

Sql server, .net and c# video tutorial

Free C#, .Net and Sql server video tutorial for beginners and intermediate programmers.

Support us .Net Basics C# SQL ASP.NET ADO.NET MVC Slides C# Programs Subscribe Buy DVD

ASP.NET Core Model Binding

Suggested Videos

Part 38 - ASP.NET Core Environment Tag Helper | Text | Slides

Part 39 - Bootstrap navigation menu in asp.net core application | Text | Slides

Part 40 - Form tag helpers in asp.net core | Text | Slides

In this video we will discuss **Model Binding in ASP.NET Core with examples.**

What is Model Binding

- Model binding maps data in an HTTP request to controller action method parameters
- The action parameters may be simple types such as integers, strings, etc or complex types like Customer, Employee, Order etc.
- Model binding is great, because without it we have to write lot of custom code to map request data to action method parameters which is not only tedious but also error prone.

ASP.NET Core Model Binding Example

When an HTTP request arrives at our MVC application it is the Controller action method that handles the incoming request. Let's say we want to view **employee details whose ID is 2**. For this we issue a GET request to the following URL

<http://localhost:48118/home/details/2>

Our application default route template (`{{controller=Home}}/{{action=Index}}/{id?}}`) routes this request to `Details(int? id)` action method of the `HomeController`.

```
public IActionResult Details(int? id)
{
    HomeDetailsViewModel homeDetailsViewModel = new HomeDetailsViewModel()
    {
        Employee = _employeeRepository.GetEmployee(id ?? 1),
        PageTitle = "Employee Details"
    };

    return View(homeDetailsViewModel);
}
```

The `id value 2` in the request URL is mapped to the `id` parameter of the `details(int? id)` action method. MVC will bind the data in the request to the action parameters by name. Notice, in the above example, the parameter name in the default route template is `"id"` and the parameter name on the `Details(int? id)` action method is also `id`. So the value `2` in the URL (<http://localhost:48118/home/details/2>) is mapped to the `id` parameter on the `Details(int? id)` action method.

Another Example



Best software training and placements in marathahalli, bangalore. For further details please call 09945699393.

Complete Tutorials

JavaScript tutorial

Bootstrap tutorial

Angular tutorial for beginners

Angular 5 Tutorial for beginners

Important Videos

The Gift of Education

Web application for your business

How to become .NET developer

Resources available to help you

Dot Net Video Tutorials

ASP.NET Core Tutorial

Angular 6 Tutorial

Angular CRUD Tutorial

Angular CLI Tutorial

Angular 2 Tutorial

Design Patterns

SOLID Principles

ASP.NET Web API

Bootstrap

AngularJS Tutorial

jQuery Tutorial

JavaScript with ASP.NET Tutorial

Request URL

<http://localhost:48118/home/details/2?name=pragim>

The following *Details()* action method handles the above request URL and maps the value *2* to the *id* parameter and the value *pragim* to the *name* parameter

```
public string Details(int? id, string name)
{
    return "id = " + id.Value.ToString() + " and name = " + name;
}
```

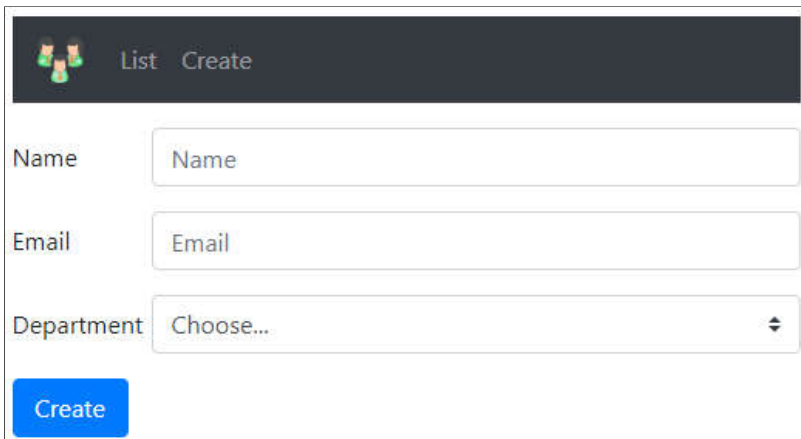
HTTP Request Data

To bind the request data to the controller action method parameters, model binding looks for data in the HTTP request in the following places in the order specified below.

