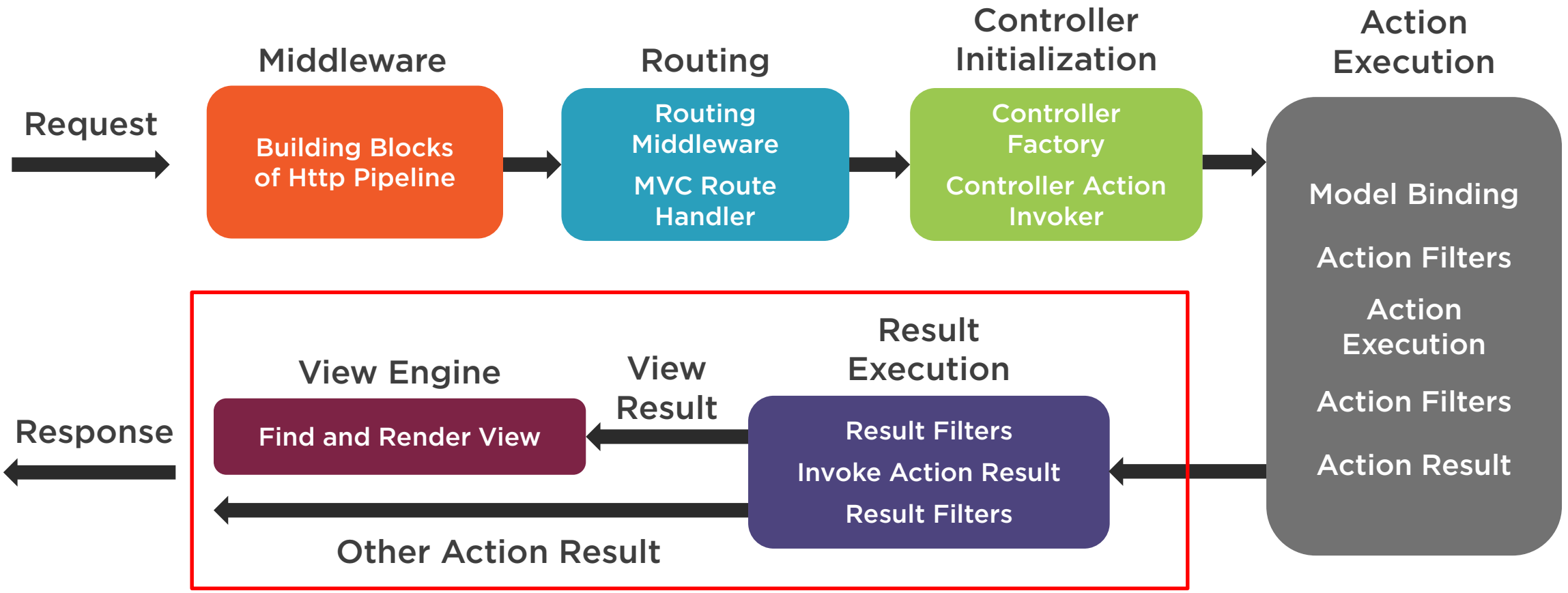
Exploring Action Results and the View Engine

Alex Wolf

www.alexwolfthoughts.com

The MVC Request Life Cycle



To-Do List

- Action Results and the Request Life Cycle
- Demo – Working with Result Filters
- Demo – Action Result Execution
- Understanding Content Negotiation
- Demo – Influencing Content Negotiation
- Exploring the View Rendering Process
- Demo – View Result Execution
- Demo – View Rendering

Action Results and the Request Life Cycle

```
public interface IActionResult
{
    Task ExecuteResultAsync(ActionContext context);
}
```

The IActionResult Interface

ExecuteResultAsync writes out the response for the request

Different Action Result implementations generate different responses

Types of Action Results

Content Results

JSON Results

View Results

File Results

```
public IActionResult Index()
{
    return View();
}

public IActionResult About()
{
    return Content("Hello world!");
}
```

