**MVC Tutorial - Developing the Controller 3 - Creating the Delete Facility**

|  |  |  |
| --- | --- | --- |
| **Introduction**  This tutorial will build on the practices and the concepts you were introduced to in the lecture. | | **Required Software**  For this tutorial, you will need access to the following software applications:  MS Explorer  VS.NET  SQL SERVER |
| **Reference**  Walther et al, ‘ASP.NET MVC Framework Unleashed’, SAMS | **Breakdown of Tutorial**  This tutorial consists of the following tasks:   * To develop the Delete facility for a music recording database record. * To extend functionalities for video recordings.   You will be given a series of practical exercises, along with selected questions to consolidate your learning. | | |

**Review questions**

|  |  |
| --- | --- |
| There is a number of *ActionResults*. For the following checkpoints write a snippet of code and point out what its *HelperMethod* is and what it returns. Use an example to explain when we use this. | |
|  | *ViewResult* is one type of ActionResult. |
|  | |
|  | *RedirectResult* is one type of ActionResult. |
|  | |
|  | *FileResult* is one type of ActionResult. |
|  | |
|  | *ContentResult* is one type of ActionResult. |
|  | |
|  | *RedirecttoActionResult* is one type of ActionResult. |
|  | |
|  | *JavascriptResult* is one type of ActionResult. |
|  | |
|  | *JasonResult* is one type of ActionResult. |
|  | |

**Tutorial objectives**

In the previous tutorial, we built the *Fores*t MVC application. This application served some of the purposes of a client. Forest also requires managing database records such as editing an existing record, adding a new record and deleting an existing record for music categories, music recording, video recording, etc. The object of this exercise is to extend admin functionalities to include ability to delete records from the database. For this we will work in the integration layer, service layer and the MVC presentation layer.

**Deleting a Music Recording record**

|  |  |
| --- | --- |
| To delete a record we need to develop a HTML workflow. Developing the workflow consists of the following steps;   1. Sketch the application flow and HTML screens – Define views for the application – Decide what data will be captured from the user – etc. 2. Extend *Forest.Data* to include the integration logic for creating a record. 3. Extend *Forest.Services* to include the creating a record service using the integration layer's functionality. 4. Develop an action method in the *MusicAdminController* controller and the associated view that creates a confirmation-delete page for the *Music\_Recording* object to be deleted. 5. Develop an action method in the *MusicAdminController* controller that results in processing the deleting of the record. | |
| C:\Users\mo\AppData\Local\Microsoft\Windows\Temporary Internet Files\Content.IE5\H5P0LERU\MP900437370[1].jpg | The design of the workflow is up to you - you can design your own application and you should know that although there are patterns we follow; Designing the application workflow is an intuitive process. No two designs are the same though one would definitely work better than the other. However for the purpose of this tutorial we have had to design our workflow. It is important that you understand that workflow before you begin to build the application. Carefully read through 'Deleting a Music Recording record' steps1-5, make notes and sketch the application's workflow. Sketch the application flow and HTML screens – Define views for the application – Decide what data will be captured from the user – etc. |
|  | \_ Sketch the application’s workflow; Sketch the application's workflow; |
|  | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **About the Architecture** | | | | |
| Capture  Always from right to left ... | | | | |
| C:\Users\mo\AppData\Local\Microsoft\Windows\Temporary Internet Files\Content.IE5\H5P0LERU\MP900437370[1].jpg | Data Tire | | Our application tends to work with a variety of data sources including various servers such as SQL Server, Oracle Database, MySQL, XML databases and local database files. In our case, we will make use of a SQL Server Local Database file. This database is integrated into the Integration tier. |
| Model | Integration Tier | The integration tier facilitates access to data sources. Looking at code in this layer we expect to see SQL, LINQ, XPath, etc for querying the data provider and data manipulator objects. The integration tier also facilitates access to 3rd party data services such as Web Services. |
| Service Tier | The service tier is concerned with business logic and data manipulation. We implement all the business rules that the application performs in this layer. Looking at code in this layer, we would expect to see references to integration tier objects as well as implementation of mathematical calculations, conditional decisions, etc. |
| MVC | | MVC consists of the controllers, views, JScripts and the various components that deal with the user interaction. MVC is the presentation layer of the architecture. |

|  |  |  |
| --- | --- | --- |
| **C:\Users\mo\Desktop\Capture.PNG** Extending *Forest.Data* to include the integration logic for deleting a recording. | | |
| \_ Add a delete sub to *IMusicDAO.cs*: |  |
| \_ Develop the sub in *MusicDAO.cs*: |  |
| Build *Forest.Data*. | |

|  |  |  |
| --- | --- | --- |
| **C:\Users\mo\Desktop\Capture1.PNG** Extend ‘*Forest.Services*’ to include a create service using the integration layer functionality. | | |
| Update the reference to ‘*Forest.Data.dll*’ assembly. | |
| \_ Add a delete sub to *IMusicService.cs*: |  |
| \_ Develop the sub in *MusicService.cs*: |  |
| Build ‘*Forest.Services*’. | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Capture** Working on *Forest* project and the *MusicAdminController* controller, create the *DeleteMusicRecording* functionality; | | | | | |
| \_ Develop an action method in the controller that loads a page with the *Music\_Recording* object that is to be deleted. The object of this display is to seek confirmation before the Delete is executed. | | | | | |
| \_ Create the controller *Get* action; | |  | | |
| Working on *Recordings* view;  \_ Add navigation to point to the *MusicAdminController* controller, *DeleteMusicRecording* action, passing *id* as parameter; |  | | |
| \_ Create a view for the *AddMusicRecording* action that displays the input form for the *Music\_Recording* object with the appropriate genre; | | |  |
| \_ Test your application to make sure that this *DeleteMusicRecording* view is rendered, displaying the confirmation page for the *Music\_Recording* object to be deleted. | | | |

|  |  |  |
| --- | --- | --- |
| \_ Develop a *Post* action method in the *MusicAdminController* controller that results in processing the deleting of *Music\_Recording* object. | | |
| \_ Develop the controller *Post* action; |  |

|  |
| --- |
| \_ Test your application. |

**Student-centred exercise**

You are to work on your own to extend the application:

For Music;

• Create Admin facilities for music categories

For Video;

• Create Admin facilities for video recordings