MVC [MODEL VIEW CONTROLLER]

Complete Name : MVC ASP.NET.

∵ MVC ≠ MVC ASP.NET.

Design Pattern. Web development Env.

MVC → ASP.NET
MVC → PHP

Web Application: Executed On a Web Server & accessed from Web browser. [it could be desktop Application with Global data Base]

divided into three layers

- Presentation Layer → User End
- Business Layer → Manipulation / code End
- Data Layer → Data Storage End

ARCHITECTURE OF WEB APP.

→ Single tier. All three layers on single end.
→ Two tier. Presentation / Business on end. Presentation other
→ Three tier.

Presentation	Bussiness	Data
↓	↓	↓
Browser	ASP.NET (development tool)	SQL (data storage

Now let's talk about Webpages.

Web Pages → Static webpages.
→ Dynamic webpages.

• To develop Dynamic web Application we will use MVC, ASP.NET.

→ Backend Language : C#
→ Presentation Language: Html.
→ Editor: VS (Visual Studio).

CONTROLLER.

- It's a C# file (.cs extension).
- Inherited from Controller class. (present in System.Web.MVC)
- Uses Sufix (Controller).

Action Methods.

Access Modifier return type.

```
Public  ActionResult  Index()
{
    return View();
}
```

- Must be public
- Cannot be Static.

VIEW

- cshtml file (cs + html).
- Used to design front End (UI).
- This is visible to User (on browser).

URL

default: Domain / Controller / Action Method.

Localhost.. / Home / Index.

- Views are called through Action Methods.

Example: 01:

```
Public ActionResult Index()
{
return View();
}
```

=> Action Method: Index
View: Index

Example: 02.

```
Public ActionResult Index()
{
return View("Home");
}
```

=> Action Method: Index
View: Home.

SHARED FOLDERS.

→ Should be in View folder.

→ Name should always be the same [Shared]

→ Need: Common folder.
Accesible to all Controller.

PARTIAL VIEW

→ View with in a view.

→ Should be in Shared folder. (if not, full path required)

→ Called using @Html.Partial("ViewName");

RETURN TYPES IN ACTION METHODS

```
Public String Index()
{
    return "Hello World";
}
```

URL: Domain / Home / Index.

* as this one is String, it could be int etc.

PARAMETERS IN ACTION METHOD

You can pass data from URL to controller.

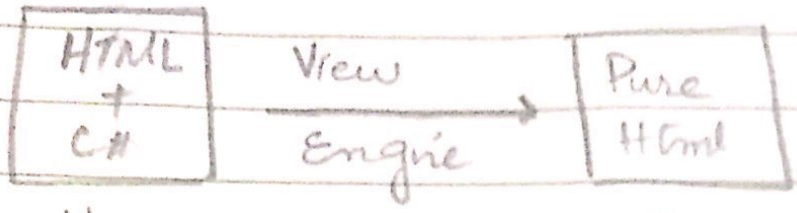
```
Public String hello(int? id)
{
    return "your id is" + id.toString();
}
```

→ always add this when working with int.

=> You can also pass more than one parameters in URL separated by '&'

RAZOR ENGINE:

Since, extension of View is .cshtml where is C#.



- Razor Engine. → System.Web.Razor.

Example: 01.

 5+5 . output: 5+5.

-through Razor. @(5+5) output: 10.

- Comments in Razor like C#.
- If you want to display @ on Browser with razor use @@ (double @).

Example: I am @@ aptech.

- In mail: aatikaliaquat@aptechnn.com (Single ace fbbb6)

Example: 02.

@{ int a=1; int b=2;
int c=a+b; }.

 @c .

FROM CONTROLLER TO VIEW.

→ ViewBag. → ViewData.
→ TempData → Session

ViewData. ViewBag.

In Controller

```
Public ActionResult Index()
{
    ViewData["abc"] = "Aatika";
}
```

```
Public ActionResult Index()
{
    ViewBag.abd = "Liaqat";
}
```

In View

```
<b>@ViewData["abc"]</b>
```

```
<b>@ViewBag.abd</b>
```

Saving in a Variable

```
@{
    String x = @ViewData["abc"].toStyl();
}
```

// Requires Type Casting

```
@{
    Stig x = @ViewBag.abd.
}
```

// Doesn't Require Type Casty

Interchangeble.

```
<b>@ViewData["abd"]</b>
```

```
<b>@ViewBag.abc</b>
```

TempData:

- To store data temporarily for passing from One Action Method to another.
- To hold data to One(next) subsequent request

TempData.Keep() => To store data from one action method to another.

