



ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

VNUHCM - UIT

LẬP TRÌNH WEB

CHƯƠNG 4 MODEL TRONG ASP.NET MVC

NỘI DUNG

1

- Kiến trúc MVC

2

- ASP.NET MVC - Model

3

- Tích hợp Controller, View và Model

4

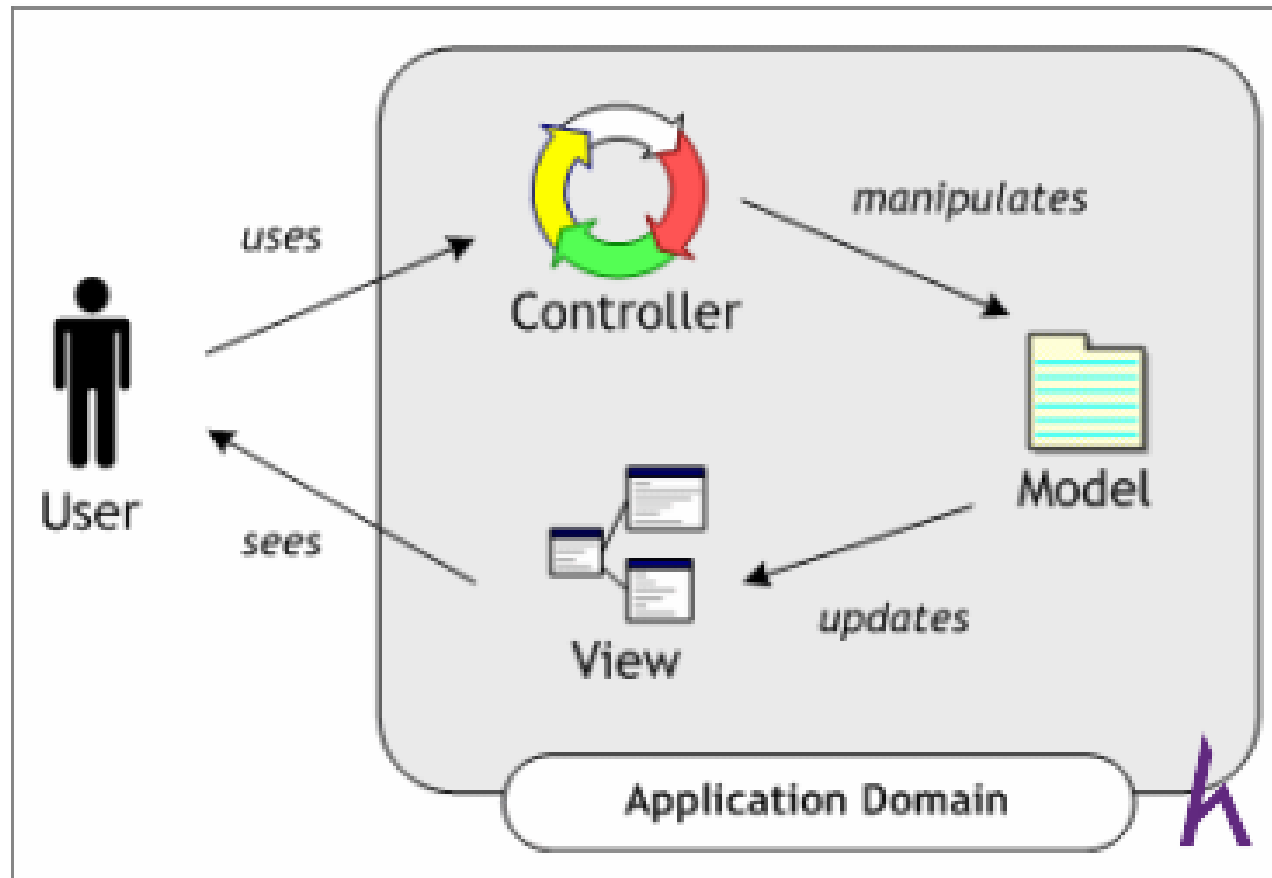
- Model Binding

5

- Tạo View Edit

1. KIẾN TRÚC MVC

❖ MVC (Model – View - Controller)

