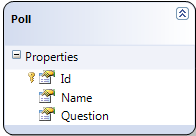
**To Do Of This Tutorial:**

In this tutorial, our goal is to become familiar with the fantastic ready made solution of asp.net mvc for show,insert,edit and delete data. Lets assume, we want to build a poll application. so, we will have to prepare a set of questions and their answers. In this tutorial, we will only see how to show, add, insert and delete questions.

**Database Structure:**

See the simple dbml structure to get the idea of this table. Although you can put the dbml file anywhere you want, its best to put that in the models directory,  
[](http://cdn.codesamplez.com/wp-content/uploads/2011/05/dbml-presentation-of-sql-database-table.png)

**Creating The model:**

In the model directory, create a new model named ‘QuestionModel’. This will contains all our model implementation code block. Create and interface named ‘IQuestionService’ , which will be base and define the methods to be implemented.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | public interface IQuestionService          {              bool ValidateQuestion(Question poll);              bool CreateQuestion(Question poll);              List<Question> GetAllQuestions();              Question GetQuestion(int id);              bool DeleteQuestion(int id);              bool SaveQuestion(Question poll);          } |

Now create a new class named ‘QuestionService’ which implement the interface ‘IQuestionService’. Besides, this class should contain the declaration of an object of ‘MyDataContext’ type(according to dbml file). Also, we can implement the first part, which list all existing questions that means the ‘GetAllQuestions’ method. So, the constructor and this method should look like as follows:

|  |  |
| --- | --- |
| 01  02  03  04  05  06  07  08  09  10 | private MPDataContext qDB;    public QuestionService()  {       qDB = new MPDataContext();  }  public List<Question> GetAllQuestions()  {       return qDB.Questions.ToList();  } |

**Create New Page And Controller:**

First create a new mvc application and select razor as the template engine as i described on my previous basic mvc 3 tutorial  
Lets create a new page and its controller, that we will be using through out the tutorial. Open ‘Views’=>’Shared’=>’\_Layout.chtml’ and add a new menu link Like as follows:

|  |  |
| --- | --- |
| 1  2  3  4  5 | <ul id="menu">         <li>@Html.ActionLink("Home", "Index", "Home")</li>         <li>@Html.ActionLink("About", "About", "Home")</li>         <li>@Html.ActionLink("Questions", "Index", "Question")</li>  </ul> |

Notice the structure @Html.ActionLink(“Questions”, “Index”, “Question”) . First parameter is the text of the anchor link, second is the method name that will be called for the associated controller and third one the name of the controller. Save it. And create a new controller named “QuestionController.cs” and check the option for automatically creating action methods for create/update/delete.

Now, Create an instance of our model class here in the constructor and implement the ‘index’ method to send the question lists to view.

|  |  |
| --- | --- |
| 01  02  03  04  05  06  07  08  09  10  11  12  13  14  15  16  17  18  19  20  21  22  23 | private QuestionModels.IQuestionService qService;    public QuestionController()  {        qService = new QuestionModels.QuestionService();  }         //      // GET: /Question/        public ActionResult Index()      {          List<Question> questions;          try          {              questions = qService.GetAllQuestions();          }          catch          {              questions = new List<Question>();          }          return View(questions);      } |

**Creating The View:**

Lets create the view for this index method. Instead of creating it manually, we can create it easily just by give some commands(I have already explain how to do it in my previous basic mvc 3 tutorial). Don’t forget to select the ‘Scaffold Template’ as ‘List’ while creating the view.Now please build the application and run it. Navigate to the ‘Questions’ Menu. You should See something like as follows:

[](http://cdn.codesamplez.com/wp-content/uploads/2011/05/new-contoller-index-page1.jpg)

As We have no data in database yet, we get this empty list. However, ‘create’ option won’t work also as we haven’t implement the controller method for that yet.

**Implement Create Method:**

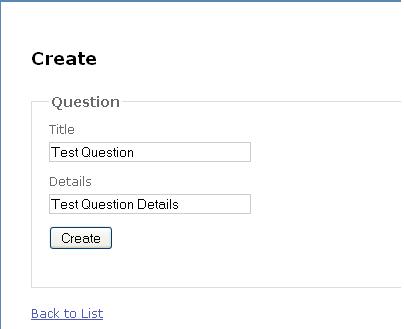
You will see, among the automatically generated methods, there are two methods named ‘Create’, one is with ‘ [HttpPost]’ and parameter another is without attribute and parameter. Without one will be called when we show the create template to user. when user fills up the form and submits, with the ‘[HtttpPost]’ one will be called. withe the corresponding ‘Question’ object as parameter.