TempData.Peek("ab") => To store as well as display data at a time.

=> if Session is disabled: tempdata won't work.

Session:

- Same as TempData But doesnot go away after refreshing the page
- Only need to be made 'once' then can be in different Views.

HTML HELPER METIDDS.

- Special Methods that returns Html String
- C# Methods used to return HTML.

Types of Html Helpers.

- Inline · Html Helpers.
- Built In · Html Helpers.
  - Standard · Html Helpers.
  - Strongly typed Html Helpers.
  - Templated Html helpers.
- Custome Html Helpers.

Inline Html Helpers.

- Using @helper tag.
- Only on the same view.
- Basically it's a method that can be called in the same view.

Syntax

```
@helper name (parameters)
{
    //code
}
```

Example 01.

```
@helper bold (String xyz)
{
    <b> @xyz </b>
}
```

```
<h1>My name is @bold("aatika")</h1>
```

Example 02.

```
@helper Sum(int a, int b)
{
    <b> @(a+b) </b>;
}
```

```
<h1> The Sum is @Sum(1,3) </h1>
```

Standard Html Helpers.

- Some Commonly Used Standard helpers are,

@Html.ActionLink();
@Html.BeginForm() & @Html.EndForm();
@Html.Label()
@Html.TextBox()
@Html.TextArea()
@Html.Password()
@Html.CheckBox()
@Html.RadioButton()
@Html.DropdownList()
@Html.Hidden()
@Url.Action() // Like Attribute (href) in <a>

Strongly Typed Html Helpers.

→ it fetches properties of the model.

Syntax:

@model <fullyqualified Model Name>
@model WinApp1.Models.MyModel

OR

Using WinApp1.Models
@MyModel.

- Strongly typed Html Helpers methods enable a view to be associated directly with model properties.

- They use Lamda Expressions.
- In Strongly typed, you dont have to worry about spelling mistakes etc.

Syntax:

@Html.TextBoxFor(x => x.property)

TextBox TextBoxFor

Difference B/w Standard Helper & Strongly typed.

@Html.TextBox("Name","Value");

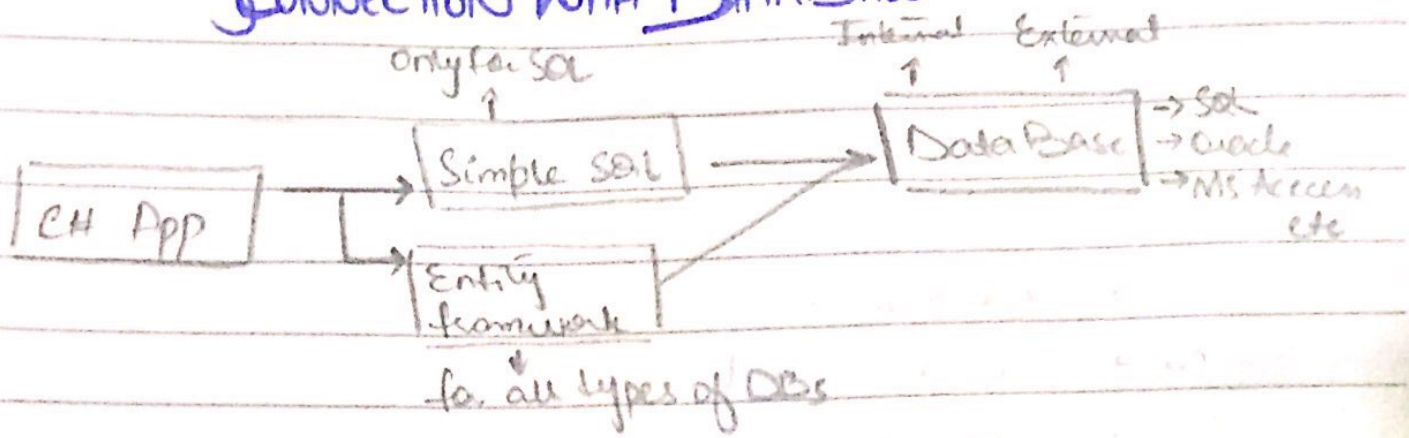
it will not throw any compile time error when you wrongly mention the property Name.
It will throw runtime exception.

But

@Html.TextBoxFor(x => x.Name, ..);

It will give Compile time error (also intellisense).

Connection With Database



Entity Framework

- A framework made using ADO.NET
- It's an ORM (Object Relational Mapping) framework.
  - Properties → Classes
  - DataBase (RDBMS)
- In Entity FramWork, DataBase can be accessed using classes and objects.