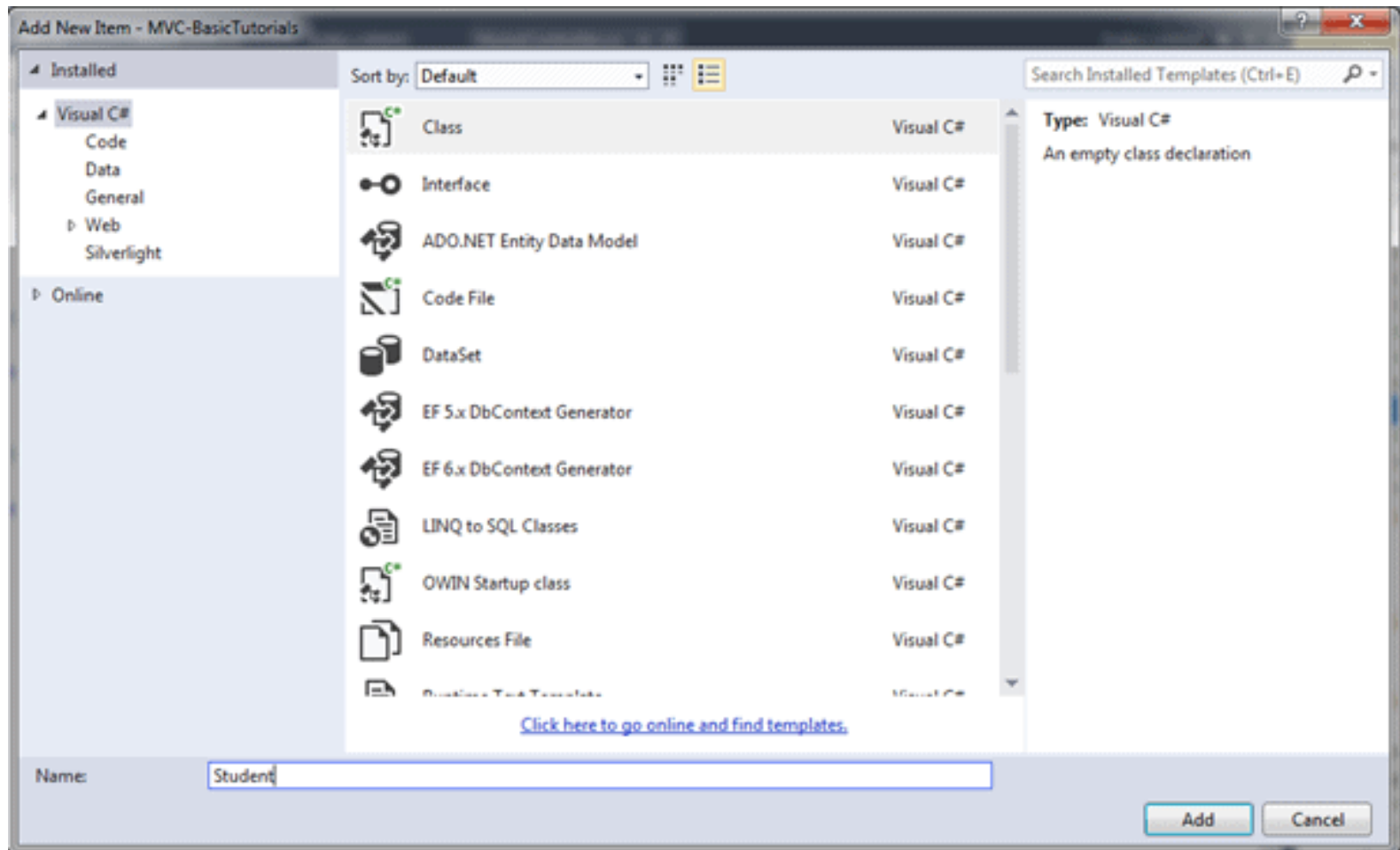


2. ASP.NET MVC - MODEL

- ❖ **Models** trong các ứng dụng dựa trên MVC là những thành phần có nhiệm vụ lưu trữ thông tin, trạng thái của các đối tượng, thông thường nó là một lớp được ánh xạ từ một bảng trong CSDL.
- ❖ Lấy ví dụ, chúng ta có lớp Product được sử dụng để mô tả dữ liệu từ bảng Products trong SQL, bao gồm ProductID, OrderDate,...

2. ASP.NET MVC - MODEL

❖ Cách thêm một Model trong ASP.NET MVC:



2. ASP.NET MVC - MODEL

❖ Ví dụ: Lớp Model

```
namespace MVC_BasicTutorials.Models
{
    public class Student
    {
        public int StudentId { get; set; }
        public string StudentName { get; set; }
        public int Age { get; set; }
    }
}
```

3. TÍCH HỢP CONTROLLER, VIEW VÀ MODEL

- ❖ Chúng ta đã tạo **StudentController**, **Model** và **View** trong các phần trước, nhưng chúng ta chưa tích hợp tất cả các thành phần này để chạy nó.
- ❖ Đoạn mã sau đây của **StudentController**, **Model** và **View** trong lớp **Student** được tạo trong các phần trước.