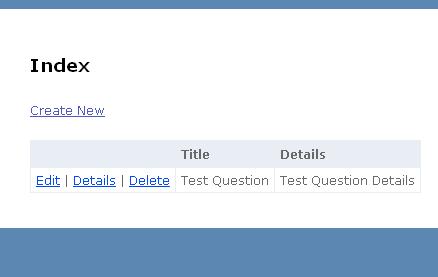
|  |  |
| --- | --- |
| 01  02  03  04  05  06  07  08  09  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26 | //          // GET: /Question/Create            public ActionResult Create()          {              //Question question = new Question();              return View();          }            //          // POST: /Question/Create            [HttpPost]          public ActionResult Create(Question question)          {              try              {                  // TODO: Add insert logic here                  qService.CreateQuestion(question);                  return RedirectToAction("Index");              }              catch              {                  return View();              }          } |

In the model section, now we need to implement the ‘CreateQuestion’ method as follows, of course using Linq

|  |  |
| --- | --- |
| 01  02  03  04  05  06  07  08  09  10  11  12  13 | public bool CreateQuestion(Question tQestion)              {                  try                  {                      qDB.Questions.InsertOnSubmit(tQestion);                      qDB.SubmitChanges();                      return true;                  }                  catch                  {                      return false;                  }              } |

OK, now lets rebuild and run the the application again. We should be able to create question now which will be saved to database and then we should see this in the main list.

[](http://cdn.codesamplez.com/wp-content/uploads/2011/05/create-new-page.jpg)

[](http://cdn.codesamplez.com/wp-content/uploads/2011/05/list-page.jpg)

**Create Details Page:**

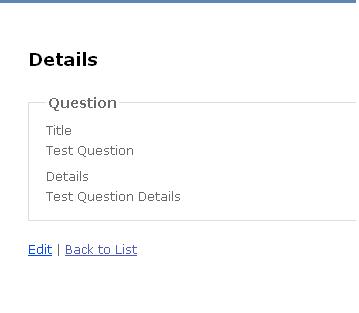
Details page are called with the id or the record as parameter. We can implement the controller in the following way:

|  |  |
| --- | --- |
| 01  02  03  04  05  06  07  08  09  10  11  12  13  14  15  16 | //          // GET: /Question/Details/5            public ActionResult Details(int id)          {              Question question;              try              {                  question = qService.GetQuestion(id);              }              catch              {                  question = new Question();              }              return View(question);          } |

Also, implement the model’s method for return the record details in linq to sql, which is very simple to implement  :

|  |  |
| --- | --- |
| 1  2  3  4 | public Question GetQuestion(int id)  {      return qDB.Questions.Single(q => q.Id == id);  } |

Besides, we will also have to create the view with scaffold template as ‘Details’. And now we should be done to see the details page:

[](http://cdn.codesamplez.com/wp-content/uploads/2011/05/details-page.jpg)

**Edit Page:**

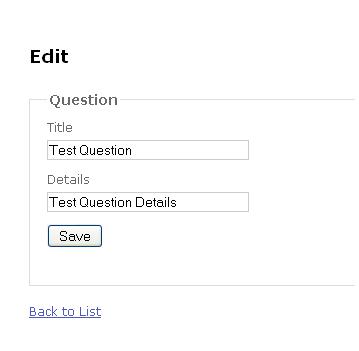
Similar to create page, there are 2 methods for edit option also, one is with ‘HttpPost’ attributes. Here is the controller code for this option:

|  |  |
| --- | --- |
| 01  02  03  04  05  06  07  08  09  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36 | //          // GET: /Question/Edit/5            public ActionResult Edit(int id)          {                Question question;              try              {                  question = qService.GetQuestion(id);              }              catch              {                  question = new Question();              }              return View(question);          }            //          // POST: /Question/Edit/5            [HttpPost]          public ActionResult Edit(int id, Question question)          {              try              {                  // TODO: Add update logic here                  question.Id = id;                  qService.SaveQuestion(question);                  return RedirectToAction("Index");              }              catch              {                  return View();              }          } |

Also, implement the ‘SaveQuestion’ method on the model section which will retrieve the corresponding record, change it and then save it :

|  |  |
| --- | --- |
| 01  02  03  04  05  06  07  08  09  10  11  12  13  14  15 | public bool SaveQuestion(Question tQestion)              {                  try                  {                      Question question = qDB.Questions.Single(q => q.Id == tQestion.Id);                      question.Details = tQestion.Details;                      question.Title = tQestion.Title;                      qDB.SubmitChanges();                      return true;                  }                  catch                  {                      return false;                  }              } |

