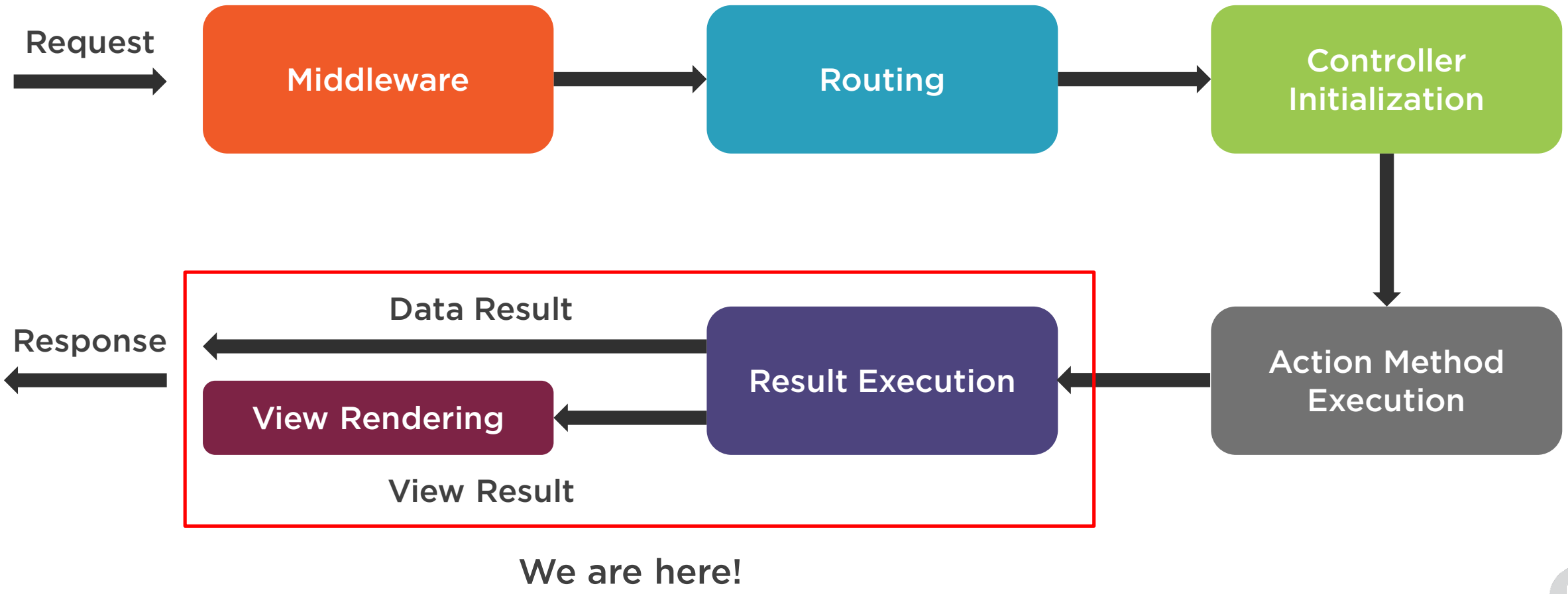
Exploring Action Results and the View Engine

Alex Wolf

www.alexwolfthoughts.com

The MVC Request Life Cycle



To-Do List

- Action Results and the Request Life Cycle
- A quick look at Result Filters
- Demo –Action Result execution
- Exploring the View rendering process
- Demo – The View Result execution process
- Demo – View compilation and rendering

Action Results and the Request Life Cycle

Action Results

Components that render the result of an Action Method to the response object.

