Assembling the Building Blocks

Matt Honeycutt

@matthoneycutt | http://trycatchfail.com/strongly-typed-angularjs

Overview

ASP.NET MVC to JavaScript

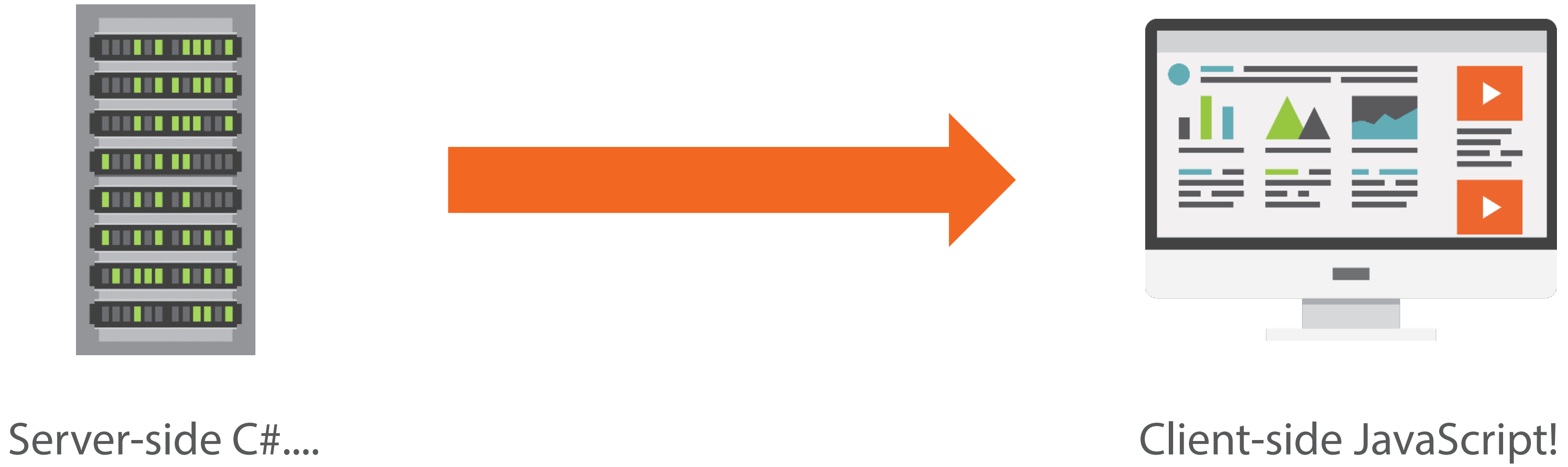
Razor to JavaScript

URLs with strong-typing!

Razor-Powered AngularJS templates!

This is the foundation!

Passing Data From MVC Actions to JavaScript



Passing Data From MVC Actions to JavaScript

```
public class CustomerViewModel
{
 public int Id { get; set; }
 public string Name { get; set; }
 public string WorkEmail { get; set; }
}
```

Our Action Result

```
{
 id: 5,
 name: ‘john smith’,
 workEmail: ‘test@user.com’
}
```

Passing Data From MVC Actions to JavaScript

```
public class CustomerViewModel
{
 public int Id { get; set; }
 public string Name { get; set; }
 public string WorkEmail { get; set; }
}
```

Standard
JsonResult

```
{
 Id: 5,
 Name: ‘john smith’,
 WorkEmail: ‘test@user.com’
}
```

Why Not Use Web API?