3. TÍCH HỢP CONTROLLER, VIEW VÀ MODEL

❖ StudentController:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using MVC_BasicTutorials.Models;

namespace MVC_BasicTutorials.Controllers
{
    public class StudentController : Controller
    {
        // GET: Student
        public ActionResult Index()
        {
            return View();
        }
    }
}
```


3. TÍCH HỢP CONTROLLER, VIEW VÀ MODEL

❖ Lớp Model Student:

```
namespace MVC_BasicTutorials.Models
{
    public class Student
    {
        public int StudentId { get; set; }
        public string StudentName { get; set; }
        public int Age { get; set; }
    }
}
```

3. TÍCH HỢP CONTROLLER, VIEW VÀ MODEL

❖ Tập tin `Index.cshtml` để hiển thị danh sách sinh viên

```
@model IEnumerable<MVC_BasicTutorials.Models.Student>

@{
    ViewBag.Title = "Index";
    Layout = "~/Views/Shared/_Layout.cshtml";
}

<h2>Index</h2>

<p>
    @Html.ActionLink("Create New", "Create")
</p>
<table class="table">
    <tr>
        <th>
            @Html.DisplayNameFor(model => model.StudentName)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.Age)
        </th>
        <th></th>
    </tr>
```

3. TÍCH HỢP CONTROLLER, VIEW VÀ MODEL

❖ Tập tin `Index.cshtml` để hiển thị danh sách sinh viên

```
@foreach (var item in Model) {  
    <tr>  
        <td>  
            @Html.DisplayFor(modelItem => item.StudentName)  
        </td>  
        <td>  
            @Html.DisplayFor(modelItem => item.Age)  
        </td>  
        <td>  
            @Html.ActionLink("Edit", "Edit", new { id=item.StudentId }) |  
            @Html.ActionLink("Details", "Details", new { id=item.StudentId }) |  
            @Html.ActionLink("Delete", "Delete", new { id = item.StudentId })  
        </td>  
    </tr>  
}
```

3. TÍCH HỢP CONTROLLER, VIEW VÀ MODEL

❖ Truyền dữ liệu trong Model từ Controller sang View

```
public class StudentController : Controller
{
    // GET: Student
    public ActionResult Index()
    {
        var studentList = new List<Student>{
            new Student() { StudentId = 1, StudentName = "John", Age = 18 } ,
            new Student() { StudentId = 2, StudentName = "Steve", Age = 21 } ,
            new Student() { StudentId = 3, StudentName = "Bill", Age = 25 } ,
            new Student() { StudentId = 4, StudentName = "Ram" , Age = 20 } ,
            new Student() { StudentId = 5, StudentName = "Ron" , Age = 31 } ,
            new Student() { StudentId = 4, StudentName = "Chris" , Age = 17 } ,
            new Student() { StudentId = 4, StudentName = "Rob" , Age = 19 }

        };

        // Get the students from the database in the real application

        return View(studentList);
    }
}
```

3. TÍCH HỢP CONTROLLER, VIEW VÀ MODEL

❖ Chạy project MVC cho kết quả như sau:

Application name Home About Contact

Index

Create New

Name	Age	
John	18	Edit Details Delete
Steve	21	Edit Details Delete
Bill	25	Edit Details Delete
Ram	20	Edit Details Delete
Ron	31	Edit Details Delete
Chris	17	Edit Details Delete
Rob	19	Edit Details Delete

© 2014 - My ASP.NET Application

4. MODEL BINDING

- ❖ Để hiểu ràng buộc mô hình trong MVC, trước tiên hãy xem cách bạn có thể nhận các yêu cầu từ http trong phương thức hành động bằng cách sử dụng kiểu ASP.NET truyền thống.
- ❖ Hình sau đây cho thấy cách bạn có thể nhận các giá trị từ yêu cầu **HttpGet** và **HttpPost** bằng cách sử dụng trực tiếp đối tượng **Request** trong phương thức hành động.

4. MODEL BINDING

HttpGet
/Student/Edit?id=1

```
public ActionResult Edit()
{
    var id = Request.QueryString["id"];

    // retrieve data from the database

    return View();
} © TutorialsTeacher.com
```

HttpPost
/Student/Edit

```
[HttpPost]
public ActionResult Edit()
{
    var id = Request["StudentId"];
    var name = Request["StudentName"];
    var age = Request["Age"];

    //update database here..

