MVC

Why the Microsoft launched Asp.Net MVC

Asp.Net MVC at a glance.

Routing in MVC

How to pass data in MVC

ViewData

ViewBag

TempData

Session variables

Modals and Strongly typed views.

Model Binders

Html Helpers in MVC

Data Annotations

View Model

Action Result

Partial Views

Layout in asp.net MVC

AntiForgery Token

ValidateInput and AllowHtml

Async Controllers in MVC

Filters in Asp.Net MVC

Authorize

ActionFilter

Bundling and Minification

Areas in asp.net MVC

Posted on 4/2/2020 11:58:33 AM by Admin

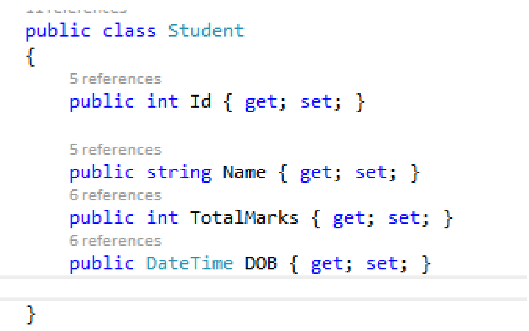
View Model in Asp.Net MVC

MVC is an architectural pattern which separates your application into three major parts for better maintenance and processing.

Asp.Net MVC just follows the principles of MVC architectural pattern. Where Model represents business Entities and business Logic, View represents the UI(Html) and controller handles the http request and coordinates with Model and View to process the request.

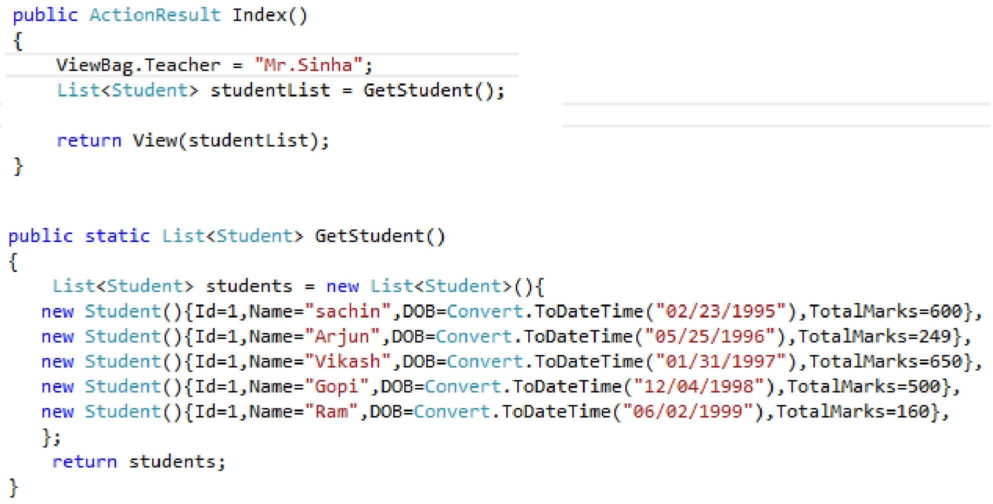
It should be very clear that in MVC the view is only for UI that is pure Html and nothing else. Any other Logic in view is actually violation of MVC principles and considered as bad programming practice.

Now see the below pictures, which will clear your every doubt that why putting any kind of logic in view is a bad idea.