Work flow:

- DataBase first Approach.
- Code first Approach.

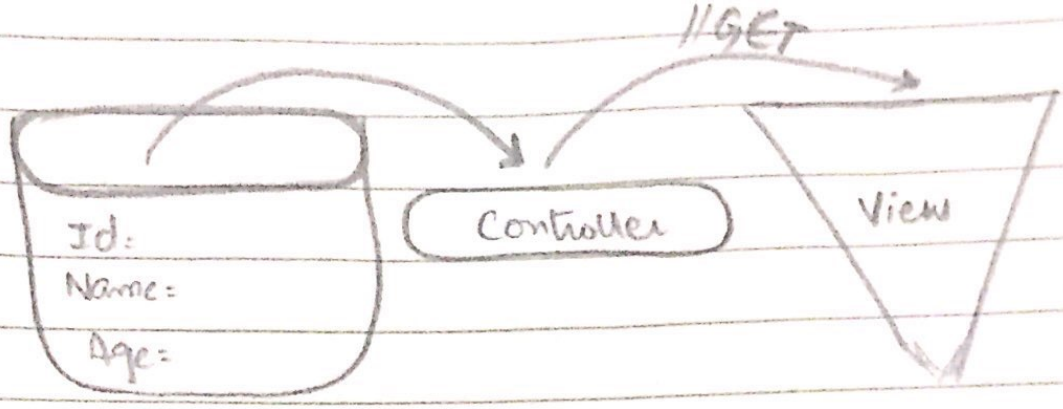
Abbreviations:

- tt : text template
- edmx : Entity Data Model Extensions.

Basic Crud Operations Using Entity frameWork.

C → Create
R → Read (List).
U → Update (Edit).
D → Delete

List



1) DataBase To Controller. //DataBase Object

DataBase1 Entities db = new DataBase1 Entities

Public ActionResult Index() // Get Method

{ return (db.tablenames.tolist()); // List to View.

}

2) Controller To View

@IEnumerator<projectname.tablename>.

<table> <tr> <th> displayname for </th>

</tr>

@foreach (var in Model)

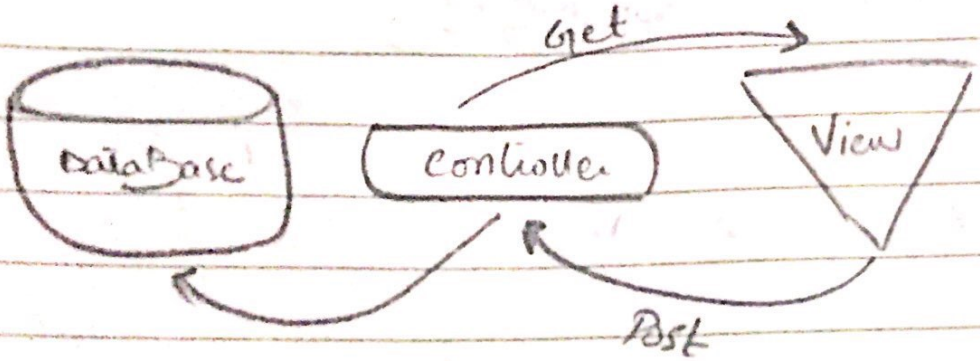
<tr>

<td> display for </td>

</tr>

</table>

Create (Add)



1) Controller to View.

```
Public ActionResult Create()
{
  Return View();
}
```

2) View to Controller.

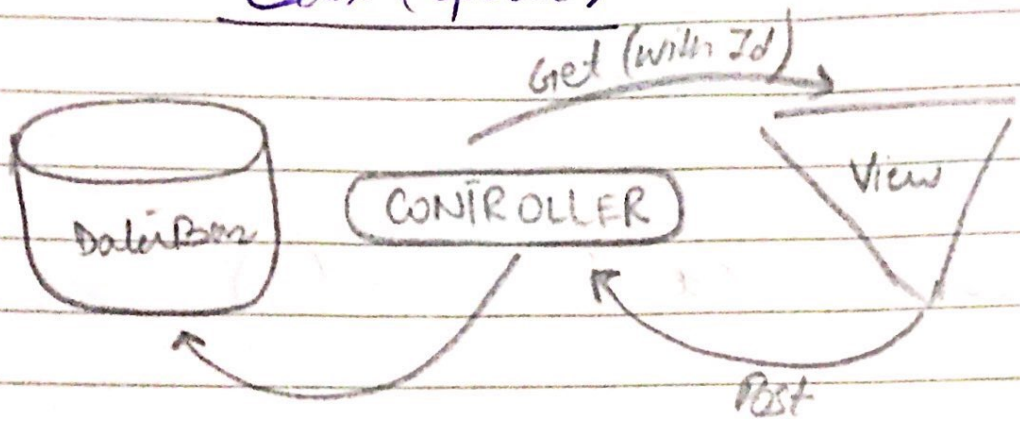
```
@Html.BeginForm{
// Strongly typed View

}
```