We don’t need it

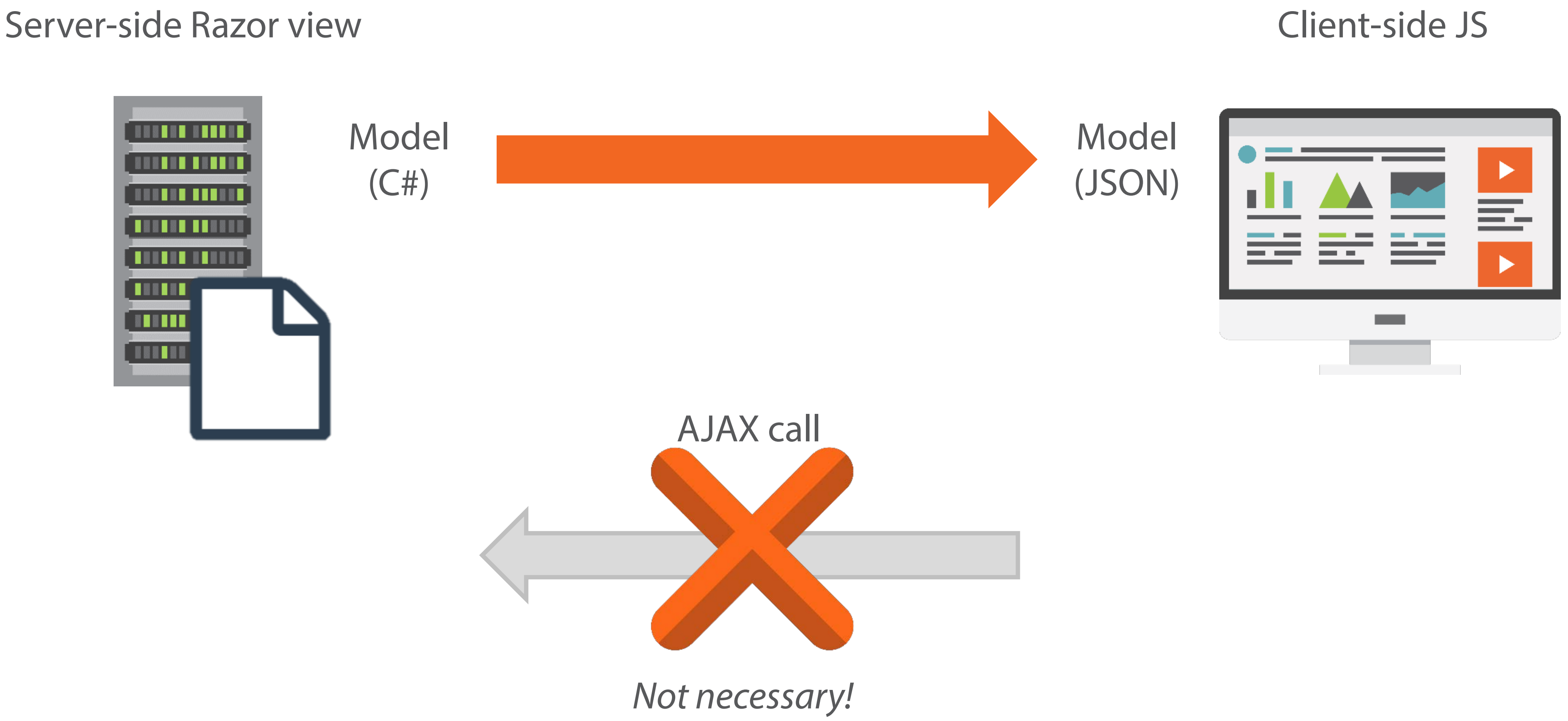
We aren’t really building an API…

We just need JSON!

Keep it simple!

Building A Better JSON Result

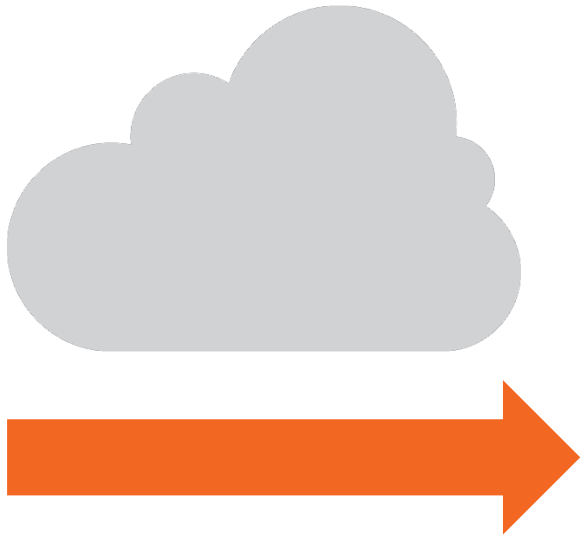
Passing Objects From Razor Views to JavaScript



Html.JsonFor

Server-side:
Razor CSHTML

```
<script>
window.app.constant('model',
 @Html.JsonFor(Model));
</script>
```