    return RedirectToAction("Index");
}
```

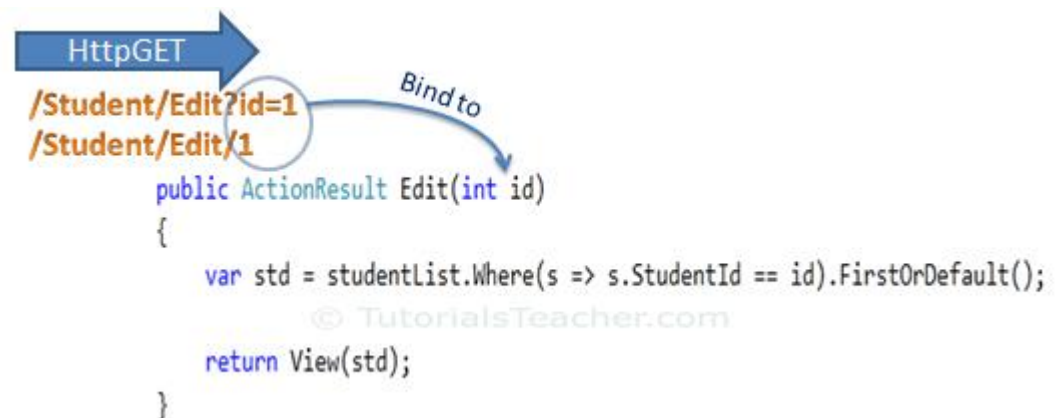
4. MODEL BINDING

- ❖ Với liên kết mô hình, framework MVC chuyển đổi các giá trị yêu cầu từ http (từ chuỗi truy vấn hoặc form) thành các tham số phương thức hành động. Các tham số này có thể là kiểu **nguyên thủy** hoặc kiểu **phức tạp**.

4. MODEL BINDING

❖ Liên kết với kiểu nguyên thủy:

- Các httpGET nhúng dữ liệu vào chuỗi truy vấn. Framework MVC tự động chuyển đổi một chuỗi truy vấn thành các tham số của phương thức hành động.
- Ví dụ: chuỗi truy vấn "id" trong yêu cầu GET sau đây sẽ tự động được ánh xạ tới tham số id của phương thức hành động Edit ().



4. MODEL BINDING

❖ Liên kết với kiểu nguyên thủy:

- Ví dụ <http://localhost/Student/Edit?id=1&name=John> sẽ ánh xạ tới id và tham số tên của phương thức hành động Edit sửa sau:

```
public ActionResult Edit(int id, string name)
{
    // do something here

    return View();
}
```

4. MODEL BINDING

❖ Liên kết với kiểu phức tạp:

- Model cũng hoạt động trên các kiểu phức tạp.
- Liên kết Model trong Framework MVC tự động chuyển đổi dữ liệu trường form của đối tượng HttpPOST thành các thuộc tính của một tham số kiểu phức tạp.

4. MODEL BINDING

❖ Liên kết với kiểu phức tạp:

```
public class Student
{
    public int StudentId { get; set; }
    [Display(Name="Name")]
    public string StudentName { get; set; }
    public int Age { get; set; }
    public Standard standard { get; set; }
}

public class Standard
{
    public int StandardId { get; set; }
    public string StandardName { get; set; }
}
```

4. MODEL BINDING

❖ Liên kết với kiểu phức tạp:

- Bây giờ, chúng ta có thể tạo một phương thức hành động có các tham số là kiểu Student. Trong ví dụ sau, phương thức Edit (HttpPost) có tham số là kiểu Student.

```
[HttpPost]
public ActionResult Edit(Student std)
{
    var id = std.StudentId;
    var name = std.StudentName;
    var age = std.Age;
    var standardName = std.standard.StandardName;

    //update database here..

    return RedirectToAction("Index");
}
```

4. MODEL BINDING

❖ Liên kết với kiểu phức tạp:

Edit
Student

Name

StandardName

Age

HttpPOST

```
[HttpPost]
public ActionResult Edit(Student std)
{
    var id = std.StudentId;
    var name = std.StudentName;
    var age = std.Age;
    var standardName = std.standard.StandardName;
    //update database here..
    return RedirectToAction("Index");
}
```

Edit

Name

Age

HttpPOST

```
[HttpPost]
public ActionResult Edit(FormCollection values)
{
    var name = values["StudentName"];
    var age = values["Age"];
    //write code to update student
    return RedirectToAction("Index");
}
```

4. MODEL BINDING

❖ Bind Attribute:

- ASP.NET MVC cũng cho phép bạn chỉ định các thuộc tính nào của lớp Model mà bạn muốn liên kết.
- Trong ví dụ sau, phương thức hành động Edit sẽ chỉ liên kết thuộc tính **StudentId** và **StudentName** của Model **Student**.

```
[HttpPost]
public ActionResult Edit([Bind(Include = "StudentId, StudentName")] Student std)
{
    var name = std.StudentName;

    //write code to update student

    return RedirectToAction("Index");
}
```

4. MODEL BINDING

❖ Bind Attribute:

- Chúng ta cũng có thể sử dụng các thuộc tính loại trừ như dưới đây.