- Form values
- Route values
- Query strings

Model Binding Complex Types

Model Binding also works with complex types such as Customer, Order, Employee etc. Consider the following *"Create Employee Form"*



When the above form is posted to the server, the values in the form are mapped to the Employee object parameter of the following *Create()* action method.

```
public RedirectToActionResult Create(Employee employee)
{
    Employee newEmployee = _employeeRepository.Add(employee);
    return RedirectToAction("details", new { id = newEmployee.Id });
}
```

- Model binder in asp.net core binds the posted form values to the properties of the Employee object that is passed as a parameter to the *Create()* action method.
- The value in the input element that has the *name* attribute set to *"Name"* is mapped to the *Name* property of the Employee object
- Similarly value in the input element with name *"Email"* is mapped to *Email* property of the Employee object
- The same is true for *Department*

Add() method - IEmployeeRepository interface

```
public interface IEmployeeRepository
{
    Employee GetEmployee(int Id);
}
```

[JavaScript Tutorial](#)
[Charts Tutorial](#)
[LINQ](#)
[LINQ to SQL](#)
[LINQ to XML](#)
[Entity Framework](#)
[WCF](#)
[ASP.NET Web Services](#)
[Dot Net Basics](#)
[C#](#)
[SQL Server](#)
[ADO.NET](#)
[ASP.NET](#)
[GridView](#)
[ASP.NET MVC](#)
[Visual Studio Tips and Tricks](#)
[Dot Net Interview Questions](#)
Slides
[Entity Framework](#)
[WCF](#)
[ASP.NET Web Services](#)
[Dot Net Basics](#)
[C#](#)
[SQL Server](#)
[ADO.NET](#)
[ASP.NET](#)
[GridView](#)
[ASP.NET MVC](#)
[Visual Studio Tips and Tricks](#)
Java Video Tutorials
[Part 1 : Video | Text | Slides](#)
[Part 2 : Video | Text | Slides](#)
[Part 3 : Video | Text | Slides](#)

```

IEnumerable<Employee> GetAllEmployees();
Employee Add(Employee employee);
}

```

Add() method - MockEmployeeRepository class

```

public class MockEmployeeRepository : IEmployeeRepository
{
    private List<Employee> _employeeList;

    public MockEmployeeRepository()
    {
        _employeeList = new List<Employee>()
        {
            new Employee() { Id = 1, Name = "Mary", Department = Dept.HR, Email =
"mary@pragimtech.com" },
            new Employee() { Id = 2, Name = "John", Department = Dept.IT, Email =
"john@pragimtech.com" },
            new Employee() { Id = 3, Name = "Sam", Department = Dept.IT, Email =
"sam@pragimtech.com" },
        };
    }

    public Employee Add(Employee employee)
    {
        employee.Id = _employeeList.Max(e => e.Id) + 1;
        _employeeList.Add(employee);
        return employee;
    }

    public IEnumerable<Employee> GetAllEmployees()
    {
        return _employeeList;
    }

    public Employee GetEmployee(int Id)
    {
        return this._employeeList.FirstOrDefault(e => e.Id == Id);
    }
}

```

HttpGet vs HttpPost

At the moment in the `HomeController` we have the following 2 `Create()` action methods.

```

public IActionResult Create()
{
    return View();
}

public RedirectToActionResult Create(Employee employee)
{
    Employee newEmployee = _employeeRepository.Add(employee);
    return RedirectToAction("details", new { id = newEmployee.Id });
}

```

If we now navigate to the URL (<http://localhost:1234/home/create>) we get the following error

AmbiguousActionException: Multiple actions matched.

This is because asp.net core does not know which action method to execute. We want the first `Create()` action method to respond to GET request and the second `Create()` action method to respond to the POST request. To tell this to asp.net core decorate the `Create()` action methods with `HttpGet` and `HttpPost` attributes as shown below.

```

[HttpGet]
public IActionResult Create()
{
    return View();
}

```

Interview Questions

C#

SQL Server

Written Test

```
[HttpPost]
public RedirectToActionResult Create(Employee employee)
{
    Employee newEmployee = _employeeRepository.Add(employee);
    return RedirectToAction("details", new { id = newEmployee.Id });
}
```

The [Create\(\)](#) action method that responds to the POST action adds the new employee to the [EmployeeRepository](#) and redirects the user to the [Details\(\)](#) action method passing it the [ID](#) of the newly created employee.

Upon redirection if you get a [NullReferenceException](#), make sure to use [AddSingleton\(\)](#) method instead of [AddTransient\(\)](#) method to register [IEmployeeRepository](#) service in [ConfigureServices\(\)](#) method of [Startup.cs](#) file.

```
public class Startup
{
    public void ConfigureServices(IServiceCollection services)
    {
        services.AddMvc();
        services.AddSingleton<IEmployeeRepository, MockEmployeeRepository>();
    }

    // Other code here....
}
```

We will discuss what's causing this error and the difference between [AddSingleton\(\)](#), [AddTransient\(\)](#) and [AddScoped\(\)](#) methods in our upcoming videos.

At the moment [Create Employee Form](#) does not have any validation in place. If we submit the form without filling any of the form fields, we will end up creating a new employee whose [name](#) and [email](#) fields are null. We will discuss form validation in our next video.



No comments:

Post a Comment

If you like this website, please share with your friends on facebook and Google+ and recommend us on google using the g+1 button on the top right hand corner.

Enter your comment...



Comment as:

toilati123vn@g ▼

Publish

Preview

Links to this post

[Create a Link](#)

[Newer Post](#)

[Home](#)

[Older Post](#)

Subscribe to: [Post Comments \(Atom\)](#)

Powered by Blogger.