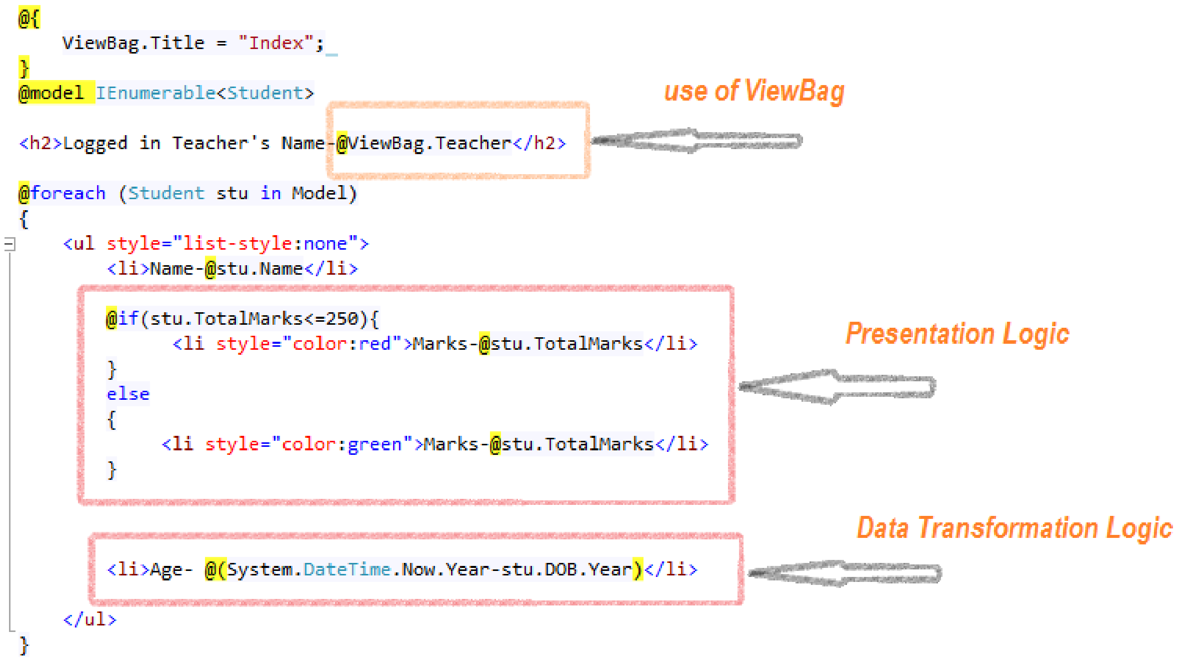
```
public class Student
{
    5 references
    public int Id { get; set; }

    5 references
    public string Name { get; set; }
    6 references
    public int TotalMarks { get; set; }
    6 references
    public DateTime DOB { get; set; }

}
```

Student Model



```
public ActionResult Index()
{
    ViewBag.Teacher = "Mr.Sinha";
    List<Student> studentList = GetStudent();

    return View(studentList);
}


public static List<Student> GetStudent()
{
    List<Student> students = new List<Student>(){
   new Student(){Id=1,Name="sachin",DOB=Convert.ToDateTime("02/23/1995"),TotalMarks=600},
   new Student(){Id=1,Name="Arjun",DOB=Convert.ToDateTime("05/25/1996"),TotalMarks=249},
   new Student(){Id=1,Name="Vikash",DOB=Convert.ToDateTime("01/31/1997"),TotalMarks=650},
   new Student(){Id=1,Name="Gopi",DOB=Convert.ToDateTime("12/04/1998"),TotalMarks=500},
   new Student(){Id=1,Name="Ram",DOB=Convert.ToDateTime("06/02/1999"),TotalMarks=160},
   };
    return students;
}
```