```
HttpPost]
public ActionResult Edit([Bind(Exclude = "Age")] Student std)
{
    var name = std.StudentName;

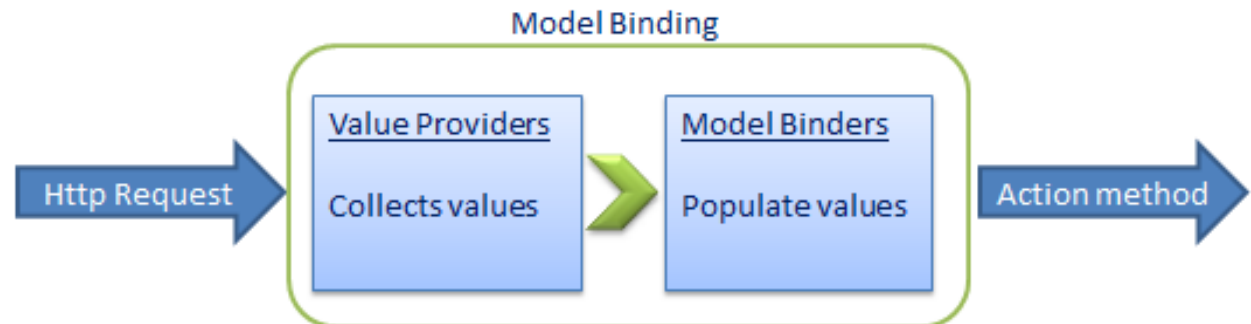
    //write code to update student

    return RedirectToAction("Index");
}
```


4. MODEL BINDING

❖ Liên kết mô hình bên trong:

- Như bạn đã thấy rằng ràng buộc Model tự động chuyển đổi các giá trị yêu cầu thành một đối tượng kiểu nguyên thủy hoặc phức tạp. Model ràng buộc là một quá trình hai bước. Đầu tiên, nó thu thập các giá trị từ yêu cầu http đến và sau đó điền vào kiểu nguyên thủy hoặc kiểu phức tạp với các giá trị này.



4. MODEL BINDING

❖ Các giá trị mặc định đánh giá từ các nguồn sau:

- Các tham số hành động bị ràng buộc trước đó, khi hành động là hành động con
- Các trường Form (`Request.Form`)
- Các giá trị thuộc tính trong JSON (`Request.InputStream`), nhưng chỉ khi yêu cầu là một AJAX
- Dữ liệu Route (`RouteData.Values`)
- Các tham số Querystring (`Request.QueryString`)
- Posted files (`Request.Files`)

5. TẠO VIEW EDIT

Me [minimize] [maximize] [close]

Edit - My ASP.NET Applic x

← → ↻ ⓘ localhost:54710/student/Edit/1 ☆ ⋮

Application name Home About Contact

Edit

Student

StudentName

Age

Save

[Back to List](#)

© 2018 - My ASP.NET Application

5. TẠO VIEW EDIT

Index

Create New

Name	Age	
John	18	Edit Details Delete
Steve	21	Edit Details Delete
Bill	25	Edit Details Delete
Ram	20	Edit Details Delete
Ron	31	Edit Details Delete
Chris	17	Edit Details Delete
Rob	19	Edit Details Delete

1 <http://localhost/Edit/1>

HttpGET

2

```
public ActionResult Edit(int Id)
{
    var std = students.Where(s => s.StudentId == Id).FirstOrDefault();

    return View(std);
}
```

Renders

Edit

Student

3

[HttpPost]

4

```
public ActionResult Edit(Student std)
{
    var name = std.StudentName;
    var age = std.Age;
    //write code to update student

