# ASP.NET MVC 5.0

Lesson 4: Exploring Models & working with data

# Lesson Objectives

- Model Binding
- Data Annotations

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

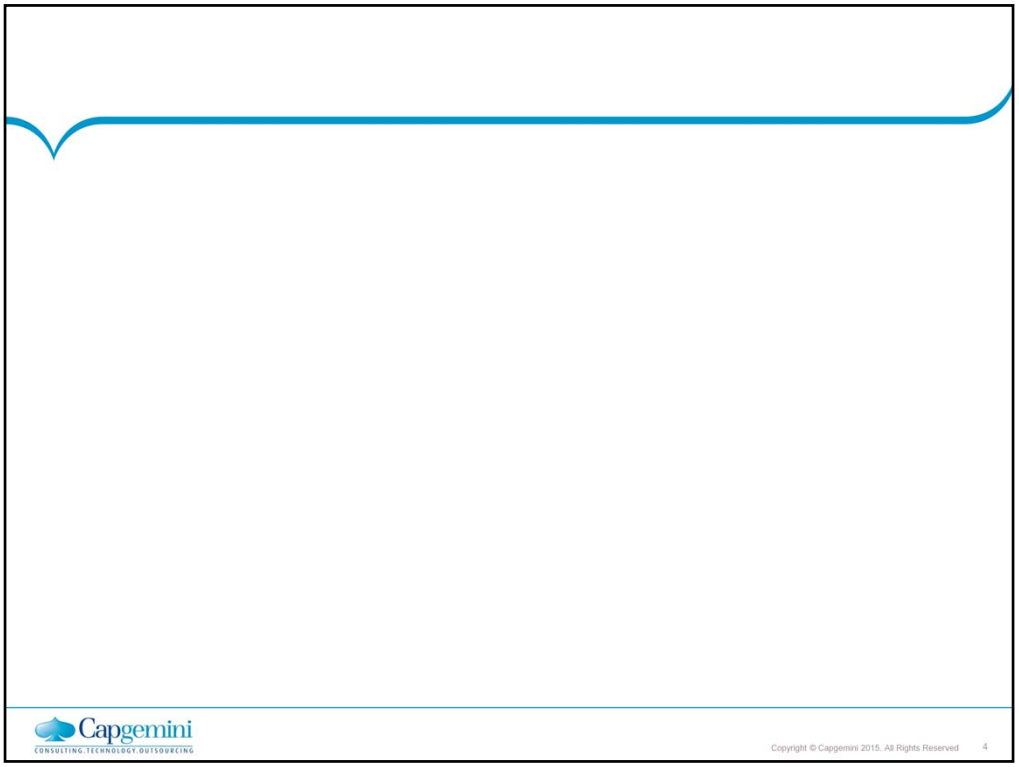
 2

4.1
## Model Binding

- Model Binding is mapping the HTTP request data directly to Action method parameters and .NET objects (a Model).
- ASP.NET MVC framework provides a very powerful model binder that can bind most of the data types like
  - Primitive types
  - Array & Collection
  - Objects

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 3




Binding to Primitive Values

Query String Parameters can directly map to the Action method parameters(Primitive types)

Request Uri:

http://hostname/home/printemployeeidname?employeeid=1&name=Karthik

HomeController.cs

```
 public ActionResult PrintEmployeeIdName(int employeeId,string name)
 {
   return Content(string.Format("EmployeeId = {0} Name= {1}",employeeId,name));
      }
   }
```

Binding to Collections

Collection can be simply an array, or a collection like IEnumerable<T>, ICollection<T>, and IList<T>. The type IDictionary<TKey, TValue> can also be binded.

HomeController.cs

```
 public ActionResult PrintListNumbers(IList<int> numbers)
      {
         StringBuilder sb = new StringBuilder();
         foreach (var number in numbers)
         {
            sb.AppendFormat("{0}<br/>", number);
         }
         return Content(sb.ToString());
      }
```

**Index.cshtml**

```
<form method="get" action="@Url.Action("PrintListNumbers")">
   <input type="text" name="numbers[0]" value="1" />
   <input type="text" name="numbers[1]" value="2" />
   <input type="text" name="numbers[2]" value="3" />
   <input type="submit" value="Submit" />
</form>
```

**Binding to Object**

Trainer.cs(Model)

```
 public class Trainer     {
       public int TrainerId { get; set; }
       public string TrainerName { get; set; }
       public Technology Subject { get; set; }          }
```

Technology.cs(Model)

```
    public class Technology     {
       public int TechnologyId { get; set; }
       public string TechnologyName { get; set; }        }
```

HomeController.cs

```
 public ActionResult PrintTrainer(Trainer trainer)         {
          return Content(string.Format("TrainerId = {0} Name= {1} Technology = {2}",
trainer.TrainerId, trainer.TrainerName, trainer.Subject.TechnologyName));           }
```

We can add binding attribute to instruct model binder when to bind property and when to exclude property

```
public ActionResult Create([Bind(Exclude="StudentId",
Include="StudentName")]Student student)
{ ... }
```

