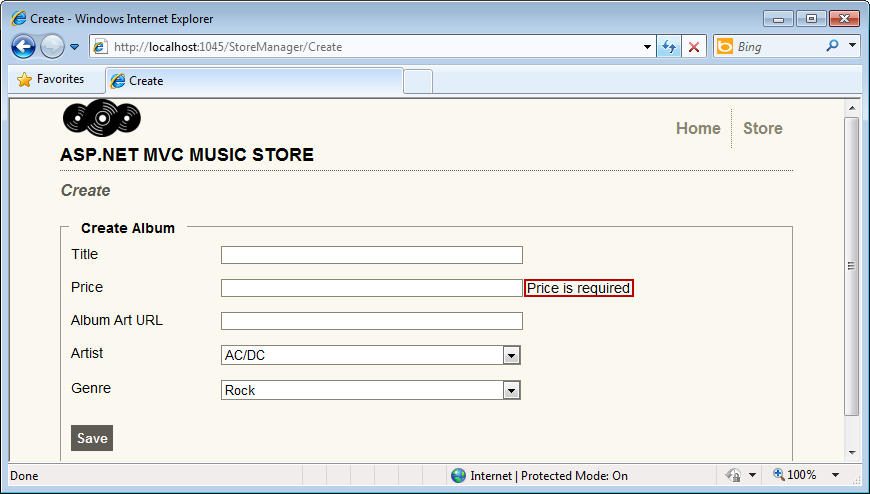
Task 3 – Thêm Client-Side (AJAX) validation

Trong task này, bạn sẽ thêm client-side (AJAX) validation để kiểm tra toàn bộ dữ liệu trước khi gửi cho controller actions.

* 1. Mở **/Views/Shared/EditorTemplates/Album.ascx**
  2. Thêm các thư viện để sử dụng client-side validation:
     1. HTML(C#)
     2. <%@ Control Language="C#" Inherits="System.Web.Mvc.ViewUserControl<MvcMusicStore.Models.Album>" %>
     3. **<script src="/Scripts/MicrosoftAjax.js" type="text/javascript"></script>**
     4. **<script src="/Scripts/MicrosoftMvcAjax.js" type="text/javascript"></script>**
     5. **<script src="/Scripts/MicrosoftMvcValidation.js" type="text/javascript"></script>**
     6. HTML(Visual Basic)
     7. <%@ Page Title="" Language="VB" Inherits="System.Web.Mvc.ViewPage(Of MvcMusicStore.Album)" %>
     8. **<script src="/Scripts/MicrosoftAjax.js" type="text/javascript"></script>**
     9. **<script src="/Scripts/MicrosoftMvcAjax.js" type="text/javascript"></script>**
     10. **<script src="/Scripts/MicrosoftMvcValidation.js" type="text/javascript"></script>**
  3. Sử dụng **Html.EnableClientValidation** helper để hữu hiệu hóa client-side validation. Mở **/Views/StoreManager/Create.aspx** và **/Views/StoreManager/Edit.aspx**, thêm **Html.EnableClientValidation** vào bên trên **Html.BeginForm**:
     1. C#
     2. **<% Html.EnableClientValidation(); %>**
     3. <% using (Html.BeginForm()) {%>
     4. <%: Html.ValidationSummary(true) %>
     5. <fieldset>
     6. Visual Basic
     7. **<%Html.EnableClientValidation()%>**
     8. <% Using Html.BeginForm() %>
     9. <%: Html.ValidationSummary(True) %>
     10. <fieldset>
  4. Chỉnh sửa **Web.config** trong project để vô hiệu hóa UnobtrusiveJavaScript.:
     1. XML
     2. <configuration>
     3. <appSettings>
     4. <add key="ClientValidationEnabled" value="true"/>
     5. <add key="UnobtrusiveJavaScriptEnabled" value="**false**"/>
     6. </appSettings>

Task 4 – Chạy ứng dụng

Trong task này, bạn sẽ kiểm tra vài validate trước khi click submit.

* 1. Nhấn **F5** để chạy ứng dụng.
  2. Ứng dụng bắt đầu với trang Home mặc định. Thêm vào cuối URL **/StoreManager/Create**. Thêm vài giá trị không hợp lệ, nếu nhập sai nó sẽ cảnh báo ngay mà không cần phải submit .
     1. 
     2. Figure 39
     3. Validated client-side
     4. 
     5. Figure 40
     6. Validated client-side