Action Method

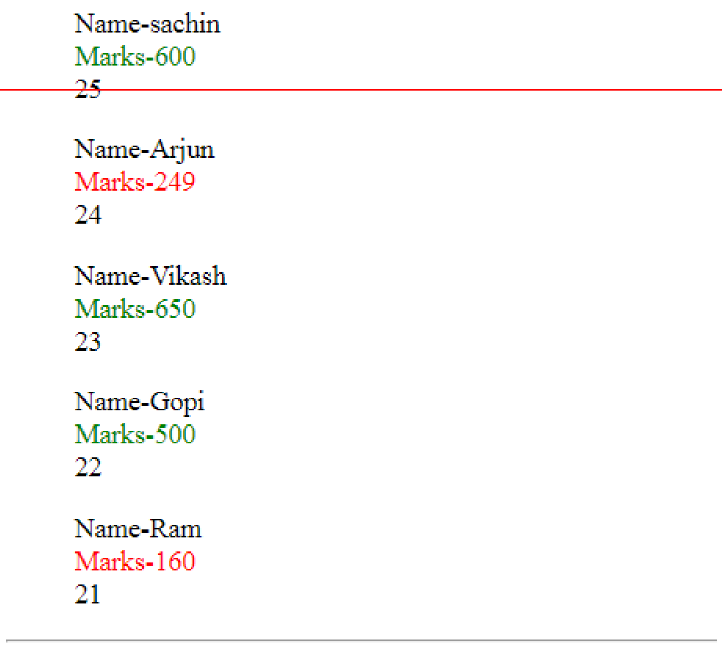


```
@{
    ViewBag.Title = "Index";
}
@model IEnumerable<Student>

<h2>Logged in Teacher's Name-@ViewBag.Teacher</h2>

@foreach (Student stu in Model)
{
    <ul style="list-style:none">
        <li>Name-@stu.Name</li>

        @if(stu.TotalMarks<=250){
              <li style="color:red">Marks-@stu.TotalMarks</li>
        }
        else
        {
             <li style="color:green">Marks-@stu.TotalMarks</li>
        }

        <li>Age- @(System.DateTime.Now.Year-stu.DOB.Year)</li>
    </ul>
}
```

View

MVC

Why the Microsoft launched Asp.Net MVC
Asp.Net MVC at a glance.
Routing in MVC
How to pass data in MVC
ViewData
ViewBag
TempData
Session variables
Modals and Strongly typed views.
Model Binders
Html Helpers in MVC
Data Annotations
View Model
Action Result
Partial Views
Layout in asp.net MVC
AntiForgery Token
ValidateInput and AllowHtml
Async Controllers in MVC
Filters in Asp.Net MVC
Authorize
ActionFilter
Bundling and Minification
Areas in asp.net MVC



```
Name-sachin
Marks-600
25

Name-Arjun
Marks-249
24

Name-Vikash
Marks-650
23

Name-Gopi
Marks-500
22

Name-Ram
Marks-160
21
```

Final Result

The final output is good and expected ,yet it has following problems.

Why putting Presentation Logic in View is bad:-

Here ,If Student's Total Marks is less than or equals to 250 ,his marks is showing in red which symbolizes student has failed in examination and if his marks is greater than 250, then the mark is showing in green meaning he has cleared the examination.

Now ,wait for a moment and think, if tomorrow the criteria for passing the examination changes from 250 ,then we will have to change the UI also, but in reality a UI should only change when any Html related modification is needed .

Why putting Data Transformation Logic in View is bad:-

Here ,I am converting Student's Date of Birth into age by calculating the difference from present date, what if ,tomorrow the requirement changes and age needs to be calculated differently like instead of present date the age should be calculated from a specific date. Again we will have to change the UI, which is obviously not good.