3) Controller To DataBase.

```
[HttpPost]
Public ActionResult (table tab)  contains data from View
{
  if(Model.IsValid)
  { db.tables.Add(tab);
    db.SaveChanges();
    return RedirectToAction("Index");
  }
  return View(tab);
}
```

Edit (Update)



1) From Controller To View

```
//Get.
Public ActionResult Edit (int? id)
{
    if (id == null) {
    return HttpNotfound(); }.
  TableName obj = db.tableNames.find(id);
    if (obj == null) {
        return HttpNot-found(); }
    return View(obj);
}
```

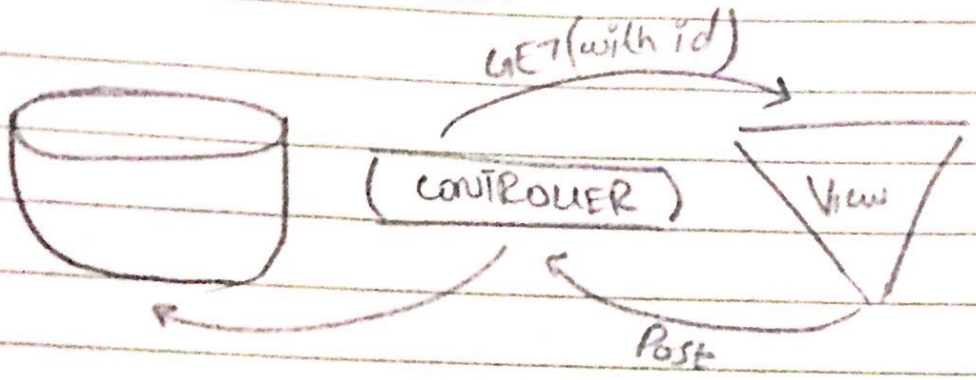
2) From View to Controller

// Strongly typed view

3) From Controller to DataBase

```
[HttpPost]
Public ActionResult Edit (tablename tab)
{
    if (Model.IsValid)
        {
            db.Entry(tab).State=
            db.SaveChages();
        return RedirectToAction("Index");
        }
    return View(tab);
}
```

Delete

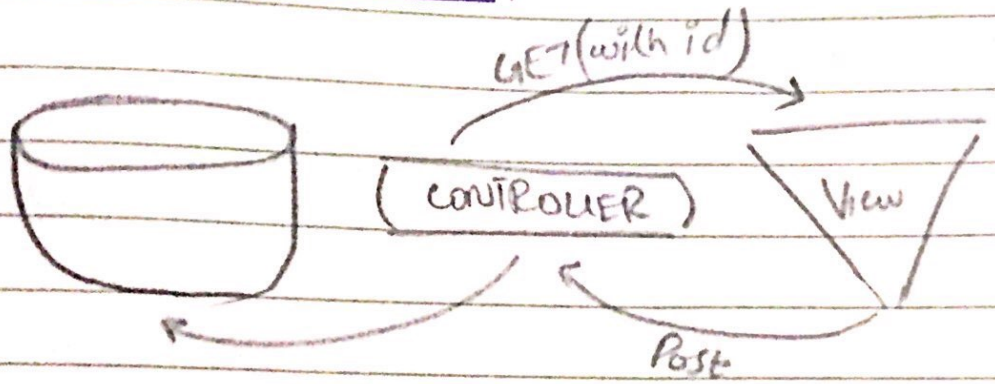


1) From Controller To View
2) From View To Controller } Same As Edit
3) From Controller to View

```
[HttpPost]
Public ActionResult Delete (int id, tablename tab)
{
    table = db.tables.Find(id);
    db.Employee tables.Remove(tab);
    db.SaveChanges();
    return RedirectToAction("Index");
}
```

Important Link: tutlane.com/tutorial/aspnet-mvc

Delete



1) From Controller To View
2) From View To Controller } Same As Edit
3) From Controller to View

```
[HttpPost].
  Public ActionResult Delete (int id, tablename tab)
{
    table = db.tables.Find (id);
    db. Employee tables. Remove (tab);
     db. SaveChanges ();
   return RedirectToAction ("Index");
   }
```

Important Link: tutlane.com/tutorial/aspnet-mvc