So, we should be able to edit and save a record from now on:

[](http://cdn.codesamplez.com/wp-content/uploads/2011/05/edit-record.jpg)

**Implement ‘Delete’ Functionality:**

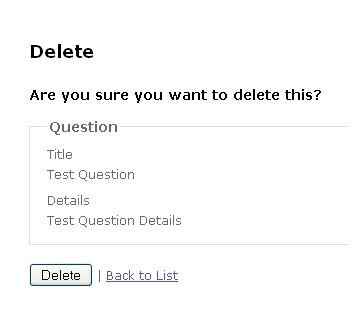
Delete option also include 2 methods. It have a basic scaffold template also, which simply ask for confirmation from the user whether he really want’s to delete or not. Here is the code sample for the controller part of delete action:

|  |  |
| --- | --- |
| 01  02  03  04  05  06  07  08  09  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34 | //          // GET: /Question/Delete/5            public ActionResult Delete(int id)          {              Question question;              try              {                  question = qService.GetQuestion(id);              }              catch              {                  question = new Question();              }              return View(question);          }            //          // POST: /Question/Delete/5            [HttpPost]          public ActionResult Delete(int id, FormCollection collection)          {              try              {                  // TODO: Add delete logic here                  qService.DeleteQuestion(id);                  return RedirectToAction("Index");              }              catch              {                  return View();              }          } |

Also, we need to implement the model class’s ‘DeleteQuestion’ method on models section using linq to sql as follows:

|  |  |
| --- | --- |
| 01  02  03  04  05  06  07  08  09  10  11  12  13  14 | public bool DeleteQuestion(int id)              {                  try                  {                      Question question = qDB.Questions.Single(q => q.Id == id);                      qDB.Questions.DeleteOnSubmit(question);                      qDB.SubmitChanges();                      return true;                  }                  catch                  {                      return false;                  }              } |

So, if we rebuild and run the application again. We should be able to delete now:

[](http://cdn.codesamplez.com/wp-content/uploads/2011/05/delete-record.jpg)

**Adding Validation On Forms:**

Till now, we haven’t added any kind of validation to our application, neither client nor server. Asp.NET MVC 3 comes with a very nice feature of binding a model class to be validated nicely in both client and server end. We don’t have to even write any java-script code for client validation, they will be automatically generated. To add validations to the entity classes in dbml files, open the dbml in design mode, right click and select ‘view code’ option. On the code file, add the following code snippet:

|  |  |
| --- | --- |
| 01  02  03  04  05  06  07  08  09  10  11  12  13 | [MetadataType(typeof(QuestionValidation))]   public partial class Question  {  }     [Bind(Exclude="Id")]   public class QuestionValidation   {       [Required(ErrorMessage="Title Required")]       public string Title { get; set; }       [Required(ErrorMessage = "Description Required")]       public string Details { get; set; }   } |

**Complete Code Reference:**

To make it more easier, I am going to share the both model and controller codes together here. As we didn’t had to touch the view part at all(all were generated automatically except the layout part that i have already given above), I am not going put their code here.