Client-side:
HTML and JavaScript

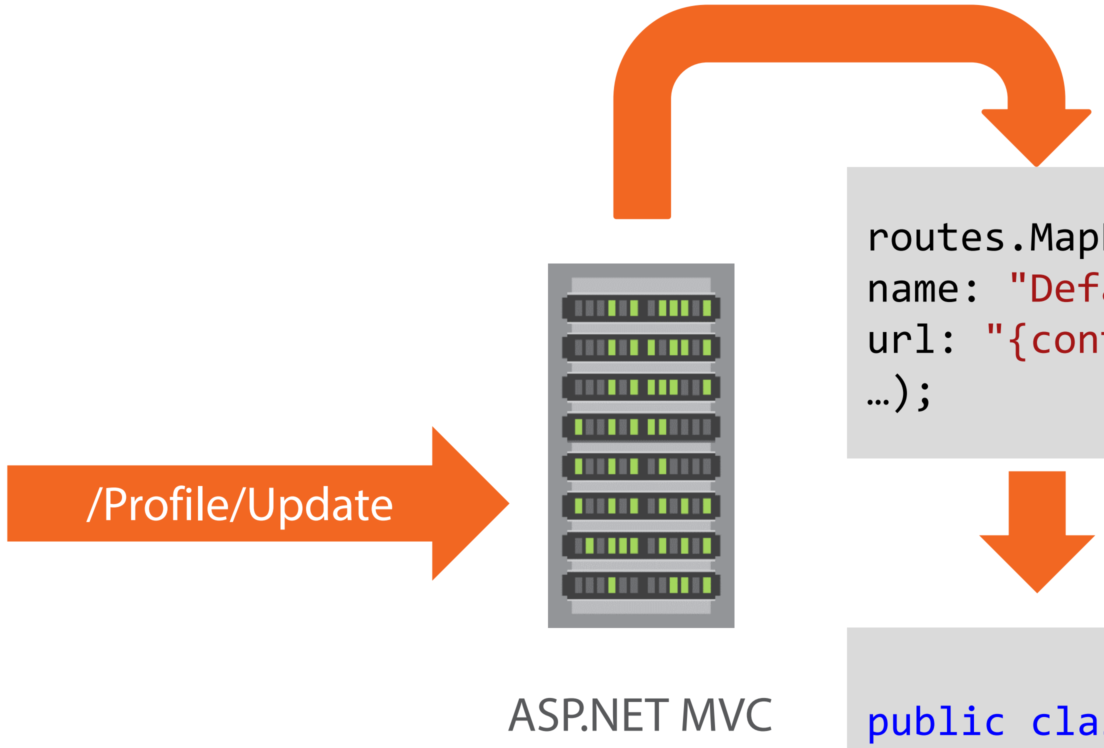
```
<script>
window.app.constant('model',
{
"fullName":"Matt Honeycutt",
"emailAddress":
"matt.honeycutt@me.com"
});
</script>
```

Passing Data From Razor to AngularJS

The Problem with URLs…

EditProfileController.js

```
$http.post(
      '/Profile/Update',
      vm.profile)
.success(function() {
      vm.success = true;
})
```