- Returns a View Result

- Returns a Content Result

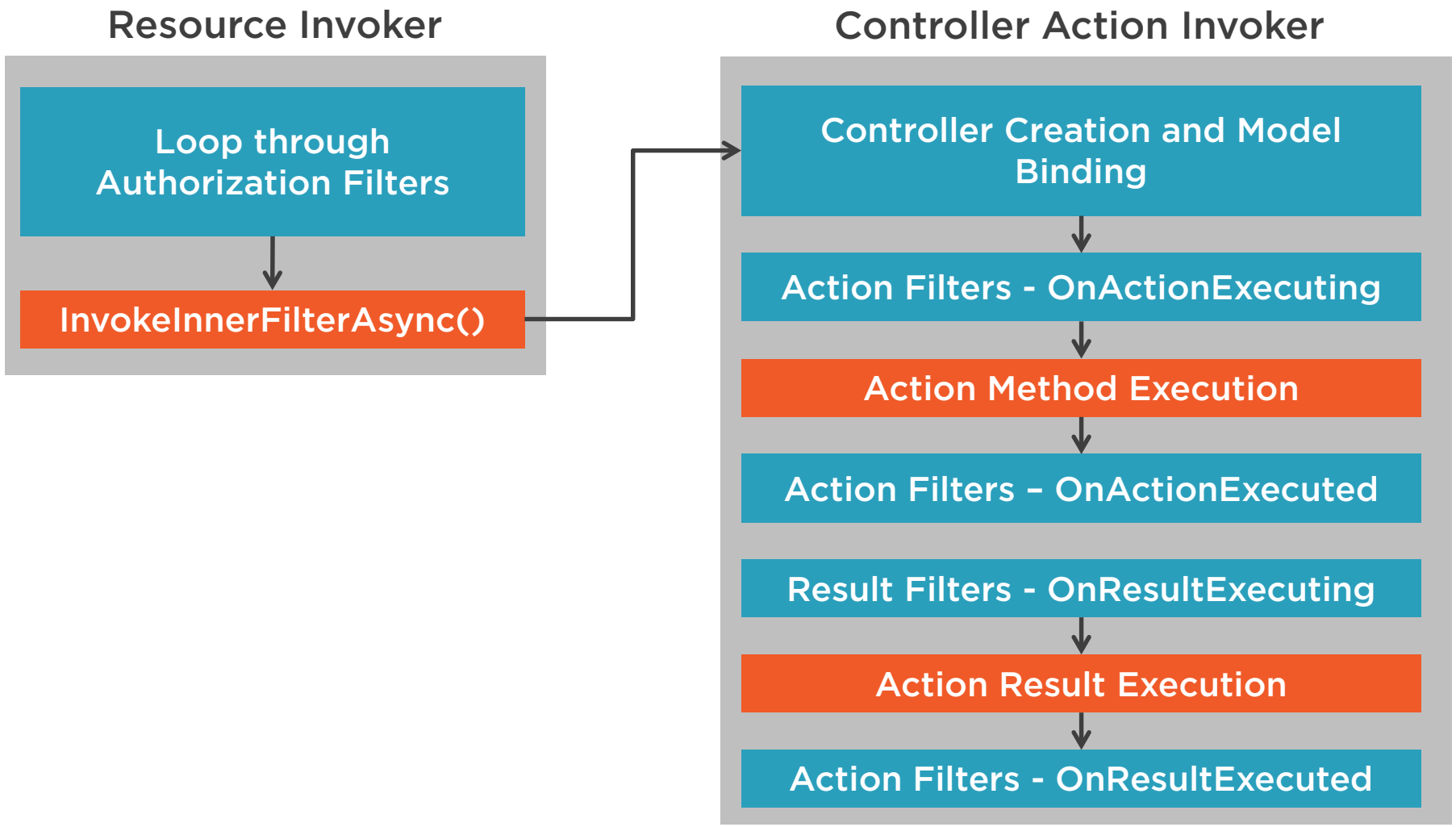
```
public interface IResultFilter : IFilterMetadata
{
    void OnResultExecuting(ResultExecutingContext context);
    void OnResultExecuted(ResultExecutedContext context);
}
```

