

Creating an API and Returning Resources



KEVIN DOCKX

ARCHITECT

@KevinDockx <https://www.kevindockx.com>



Coming Up



ASP.NET Core MVC Middleware

Returning Resources

Interacting with our API

Configuring the Response



Middleware for Building an API

ASP.NET Web API

(http services)

ASP.NET MVC

(client-facing web applications)



ASP.NET Core MVC



Clarifying the MVC Pattern



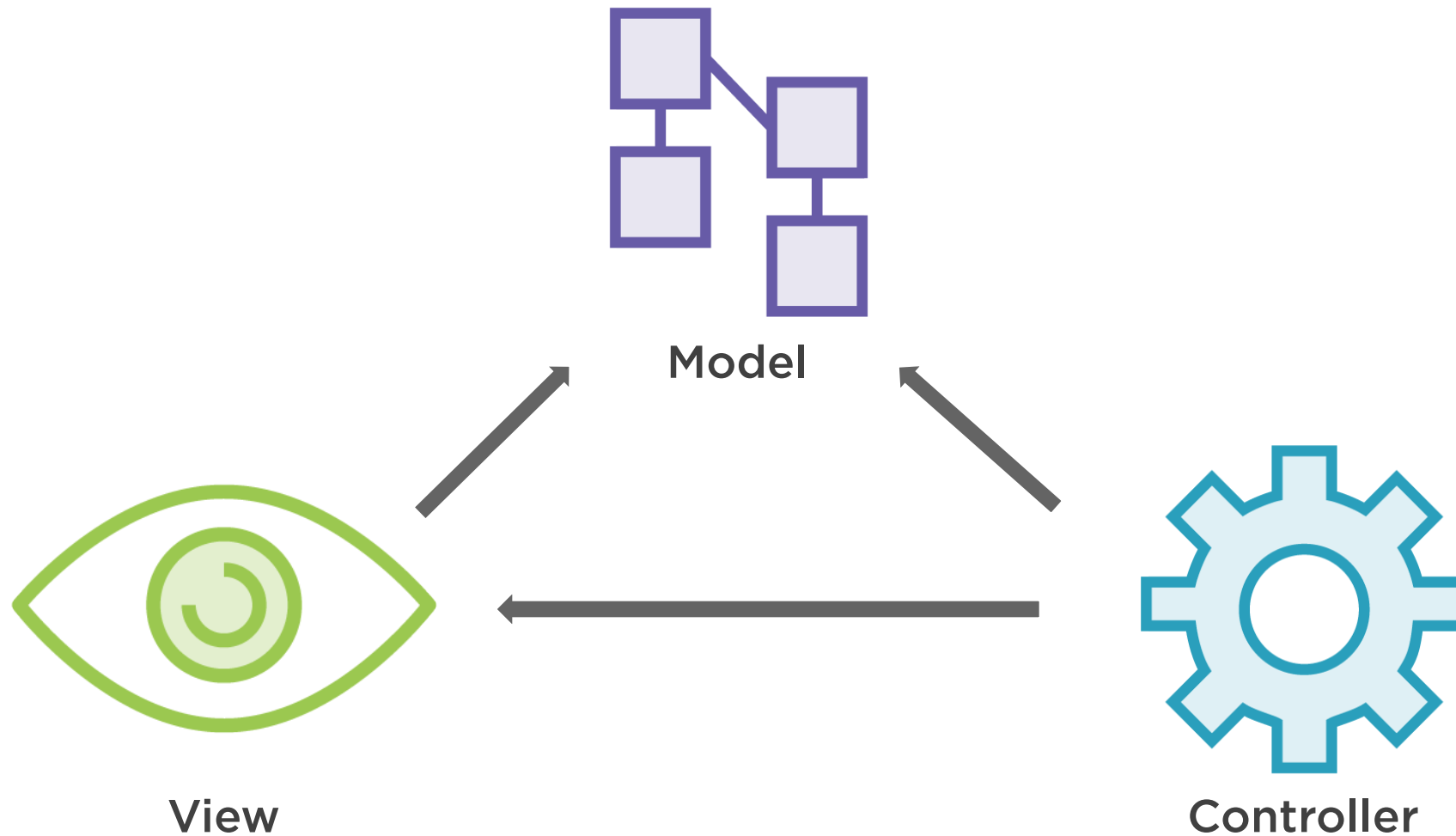
Model-View-Controller

Architectural pattern

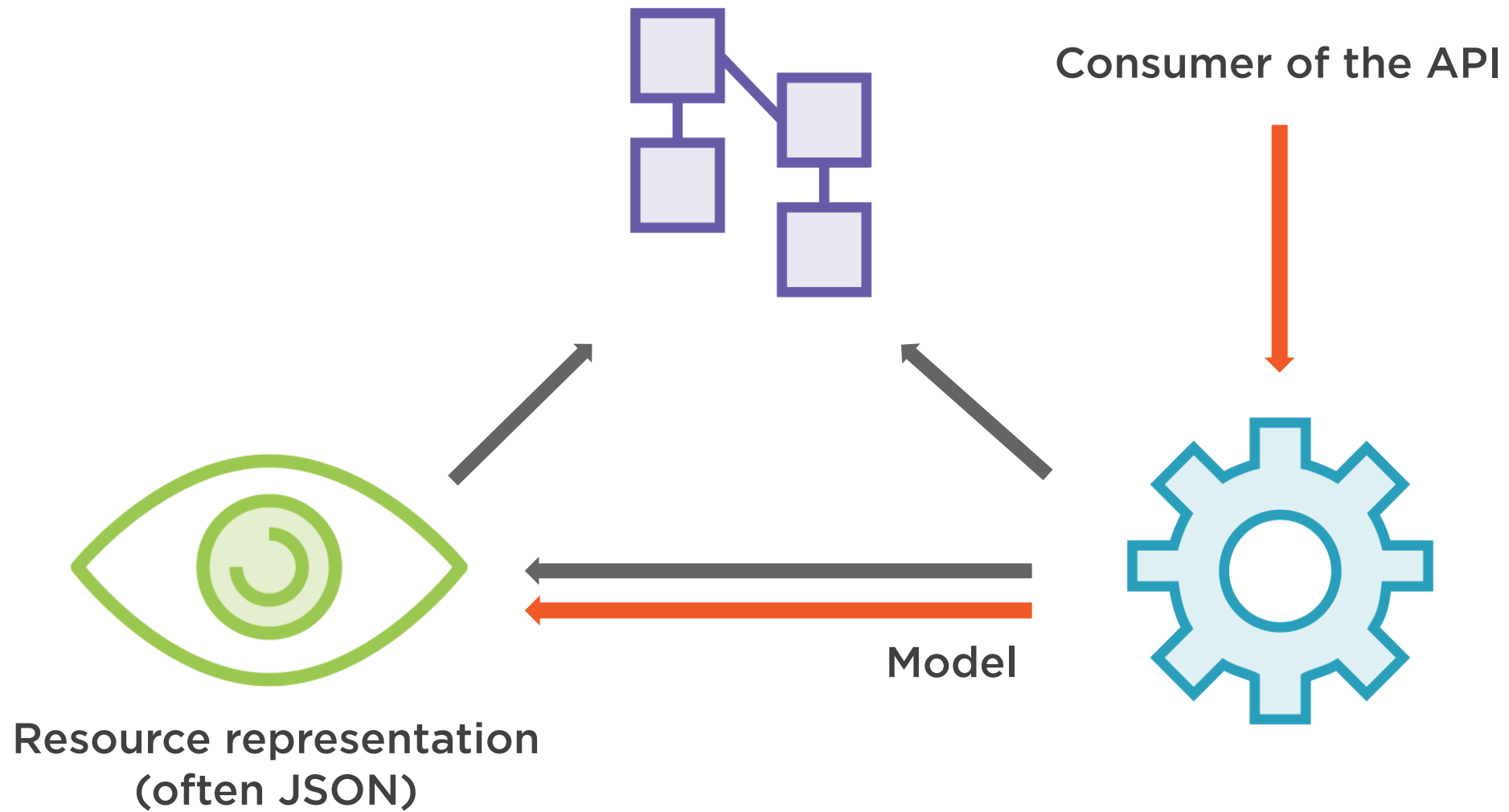
**Loose coupling, separation of concerns:
testability, reuse**

Not the full application architecture!