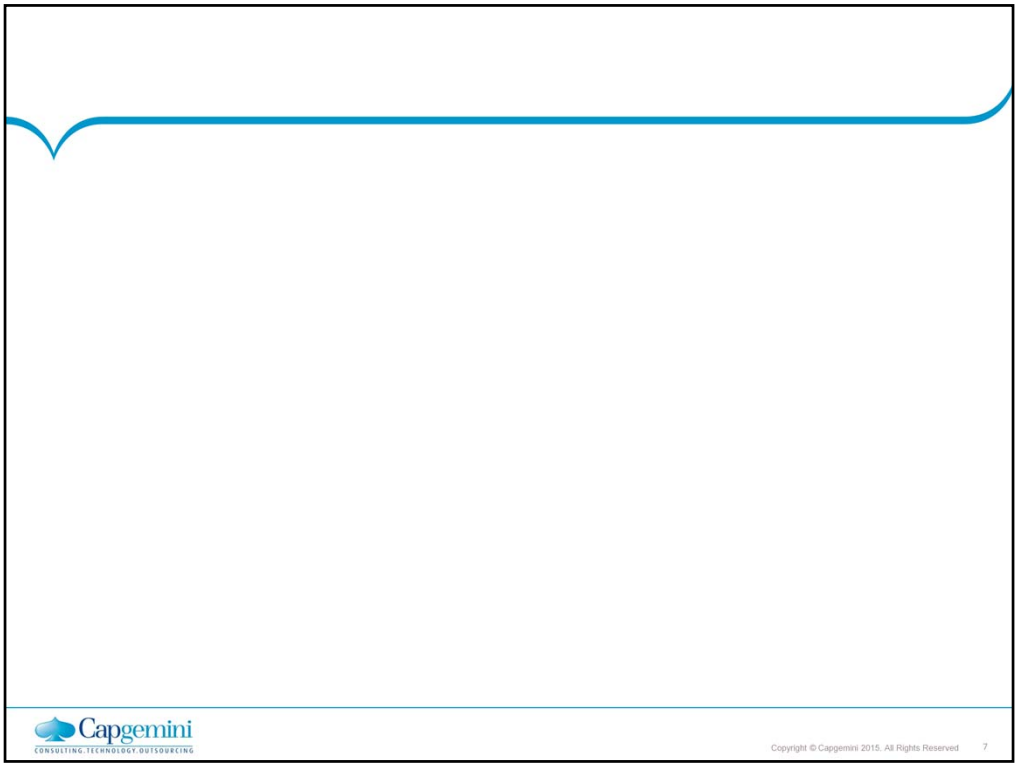
4.2

## Data Annotations

- Data Validation can be easily applied to models by using Data Annotation attribute.
- Data Annotation attributes classes are present in System. Component Model. Data Annotations
- Data Annotations help to define the rules to the model classes or properties for data validation and displaying suitable messages to end users.
- Client side validation can be enabled by referring jquery.validate.js and jquery.unobtrusive-ajax.js

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 6




MVC framework provides rich support to use data annotation to describe the model for data validation and visual hint.

Required - Specify a property as required.
[Required]
public int ID { get; set; }

RegularExpression - validate the value of a property by specified regular expression pattern.
[RegularExpression(@"^\d{6}$")]
public int TrainerId { get; set; }

Range - validate the value of a property with in a specified range of values.
[Range(15, 100)]
public int age { get; set; }

StringLength - specify min and max length for a string property.
[StringLength(50, MinimumLength = 2)]
public string name { get; set; }

MaxLength - specify max length for a string property.
[MaxLength(30)]
public string TrainerName { get; set; }

DataType
Specify the datatype of a property

[DataType(DataType.Date)]
public DateTime updatedate { get; set; }

[DataType(DataType.PhoneNumber)]
public string phone { get; set; }


Other Datatypes

| | |
|---|---|
| DateTime | -Represents a date and time of day. |
| Date | -Represents a date value. |
| Time | -Represents a time value. |
| PhoneNumber | -Represents a phone number value. |
| Currency | -Represents a currency value. |
| EmailAddress | -Represents an e-mail address. |
| Password | -Represent a password value. |
| Url | -Represents a URL value. |
| CreditCard | -Represents a credit card number. |
| PostalCode | -Represents a postal code. |

DisplayName
Specify the display name for a property.

```
[DisplayName("Employee Name")]
public String EmpName { get; set; }
```

DisplayFormat
Specify the display format for a property like different format for Date property.

```
[DisplayFormat(DataFormatString = "{0:d}")]
  public DateTime JoiningDate { get; set; }
```

ScaffoldColumn
Specify fields for hiding from editor forms.

```
 [ScaffoldColumn(false)]
 public int EmpId { get; set; }
```

# DEMO

- Models Demo


Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 9

## Summary

- We can bind most of the data types like primitive types, Array & Collection and Objects
- Data Annotations allows us to describe the rules which we need to apply to the model properties
- Using System. Component Model. Data Annotations attributes we can perform client and server validation without any additional coding
- We need to set Client Validation Enabled and Unobtrusive Java Script Enabled to true in the web. config file to perform client validation.


Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 10