The controller responds to URL request, gets data from a model and hands it over to the view. The view then renders the data. Model can be entities or business objects.

In part 7, we have built **Employee** entity.

```
public class Employee
{
    public int EmployeeId { get; set; }
    public string Name { get; set; }
    public string Gender { get; set; }
    public string City { get; set; }
}
```

In this video, we will discuss, retrieving data from a database table **tblEmployee** using entity framework. In a later video, we will discuss using business objects as our model.

Step 1: Install entity framework, if you don't have it installed already on your computer. At the time of this recording the latest version is 5.0.0.0. Using nuget package manager, is the easiest way to install. A reference to EntityFramework.dll is automatically added.
Open visual studio > Tools > Library Package Manager > Manage NuGet Packages for Solution

Step 2: Add **EmployeeContext.cs** class file to the Models folder. Add the following**"using"** declaration.

```
using System.Data.Entity;
```

Copy & paste the following code in EmployeeContext.cs

```
public class EmployeeContext : DbContext
{
    public DbSet<Employee> Employees {get; set;}
}
```

EmployeeContext class derives from **DbContext** class, and is responsible for establishing a connection to the database. So the next step, is to include connection string in web.config file.

Step 3: Add a connection string, to the web.config file, in the root directory.

```
<connectionStrings>
  <add name="EmployeeContext"
       connectionString="server=.; database=Sample; integrated security=SSPI"
       providerName="System.Data.SqlClient"/>
</connectionStrings>
```

Step 4: Map **"Employee"** model class to the database table, tblEmployee using **"Table"**attribute as shown below.

```
[Table("tblEmployee")]
public class Employee
{
    public int EmployeeId { get; set; }
    public string Name { get; set; }
    public string Gender { get; set; }
    public string City { get; set; }
}
```

Note: "Table" attribute is present in "System.ComponentModel.DataAnnotations.Schema" namespace.

Step 5: Make the changes to "Details()" action method in "EmployeeController" as shown below.

```
public ActionResult Details(int id)
{
    EmployeeContext employeeContext = new EmployeeContext();
    Employee employee = employeeContext.Employees.Single(x => x.EmployeeId == id);

    return View(employee);
}
```

Step 6: Finally, copy and paste the following code in **Application_Start**() function, in**Global.asax** file. Database class is present "in System.Data.Entity" namespace. Existing databases do not need, database initializer so it can be turned off.

```
Database.SetInitializer<MVCDemo.Models.EmployeeContext>(null);
```

That's it, run the application and navigate to the following URL's and notice that the relevant employee details are displayed as expected