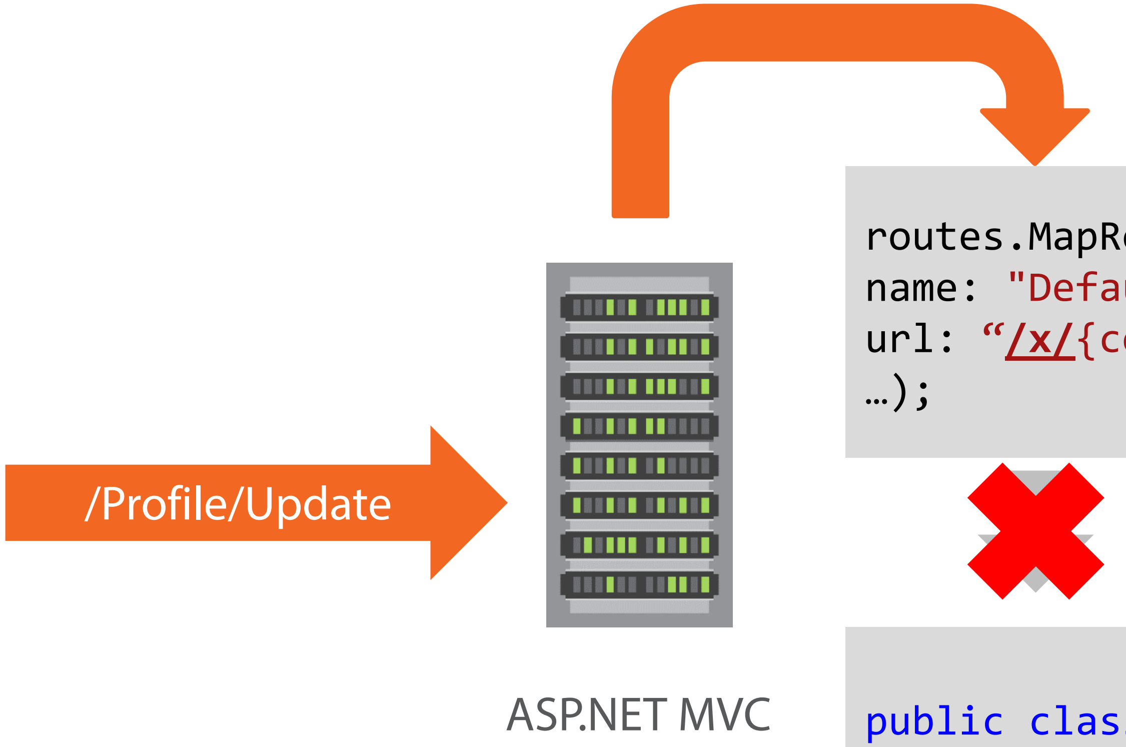
```
routes.MapRoute(
name: "Default",
url: "{controller}/{action}/{id}",
…);
```

```
public class ProfileController
{
public JsonResult Update(…) …
}
```

The Problem with URLs…

EditProfileController.js

```
$http.post(
       '/Profile/Update',
       vm.profile)
.success(function() {
       vm.success = true;
})
```



```
routes.MapRoute(
name: "Default",
url: "/x/{controller}/{action}/",
…);
```

```
public class UserProfileController
{
public JsonResult Update(…) …
}
```