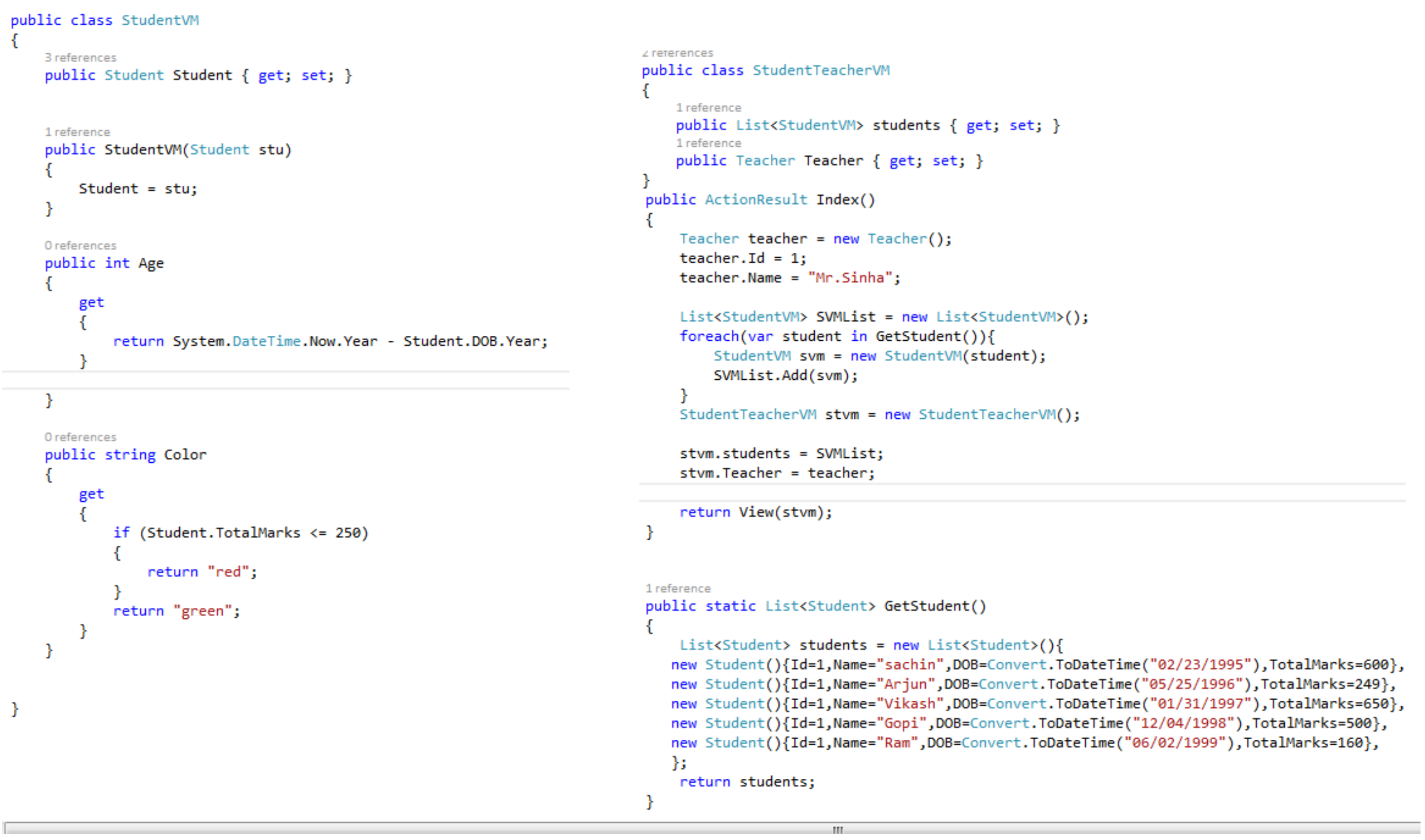
Why ViewBag is bad/How to create strongly typed view based on two model classes:-

ViewBag does boxing and unboxing , accepts Key as a string and do not support intellisense which is bad for performance and can be error prone. That is why, the strongly typed view is always recommended. But a view can only use one model at a time,so using Teacher Model class and Student model class collectively becomes impossible.

All the above problems can be solved by creating another layer Called as View Model. As the name says, a view model is specific to a view and not to the business entities.