    return RedirectToAction("Index");
}
```

HttpPOST

<http://localhost/Edit>

Name

John

Age

18

Save

5. TẠO VIEW EDIT

- ❖ Bây giờ, chúng ta tạo một phương thức **Edit** trong lớp **StudentController**. Trang Index hiển thị dữ liệu như ở trên sẽ gửi tham số **StudentId** đến phương thức **Edit** khi vào liên kết **Edit**.
- ❖ Phương thức **HttpGet Edit()** phải thực hiện hai tác vụ, trước tiên, nó sẽ tìm nạp thông tin sinh viên từ nguồn dữ liệu, có **StudentId** khớp với **StudentId** trong chuỗi truy vấn. Thứ hai, nó sẽ hiển thị chế độ xem **Edit** với thông tin sinh viên để người dùng có thể cập nhật.

5. TẠO VIEW EDIT

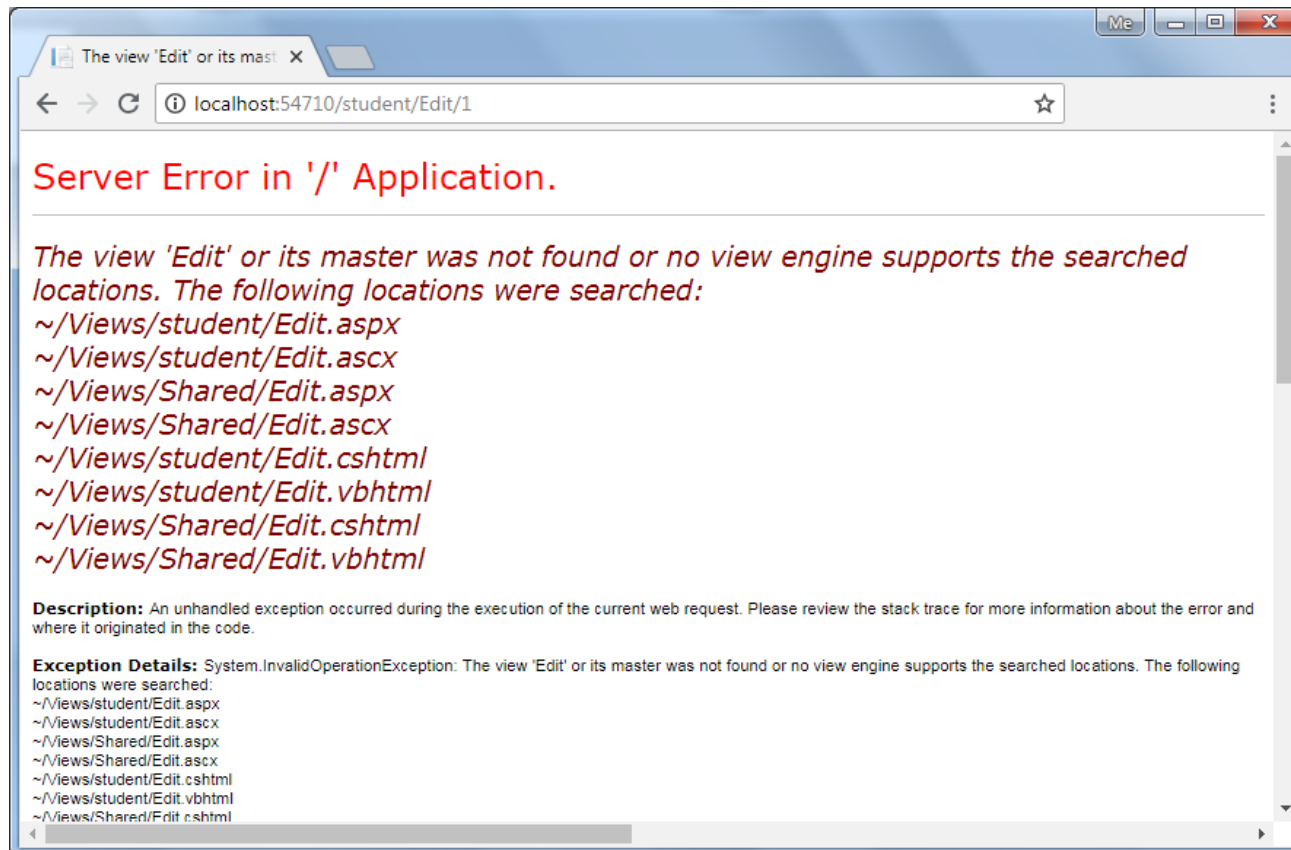
```
public class StudentController : Controller
{
    IList<Student> studentList = new List<Student>() {
        new Student(){ StudentId=1, StudentName="John", Age = 18 },
        new Student(){ StudentId=2, StudentName="Steve", Age = 21 },
        new Student(){ StudentId=3, StudentName="Bill", Age = 25 },
        new Student(){ StudentId=4, StudentName="Ram", Age = 20 },
        new Student(){ StudentId=5, StudentName="Ron", Age = 31 },
        new Student(){ StudentId=6, StudentName="Chris", Age = 17 },
        new Student(){ StudentId=7, StudentName="Rob", Age = 19 }
    };

    public ActionResult Edit(int Id)
    {
        //Get the student from studentList sample collection for demo purpose.
        //Get the student from the database in the real application
        var std = studentList.Where(s => s.StudentId == Id).FirstOrDefault();