Passing Strongly-Typed URLs to AngularJS

Using Razor in AngularJS Templates

Model metadata

Strongly-typed binding

Angular-powered templates

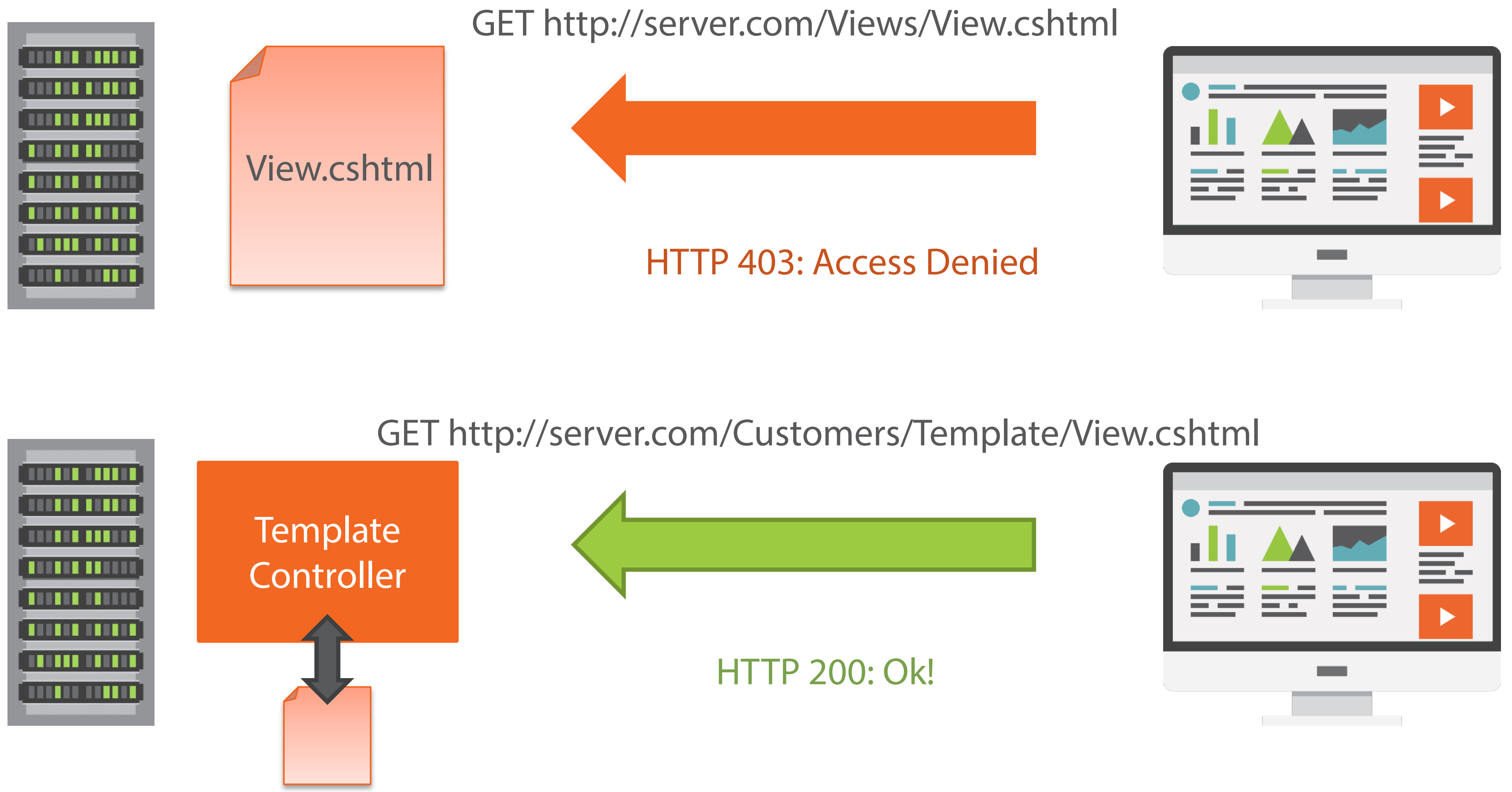
.NET's Type System

HTML5 validation attributes

Strongly-typed forms

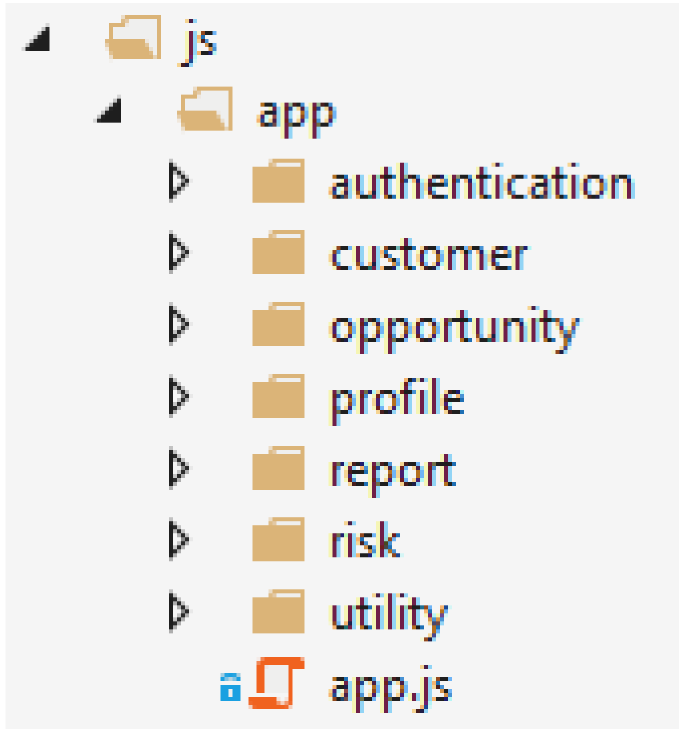
Data annoations

Accessing Razor From Client-Side Code


GET http://server.com/Views/View.cshtml
View.cshtml
HTTP 403: Access Denied
GET http://server.com/Customers/Template/View.cshtml
Template Controller
HTTP 200: Ok!