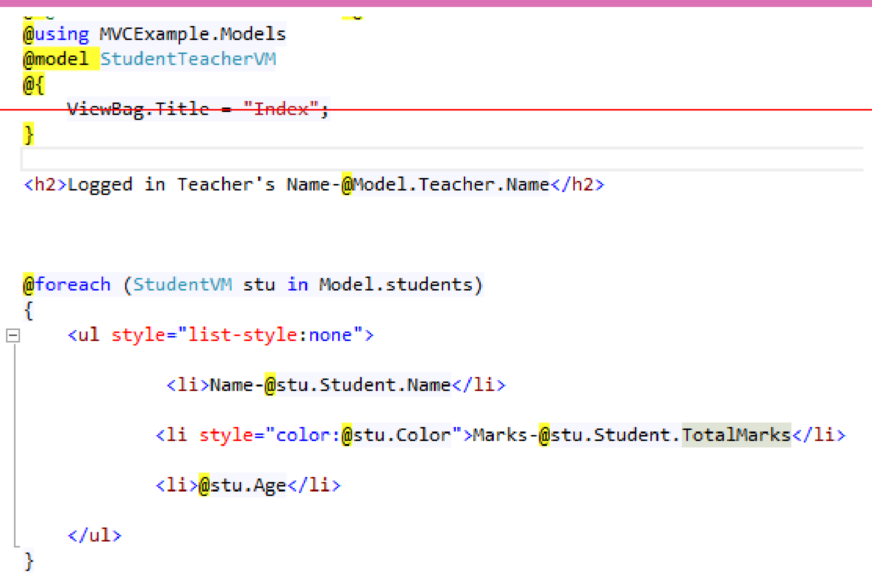
1. Any data transformation logic will be handled by view model.
2. Any presentation Logic will be handled by view model.
3. A view model can contain any number of properties or can be created with two or more model classes.

Let's replace the Student model used in above example with StudentTeacherVM and see how it solves the problems.



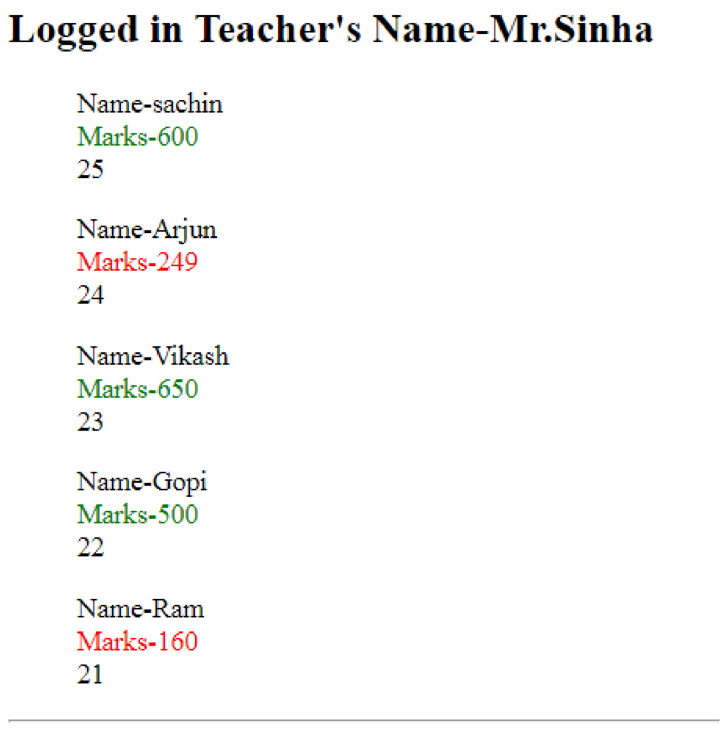
```
public class StudentVM
{
    3 references
    public Student Student { get; set; }


    1 reference
    public StudentVM(Student stu)
    {
        Student = stu;
    }

    0 references
    public int Age
    {
        get
        {
            return System.DateTime.Now.Year - Student.DOB.Year;
        }

    }

    0 references
    public string Color
    {
        get
        {
            if (Student.TotalMarks <= 250)
            {
                return "red";
            }
            return "green";
        }
    }


}
```

```
2 references
public class StudentTeacherVM
{
    1 reference
    public List<StudentVM> students { get; set; }
    1 reference
    public Teacher Teacher { get; set; }
}
 public ActionResult Index()
 {
     Teacher teacher = new Teacher();
     teacher.Id = 1;
     teacher.Name = "Mr.Sinha";

     List<StudentVM> SVMList = new List<StudentVM>();
     foreach(var student in GetStudent()){
         StudentVM svm = new StudentVM(student);
         SVMList.Add(svm);
     }
     StudentTeacherVM stvm = new StudentTeacherVM();

     stvm.students = SVMList;
     stvm.Teacher = teacher;

     return View(stvm);
 }


 1 reference
 public static List<Student> GetStudent()
 {
     List<Student> students = new List<Student>(){
    new Student(){Id=1,Name="sachin",DOB=Convert.ToDateTime("02/23/1995"),TotalMarks=600},
    new Student(){Id=1,Name="Arjun",DOB=Convert.ToDateTime("05/25/1996"),TotalMarks=249},
    new Student(){Id=1,Name="Vikash",DOB=Convert.ToDateTime("01/31/1997"),TotalMarks=650},
    new Student(){Id=1,Name="Gopi",DOB=Convert.ToDateTime("12/04/1998"),TotalMarks=500},
    new Student(){Id=1,Name="Ram",DOB=Convert.ToDateTime("06/02/1999"),TotalMarks=160},
    };
     return students;
 }
```



```
@using MVCExample.Models
@model StudentTeacherVM
@{
    ViewBag.Title = "Index";
}

<h2>Logged in Teacher's Name-@Model.Teacher.Name</h2>


@foreach (StudentVM stu in Model.students)
{
    <ul style="list-style:none">

             <li>Name-@stu.Student.Name</li>

            <li style="color:@stu.Color">Marks-@stu.Student.TotalMarks</li>

            <li>@stu.Age</li>

    </ul>
}
```