        return View(std);
    }
}
```

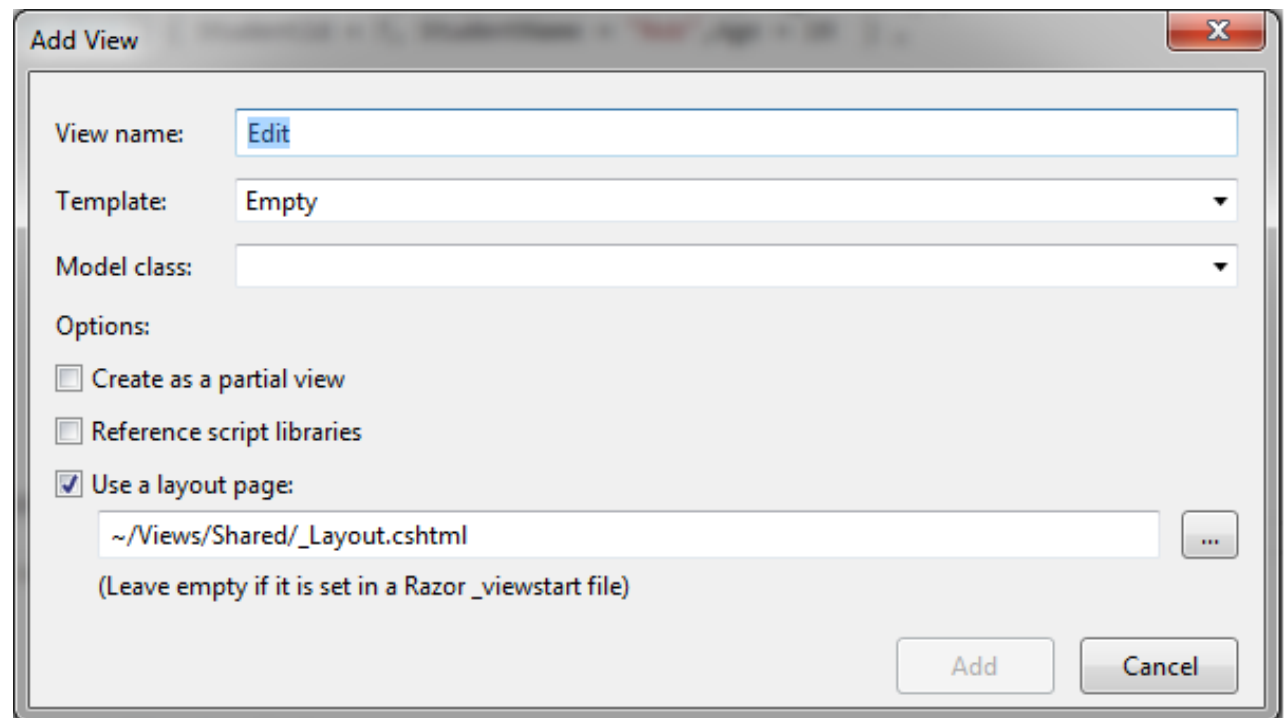
5. TẠO VIEW EDIT

- ❖ Bây giờ, nếu bạn nhấp vào liên kết Edit từ chế độ xem Chỉ mục thì bạn sẽ gặp lỗi sau:



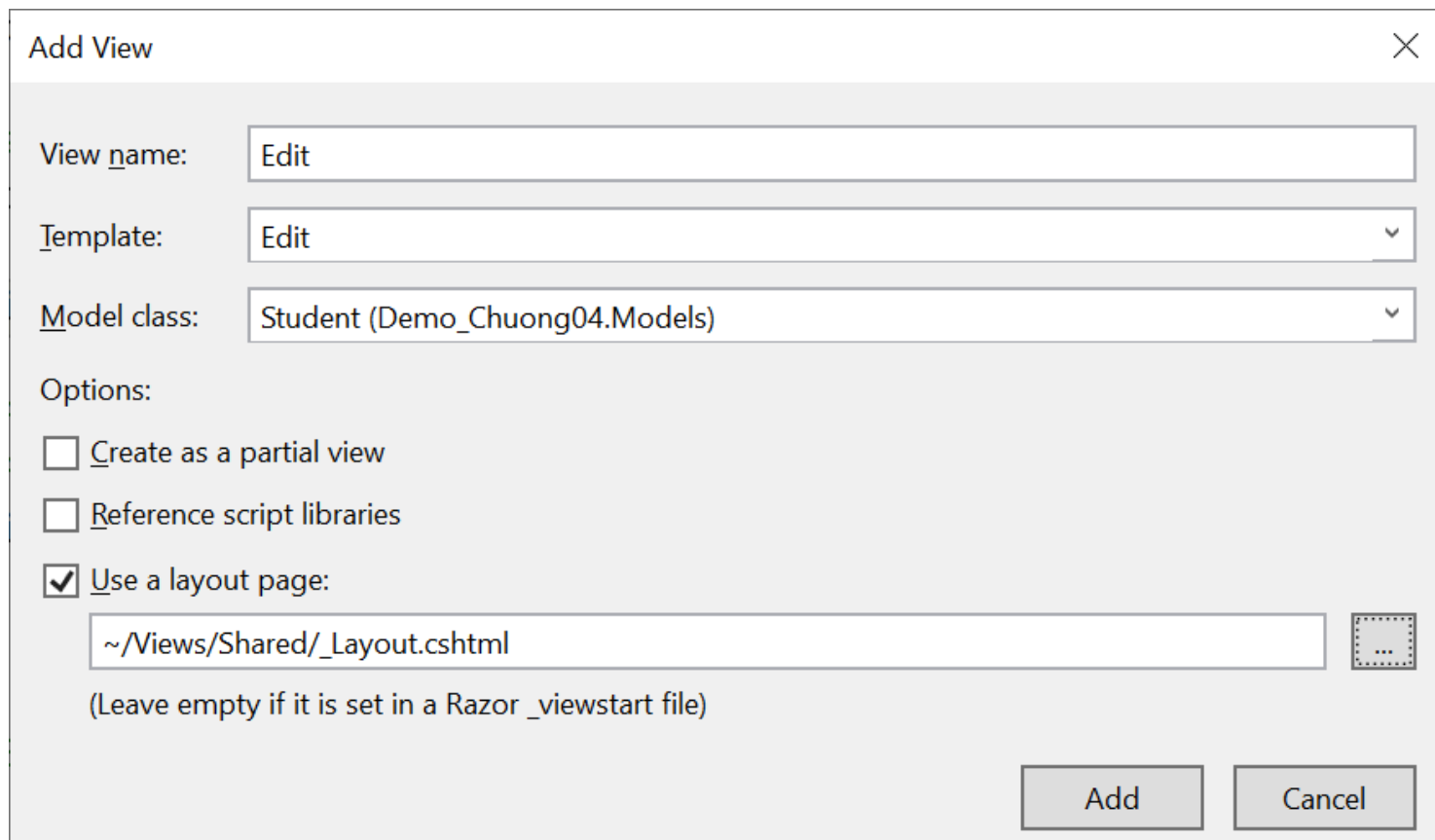
5. TẠO VIEW EDIT

- ❖ Để tạo View, nhấp chuột phải vào bên trong phương thức Edit và nhấp vào Add View... Nó sẽ mở đối thoại Add View.



5. TẠO VIEW EDIT

❖ Chọn Edit trong Template và chọn Student trong Model Class



Add View

View name: Edit

Template: Edit

Model class: Student (Demo_Chuong04.Models)

Options:

☐ Create as a partial view

☐ Reference script libraries

☒ Use a layout page:

~/Views/Shared/_Layout.cshtml

(Leave empty if it is set in a Razor _viewstart file)

Add Cancel

5. TẠO VIEW EDIT

- ❖ Sau đó, nhấp vào Add để tạo view **Edit.cshtml** trong thư mục **View/Student**.
- ❖ Code phát sinh tự động như ở slide sau.

5. TẠO VIEW EDIT

Edit.cshtml* StudentController.cs