The IResultFilter Interface

Runs logic before and after an Action Result executes

Can inject custom code late in the Life Cycle before the response is sent

The Filter Execution Pipeline



Understanding Content Negotiation

```
public IActionResult GetUser(int id)
{
    var user = userService.GetUser(id);
    return Ok(user);
}

public IActionResult EditUser()
{
    return Unauthorized();
}
```

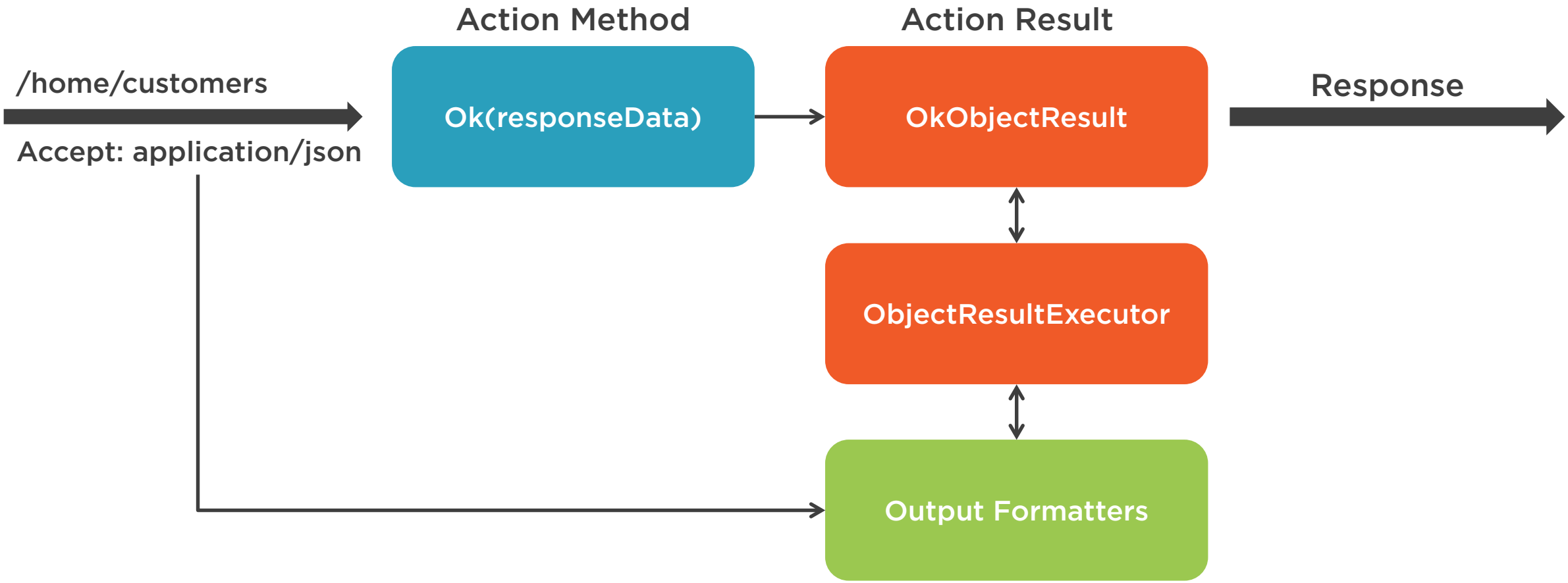
- Generates a 200 OK response with JSON data (by default)

- Generates a 401 Unauthorized response

Manging Status Codes and Response Data

Helper Method	Returned Status Code	Generated Action Result
Ok()	200	OkResult
Ok(responseData)	200	OkObjectResult
NotFound()	404	NotFoundResult
NotFound(responseData)	404	NotFoundObjectResult