View with view model



Logged in Teacher's Name-Mr.Sinha

Name-sachin
Marks-600
25

Name-Arjun
Marks-249
24

Name-Vikash
Marks-650
23

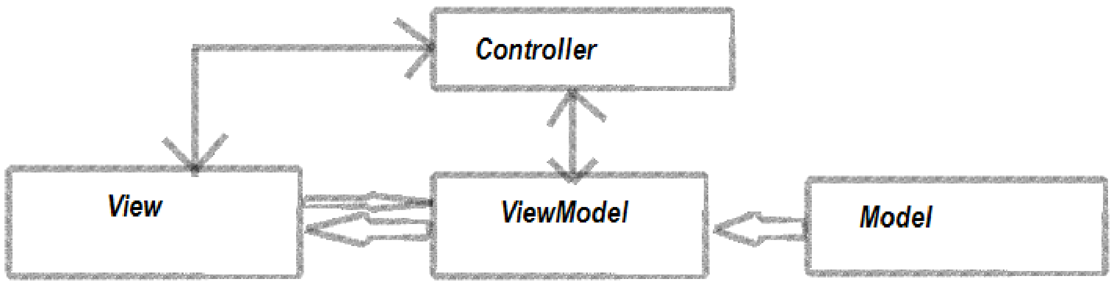
Name-Gopi
Marks-500
22

Name-Ram
Marks-160
21

Final Result

As you can see, the final result is same and all the problems has been solved. This is the power of view model.

Finally any professional MVC project has below structure.



New MVC architecture

View Comments 👍Like 0 👎unlike 0

Comments

Comment:

Submit Please login to comment

Home CSharp WebForms MVC WPF Sql More HowTo Quiz Register Log in

≡

MVC

Why the Microsoft launched Asp.Net MVC

Asp.Net MVC at a glance.

Routing in MVC

How to pass data in MVC

ViewData

ViewBag

TempData

Session variables

Modals and Strongly typed views.

Model Binders

Html Helpers in MVC

Data Annotations

View Model

Action Result

Partial Views

Layout in asp.net MVC

AntiForgery Token

ValidateInput and AllowHtml

Async Controllers in MVC

Filters in Asp.Net MVC

Authorize

ActionFilter

Bundling and Minification

Areas in asp.net MVC

»Asp.net Webforms »SQL »Web Api »WCF

»Asp.net MVC »Entity Framework »Jquery »Bootstrap

»Ado.net »Linq »Wpf »Kendo UI