In this video, we will discuss **displaying images in MVC application**. In Part 48, we will create a custom html helper to display images. We will be using the example, that we worked with in [Part 46](http://csharp-video-tutorials.blogspot.com/2013/06/part-46-accessing-model-metadata-from.html). We want to display **Employee**photo, along with the personal details as you can see in the image below.   


**Alter table tblEmployee to add Photo, and AlternateText columns.**  
Alter table tblEmployee Add Photo nvarchar(100), AlternateText nvarchar(100)  
  
**Update Photo and AlternateText columns**  
Update tblEmployee set Photo='~/Photos/JohnSmith.png',   
AlternateText = 'John Smith Photo' where Id = 1  
  
Right click on the solution explorer and add **"Photos"** folder. Download the following image and paste in **"Photos"** folder.   
   
  
**Now, in MVCDemo project, update SampleDataModel.edmx.**  
**1.** Right click on **"Employee"** table and select **"Update Model from database"** option  
**2.** On **"Update Wizard"** window, click on **"Refresh"** tab  
**3.** Expand tables node, and select **"tblEmployee"** table  
**4.** Finally click **"Finish"**  
  
At this point, **"Photo"** and **"AlternateText"** properties must be added to the auto-generated **"Employee"** class.   
  
**Generate Details view**  
**1.** Delete **"Details.cshtml"** view, if it already exists.   
**2.** Right click on **"Details"** action method and select **"Add View"**  
**3.** In **"Add View"** window, set  
View Name = Details  
View engine = Razor  
Create a strongly typed view = Select the checkbox  
Model class = Employee(MVCDemo.Models)  
Scaffold template = Details  
  
Notice that for **Photo** and **AlternateText** properties, the following HTML is generated. At this point, if you run the application, instead of rendering the photo, the PhotoPath and AlternateText property values are displayed.  
<div class="display-label">  
        @Html.DisplayNameFor(model => model.Photo)  
</div>  
<div class="display-field">  
    @Html.DisplayFor(model => model.Photo)  
</div>  
  
<div class="display-label">  
        @Html.DisplayNameFor(model => model.AlternateText)  
</div>  
<div class="display-field">  
    @Html.DisplayFor(model => model.AlternateText)  
</div>  
  
Replace the above code with the following. Notice that, we are using **Url.Content()** html helper method. This method resolves a url for a resource when we pass it the relative path.  
<div class="display-label">  
    @Html.DisplayNameFor(model => model.Photo)  
</div>  
<div class="display-field">  
    <img src="@Url.Content(@Model.Photo)" alt="@Model.AlternateText" />  
</div>  
  
Run the application, and notice that, the image is rendered as expected. In our next video, we will discuss creating a custom html image helper.