The Content Negotiation Process



```
public interface IOutputFormatter
{
    bool CanWriteResult(OutputFormatterCanWriteContext context)
    Task WriteAsync(OutputFormatterWriteContext context);
}
```

The IOutputFormatter Interface

CanWriteResult determines if the formatter is valid for the request

WriteAsync writes out the formatted response

```
public void ConfigureServices(IServiceCollection services) {
    services.AddMvc(options =>  {
        options.RespectBrowserAcceptHeader = true;
        options.ReturnHttpNotAcceptable = true;
    }))
}
```

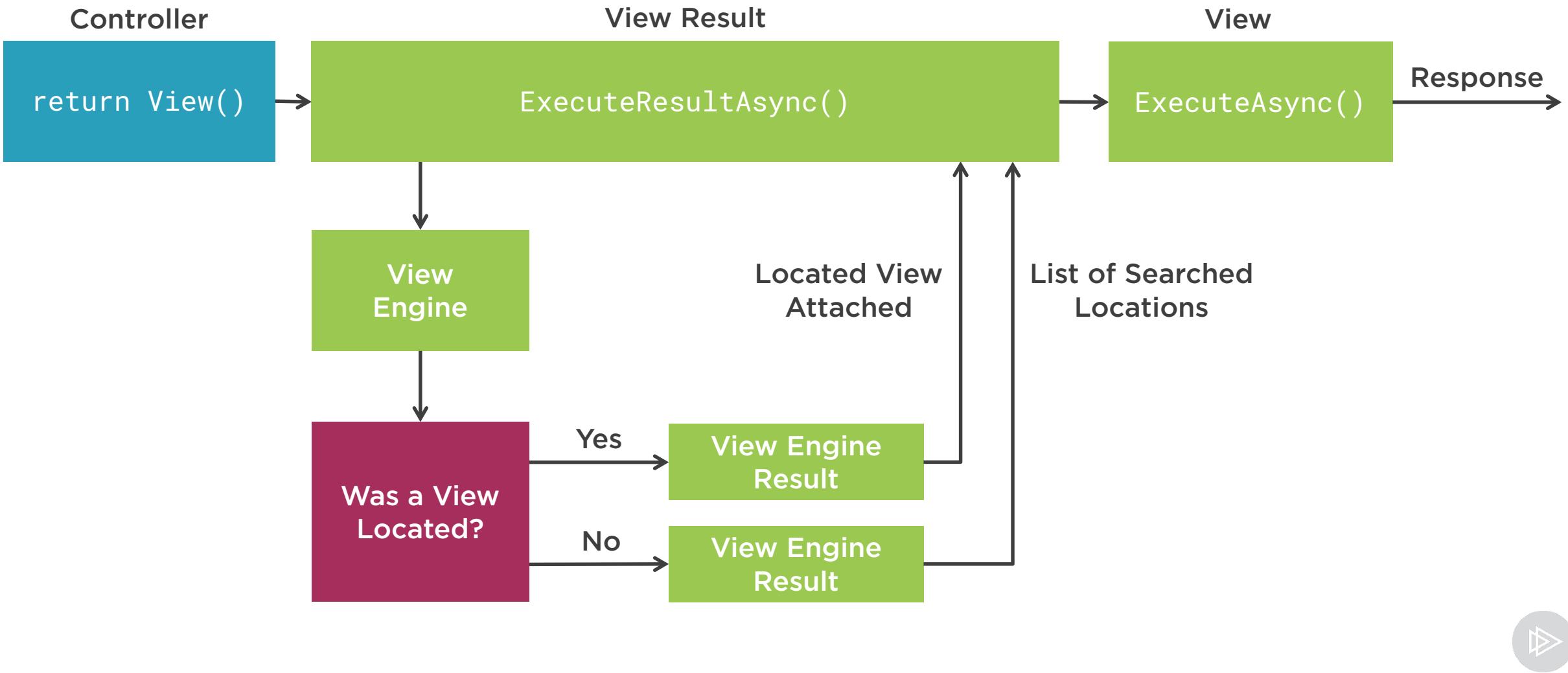
Content Negotiation Options

Browser content negotiation must be enabled using MVC options

Not Acceptable status codes can also be enabled for full negotiation

View Results and the View Engine

The View Rendering Process



```
public interface IViewEngine
{
    ViewEngineResult FindView(ActionContext context, string viewName,
    bool isMainPage);

    ViewEngineResult GetView(string executingFilePath, string viewPath,
    bool isMainPage);
}
```