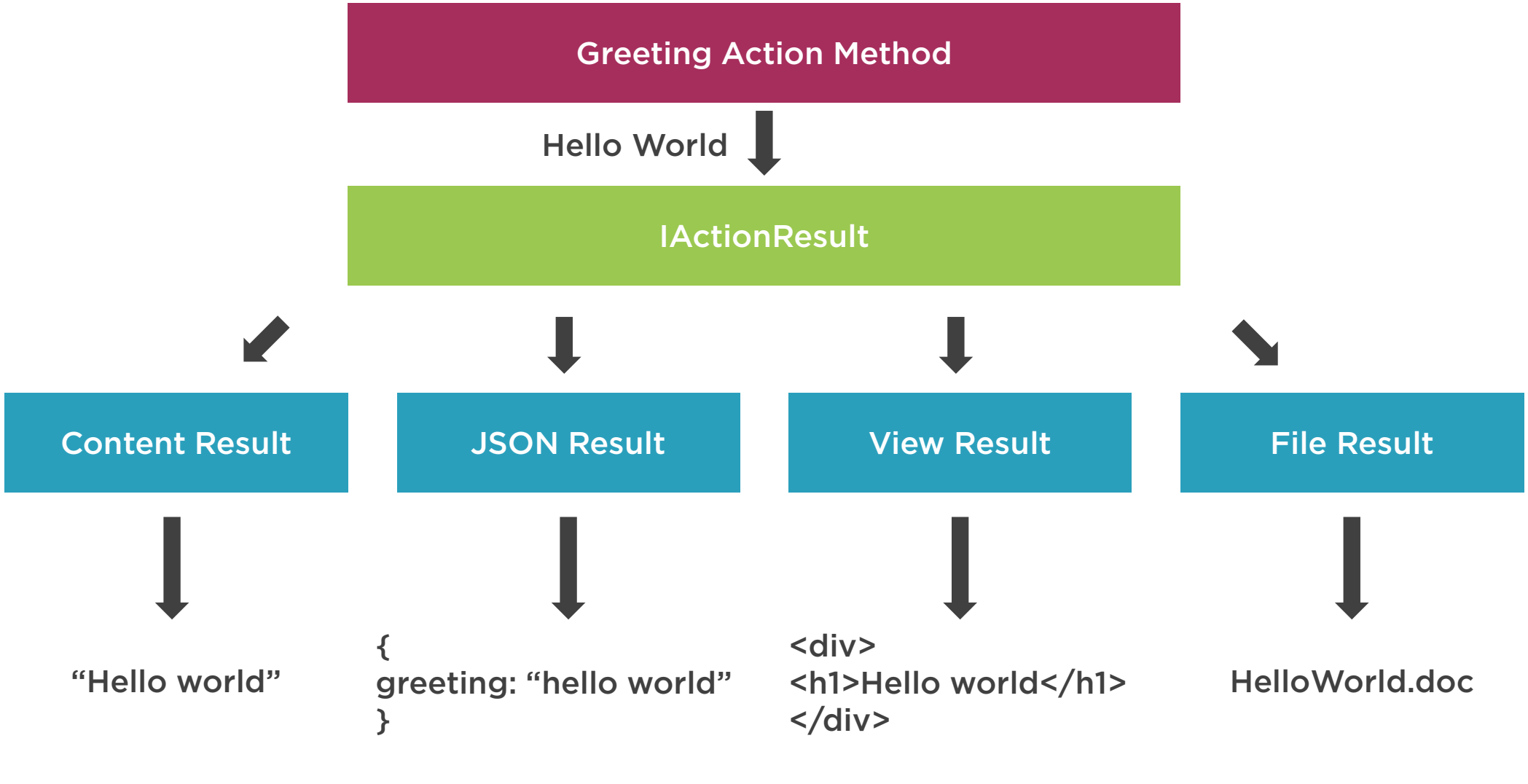
```
public interface IActionResult
{
    Task ExecuteResultAsync(ActionContext context);
}
```