Accessing Razor From Client-Side Code

- Feature folder convention
- Each folder relates to one feature
- We can leverage this convention!



```
js
  app
    authentication
    customer
    opportunity
    profile
    report
    risk
    utility
    app.js
```

Using Razor in AngularJS Templates

What We've Accomplished

```
public class CustomerViewModel
{
 public int Id { get; set; }
 public string Name { get; set; }
 public string WorkEmail { get; set; }
}
```

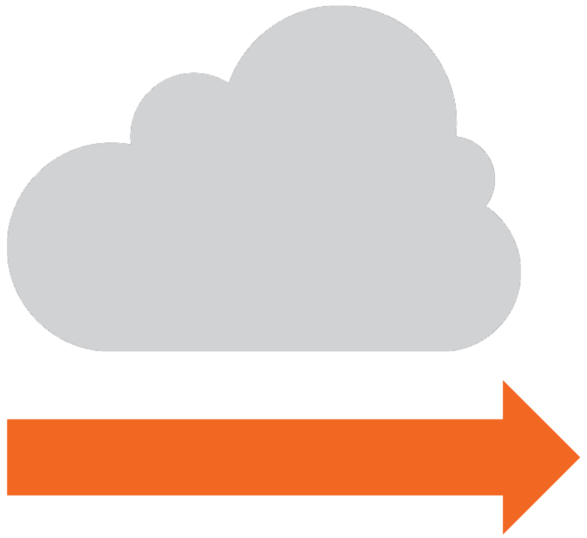
Our Action Result

```
{
 id: 5,
 name: ‘john smith’,
 workEmail: ‘test@user.com’
}
```

What We've Accomplished

Server-side:
Razor CSHTML

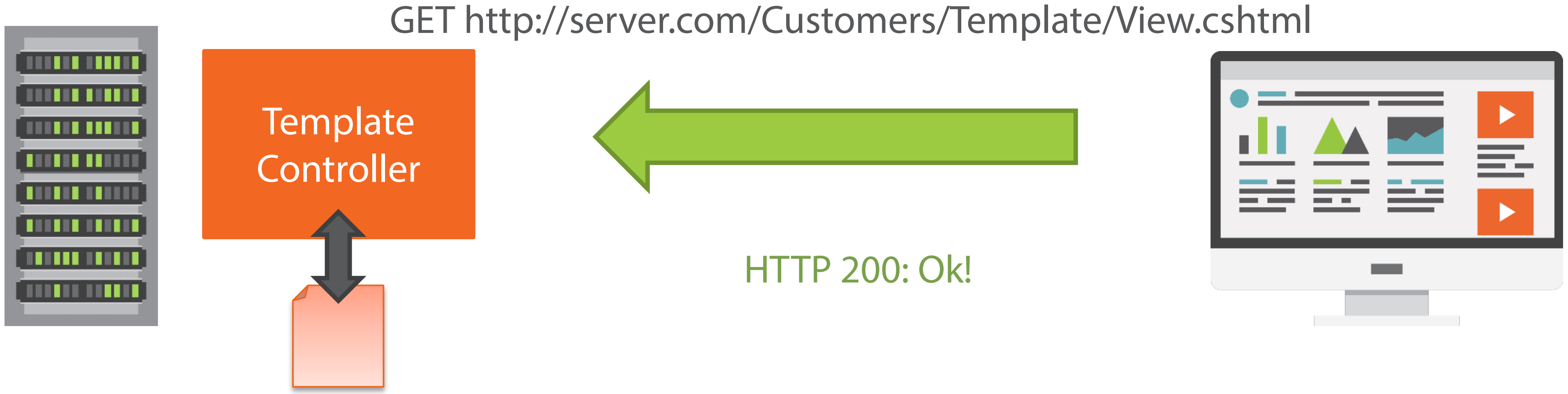
```
<script>
window.app.constant('model',
 @Html.JsonFor(Model));
</script>
```



Client-side:
HTML and JavaScript

```
<script>
window.app.constant('model',
{
"fullName":"Matt Honeycutt",
"emailAddress":
"matt.honeycutt@me.com"
});
</script>
```

What We've Accomplished



Up Next…

Strongly-typed AngularJS bindings!