The IViewEngine Interface

Every View Engine in MVC must implement this interface

Primary responsibility is to locate Views

```
public class ViewEngineResult

{

    public IEnumerable<string>
    SearchedLocations { get; }

    public bool Success { get; }

    public IView View { get; }

    public string ViewName { get; }



    public static ViewEngineResult Found
    (string viewName, IView view);

    public static ViewEngineResult  NotFound
    (string viewName, IEnumerable<string>
    searchedLocations);

}
```

- Properties used to communicate the search results

- Attaches the located View

- Attaches the list of searched locations

```
public interface IView
{
    string Path { get; }
    Task RenderAsync(ViewContext context);
}
```

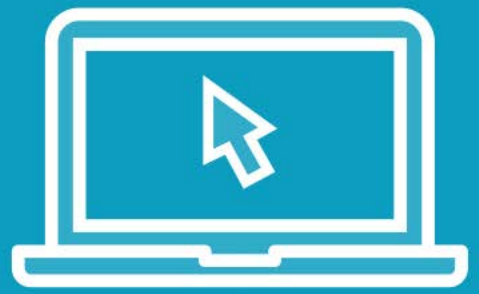
The IView Interface

Must be implemented by a View to render a response

```
@{
    ViewData["Title"] = "Home Page";
}
<div class="row">
    <div class="col-md-3">
     <h2>Sandbox</h2>
         <ul>
         <li>@Html.Action ("Home")</li>
        </ul>
    </div>
</div>
```

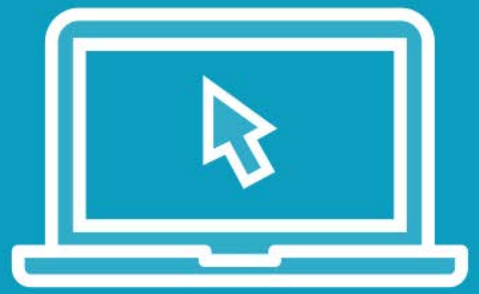
- C# code block
- Static HTML
- Razor helper method

Demo



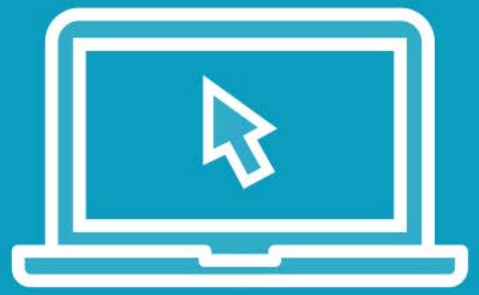
Working with Result Filters

Demo



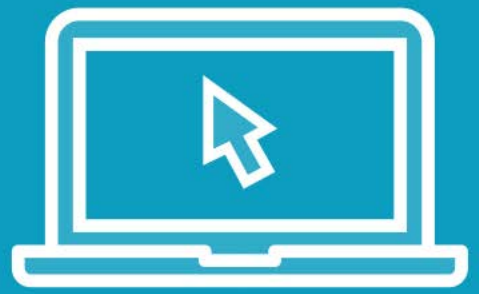
Action Result Execution

Demo



Content Negotiation in Action

Demo



View Rendering

Summary

- Result Filters inject logic before and after Action Result execution
- Action Results are executed by the Controller Action Invoker
- Content Negotiation can influence the formatting of the response data
- View Results trigger the Razor View Engine rendering process
- The View Engine locates and renders compiled Views as the response

Thank you, and good luck!