```
1  @model Demo_Chuong04.Models.Student
2  @{
3      ViewBag.Title = "Edit";
4      Layout = "~/Views/Shared/_Layout.cshtml";
5  }
6  <h2>Edit</h2>
7  @using (Html.BeginForm())
8  {
9      @Html.AntiForgeryToken()
10     <div class="form-horizontal">
11         <h4>Student</h4>
12         <hr />
13         @Html.ValidationSummary(true, "", new { @class = "text-danger" })
14         @Html.HiddenFor(model => model.StudentId)
15         <div class="form-group">
16             @Html.LabelFor(model => model.StudentName, htmlAttributes: new { @class = "control-label col-md-2" })
17             <div class="col-md-10">
18                 @Html.EditorFor(model => model.StudentName, new { htmlAttributes = new { @class = "form-control" } })
19                 @Html.ValidationMessageFor(model => model.StudentName, "", new { @class = "text-danger" })
20             </div>
21         </div>
22         <div class="form-group">
23             @Html.LabelFor(model => model.Age, htmlAttributes: new { @class = "control-label col-md-2" })
24             <div class="col-md-10">
25                 @Html.EditorFor(model => model.Age, new { htmlAttributes = new { @class = "form-control" } })
26                 @Html.ValidationMessageFor(model => model.Age, "", new { @class = "text-danger" })
27             </div>
28         </div>
29         <div class="form-group">
30             <div class="col-md-offset-2 col-md-10">
31                 <input type="submit" value="Save" class="btn btn-default" />
32             </div>
33         </div>
34     </div>
35 }
36 <div>
37     @Html.ActionLink("Back to List", "Index")
38 </div>
```

5. TẠO VIEW EDIT

Edit - My ASP.NET Application

+

localhost:59854/Student/Edit/2

Anh Dung

Diep Nhu

Thẻ mới

Download Ebook Bấ...

Bí Quyết Giàu Có V...

Download and Inst...

Download and Inst...

Application name

Home

About

Contact

Edit

Student

StudentName

Steve

Age

21

Save

Back to List

© 2023 - My ASP.NET Application