**Controller :**

|  |  |
| --- | --- |
| 001  002  003  004  005  006  007  008  009  010  011  012  013  014  015  016  017  018  019  020  021  022  023  024  025  026  027  028  029  030  031  032  033  034  035  036  037  038  039  040  041  042  043  044  045  046  047  048  049  050  051  052  053  054  055  056  057  058  059  060  061  062  063  064  065  066  067  068  069  070  071  072  073  074  075  076  077  078  079  080  081  082  083  084  085  086  087  088  089  090  091  092  093  094  095  096  097  098  099  100  101  102  103  104  105  106  107  108  109  110  111  112  113  114  115  116  117  118  119  120  121  122  123  124  125  126  127  128  129  130  131  132  133  134  135  136  137  138  139  140  141  142  143  144  145  146  147  148  149  150  151  152  153 | using System;  using System.Collections.Generic;  using System.Linq;  using System.Web;  using System.Web.Mvc;  using MyMvcApplication.Models;    namespace MyMvcApplication.Controllers  {      public class QuestionController : Controller      {            private QuestionModels.IQuestionService qService;            public QuestionController()          {              qService = new QuestionModels.QuestionService();          }            //          // GET: /Question/            public ActionResult Index()          {              List<Question> questions;              try              {                  questions = qService.GetAllQuestions();              }              catch              {                  questions = new List<Question>();              }              return View(questions);          }            //          // GET: /Question/Details/5            public ActionResult Details(int id)          {              Question question;              try              {                  question = qService.GetQuestion(id);              }              catch              {                  question = new Question();              }              return View(question);          }            //          // GET: /Question/Create            public ActionResult Create()          {              //Question question = new Question();              return View();          }            //          // POST: /Question/Create            [HttpPost]          public ActionResult Create(Question question)          {              try              {                  // TODO: Add insert logic here                  qService.CreateQuestion(question);                  return RedirectToAction("Index");              }              catch              {                  return View();              }          }            //          // GET: /Question/Edit/5            public ActionResult Edit(int id)          {                Question question;              try              {                  question = qService.GetQuestion(id);              }              catch              {                  question = new Question();              }              return View(question);          }            //          // POST: /Question/Edit/5            [HttpPost]          public ActionResult Edit(int id, Question question)          {              try              {                  // TODO: Add update logic here                  question.Id = id;                  qService.SaveQuestion(question);                  return RedirectToAction("Index");              }              catch              {                  return View();              }          }            //          // GET: /Question/Delete/5            public ActionResult Delete(int id)          {              Question question;              try              {                  question = qService.GetQuestion(id);              }              catch              {                  question = new Question();              }              return View(question);          }            //          // POST: /Question/Delete/5            [HttpPost]          public ActionResult Delete(int id, FormCollection collection)          {              try              {                  // TODO: Add delete logic here                  qService.DeleteQuestion(id);                  return RedirectToAction("Index");              }              catch              {                  return View();              }          }      }  } |

**Model :**

|  |  |
| --- | --- |
| 001  002  003  004  005  006  007  008  009  010  011  012  013  014  015  016  017  018  019  020  021  022  023  024  025  026  027  028  029  030  031  032  033  034  035  036  037  038  039  040  041  042  043  044  045  046  047  048  049  050  051  052  053  054  055  056  057  058  059  060  061  062  063  064  065  066  067  068  069  070  071  072  073  074  075  076  077  078  079  080  081  082  083  084  085  086  087  088  089  090  091  092  093  094  095  096  097  098  099  100  101  102  103  104  105  106  107  108  109  110  111  112  113  114  115  116  117  118  119  120  121 | using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Web.Mvc;  using System.ComponentModel.DataAnnotations;    namespace MyMvcApplication.Models  {        public class QuestionModels      {            #region Services          public interface IQuestionService          {              bool ValidateQuestion(Question poll);              bool CreateQuestion(Question poll);              List<Question> GetAllQuestions();              Question GetQuestion(int id);              bool DeleteQuestion(int id);              bool SaveQuestion(Question poll);          }            public class QuestionService:IQuestionService          {                private MPDataContext qDB;                public QuestionService()              {                  qDB = new MPDataContext();              }                #region IQuestionService Members                public bool ValidateQuestion(Question tQuestion)              {                  if (tQuestion.Title.Length <= 0)                  {                      return false;                  }                  if (tQuestion.Details.Length <= 0)                  {                      return false;                  }                  return true;              }                public bool CreateQuestion(Question tQestion)              {                  try                  {                      qDB.Questions.InsertOnSubmit(tQestion);                      qDB.SubmitChanges();                      return true;                  }                  catch                  {                      return false;                  }              }                public List<Question> GetAllQuestions()              {                  return qDB.Questions.ToList();              }                public Question GetQuestion(int id)              {                  return qDB.Questions.Single(q => q.Id == id);              }                public bool DeleteQuestion(int id)              {                  try                  {                      Question question = qDB.Questions.Single(q => q.Id == id);                      qDB.Questions.DeleteOnSubmit(question);                      qDB.SubmitChanges();                      return true;                  }                  catch                  {                      return false;                  }              }                public bool SaveQuestion(Question tQestion)              {                  try                  {                      Question question = qDB.Questions.Single(q => q.Id == tQestion.Id);                      question.Details = tQestion.Details;                      question.Title = tQestion.Title;                      qDB.SubmitChanges();                      return true;                  }                  catch                  {                      return false;                  }              }                #endregion          }            #endregion            #region ValidationClasses          [Bind(Exclude="Id")]          public class QuestionValidation          {              [Required(ErrorMessage="Title Required")]              public string Title { get; set; }              [Required(ErrorMessage = "Description Required")]              public string Details { get; set; }          }          #endregion      }  } |

Hope this tutorial on asp.net mvc3 and linq with razor template easier is helpful to you and now easy for you to do as your home practice  . Let me know if anything isn’t clear enough by commenting below. Happy coding