The IActionResult Interface

ExecuteResultAsync renders out the response for the request

Different Action Result implementations generate different responses

Working with Different Action Results



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

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

- Returns a View Action Result

- Returns a Content Action Result

Understanding Result Filters

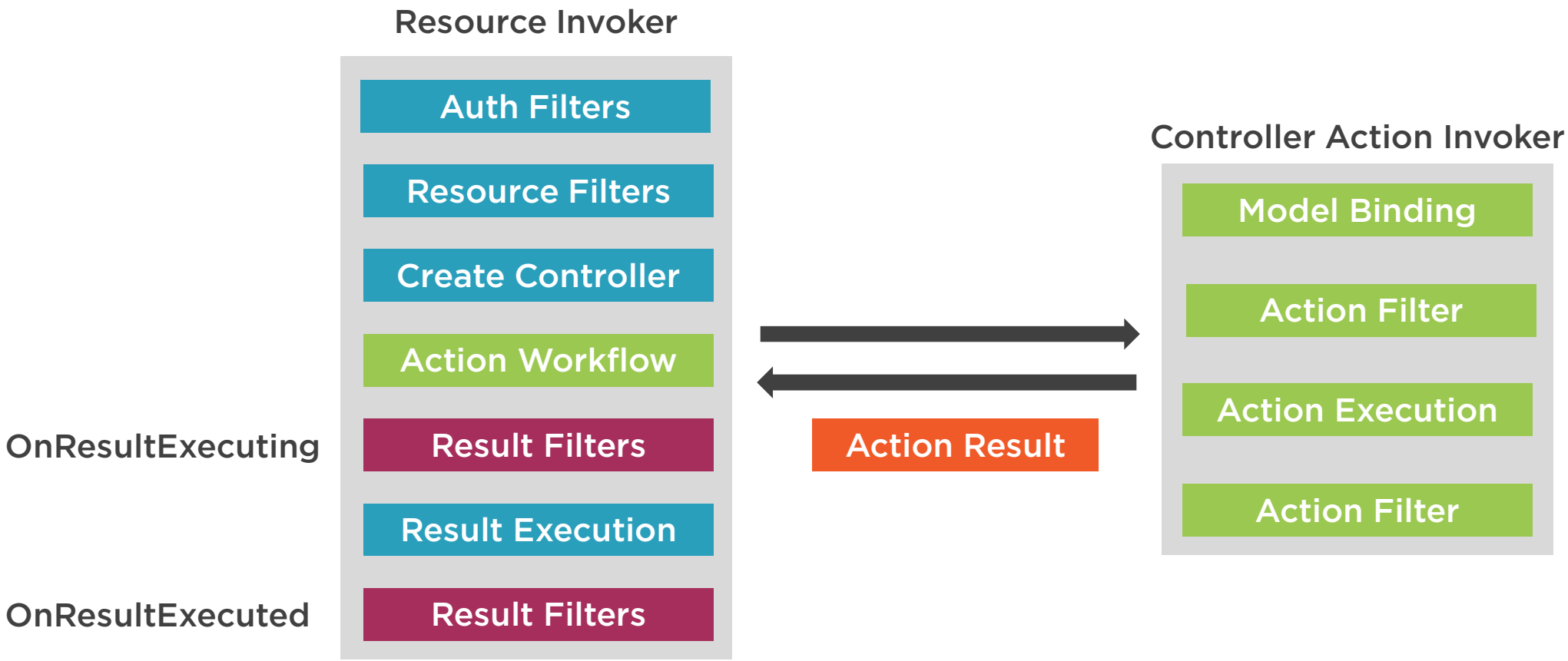
```
public interface IResultFilter : IFilterMetadata
{
    void OnResultExecuting(ResultExecutingContext context);
    void OnResultExecuted(ResultExecutedContext context);
}
```

The IResultFilter Interface

Runs logic before and after an Action Result executes

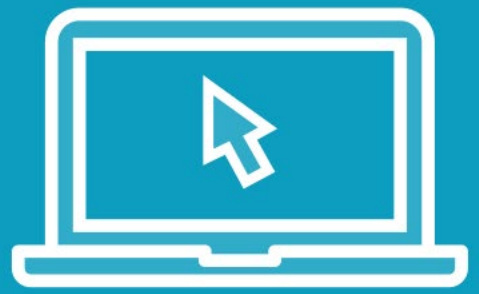
Provides a context object with application data

Understanding the Result Filter Workflow



A note about Result Filters.

Demo



Stepping through Action Result Execution

Exploring the View Rendering Process

Razor View Engine

The component responsible for locating and rendering the Views selected by an Action Method.

