**Why do we need a ViewModel in ASP.NET Core MVC**  
  
In some cases, a model object may not contain all the data a view needs. This is when we create a **ViewModel**. It contains all the data a view needs.  
  
  
Consider the following Details() action method. Notice, we are passing Employee details and PageTitle to the View. We are using Employee model object to pass Employee details and ViewBag to pass PageTitle.

public class HomeController : Controller  
{  
    // Other Code  
  
    public ViewResult Details()  
    {  
        Employee model = \_employeeRepository.GetEmployee(1);  
        ViewBag.PageTitle = "Employee Details";  
  
        return View(model);  
    }  
}

Using ViewBag to pass data from a controller to a view creates a loosely typed view and as a result we will not have intellisense and compile-time type and error checking. It does not make any sense, to include PageTitle property in the Employee class, because an Employee class should only contain Employee related properties.   
  
This is when we create a class, that contains all the data a view needs. This class is called a ViewModel.   
  
**ViewModel Example**  
  
**ViewModel**classes can live anywhere in your ASP.NET Core MVC project, but to keep things better organised, we usually place them in a folder that is named **ViewModels**   
  
We named the **ViewModel**class HomeDetailsViewModel. We have the word "Home" in the name, because, the name of the Controller is HomeController. The word "Details"because, the name of the action method is Details()   
  
This ViewModel class contains all the data the view needs. In general, we use **ViewModels**to shuttle data between a View and a Controller. For this reason ViewModels are also called as **Data Transfer Objects**or **DTO** in short.   
  
We can now use this ViewModel to pass both Employee details and PageTitle from the Controller to the View

namespace EmployeeManagement.ViewModels  
{  
    public class HomeDetailsViewModel  
    {  
        public Employee Employee { get; set; }  
        public string PageTitle { get; set; }  
    }  
}

Employee class is in **EmployeeManagement.Models**namespace, so please don't forget to include the following **using**statement.

using EmployeeManagement.Models;

**Using ViewModel - Controller Code**

public ViewResult Details()  
{  
    // Instantiate HomeDetailsViewModel and store Employee details and PageTitle  
    HomeDetailsViewModel homeDetailsViewModel = new HomeDetailsViewModel()  
    {  
        Employee = \_employeeRepository.GetEmployee(1),  
        PageTitle = "Employee Details"  
    };  
  
    // Pass the ViewModel object to the View() helper method  
    return View(homeDetailsViewModel);  
}

**Using ViewModel - View Code**

* The view has access to the **ViewModel**object that the controller has passed using the View() helper method
* Using the @model directive, set HomeDetailsViewModel as the Model for the view
* We can then access Employee details and PageTitle uisng @Model property
* Please note, @model directive has a lowercase **"m"** and @Model property has a capital letter **"M"**

@model EmployeeManagement.ViewModels.HomeDetailsViewModel

<html>

<head>

    <title></title>

</head>

<body>

    <h3>@Model.PageTitle</h3>

    <div>

        Name : @Model.Employee.Name

    </div>

    <div>

        Email : @Model.Employee.Email

    </div>

    <div>

        Department : @Model.Employee.Department

    </div>

</body>

</html>