Core View Rendering Components

View Engine

Locates the View to be rendered

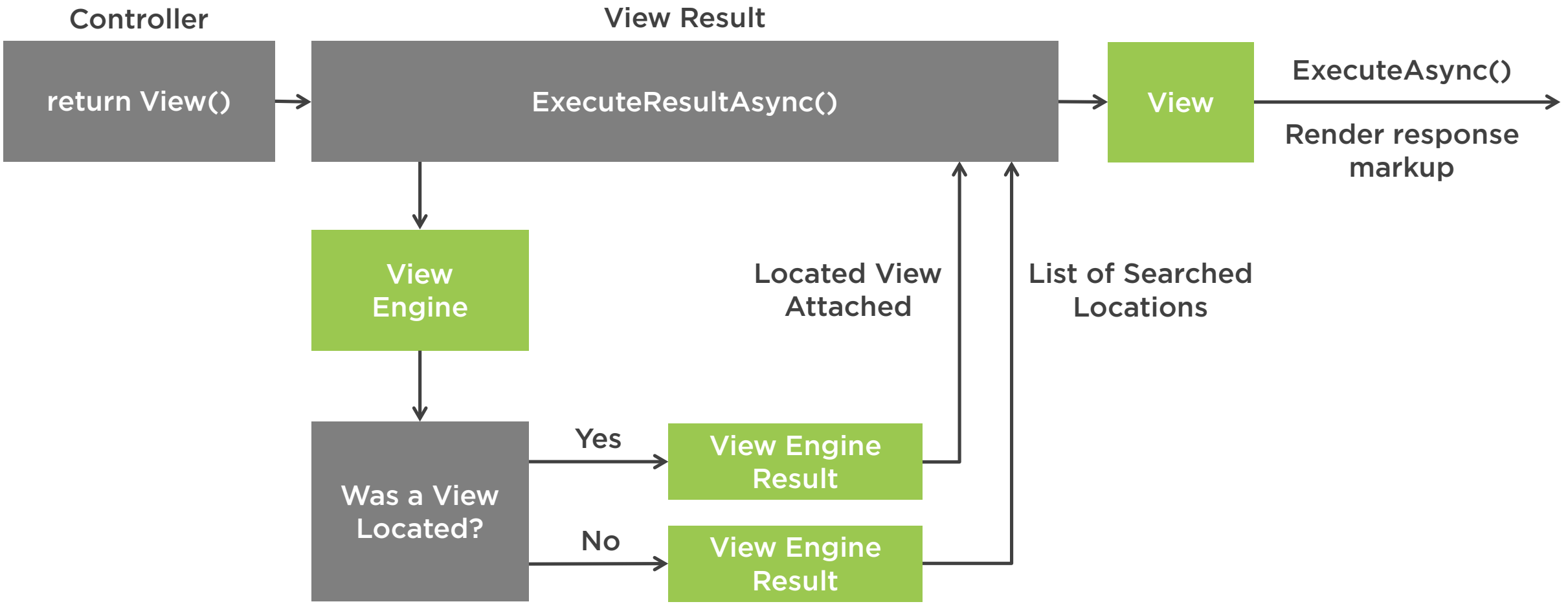
View Engine Result

Reports the results of the View Engine search

View

Renders the data and markup

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

FindView and GetView locate Views using different techniques

The returned View Engine Result is crucial to moving forward with View rendering

```
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

- Populates the list of searched locations

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

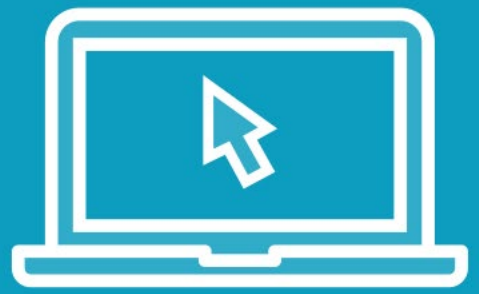
The IView Interface

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>
```

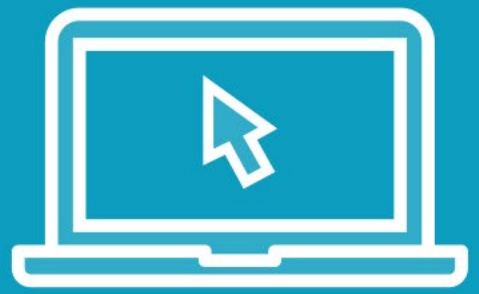
- C# code block
- Static HTML
- Razor helper method

Demo



Working with View Result Execution

Demo



Touring a Compiled Razor View

A note about Razor View compilation.

Summary

Action Results are created by Action Methods and executed by the Resource Invoker logic

Action Results render the response data back to the client in different formats

Result Filters inject logic before and after Action Result execution

View Results trigger the Razor View Engine rendering process

- The View Engine is responsible for locating a View to render
- The View Engine Result communicates the result of that search
- The View renders markup to the response

Thank you, and good luck!