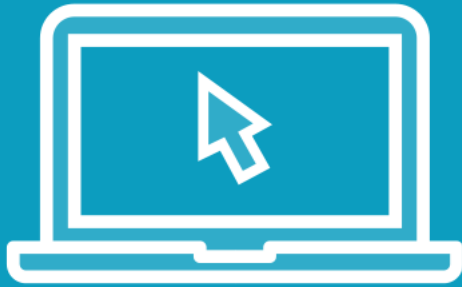
Clarifying the MVC Pattern



Clarifying the MVC Pattern



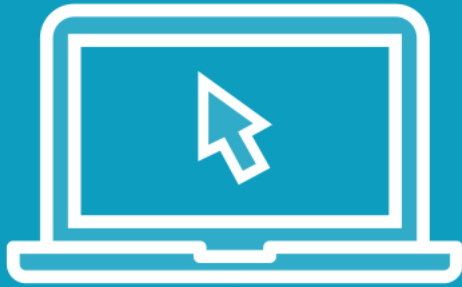
Demo



Adding the ASP.NET Core MVC Middleware



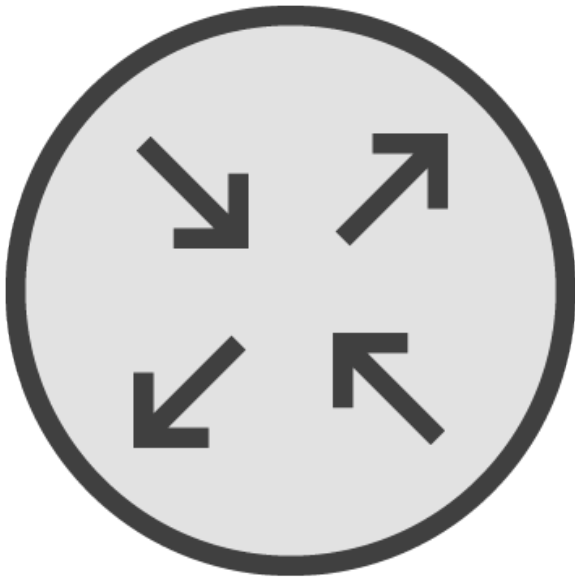
Demo



Returning Resources (part 1)



Learning About Routing



Matches request URI to controller method

Convention-based and attribute-based routing

```
app.UseMvc(config => {  
    config.MapRoute(  
        name: "Default",  
        template: "{controller}/{action}/{id?}",  
        defaults: new { controller="Home", action="Index" }  
    );  
});
```

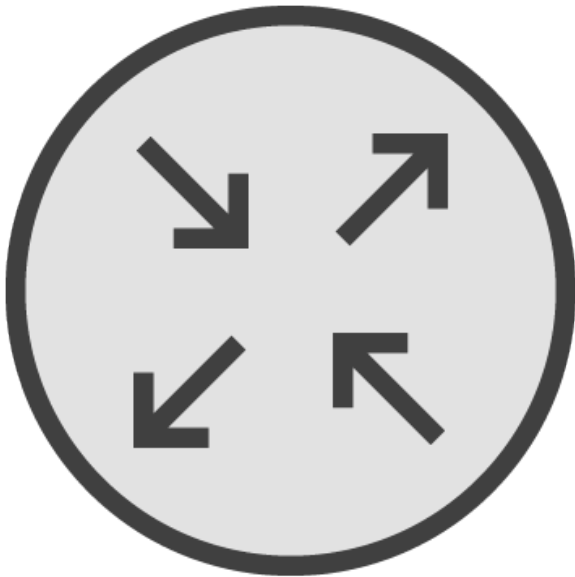
Convention-based Routing

Conventions need to be configured

Not advised for API's



Attribute-based Routing



**Attributes at controller & action level,
including an (optional) template**

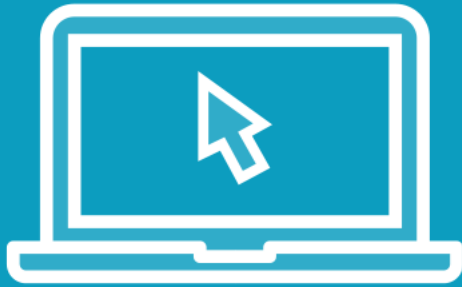
**URI is matched to a specific action on a
controller**

Routing

HTTP Method	Attribute	Level	Sample URI
-------------	-----------	-------	------------



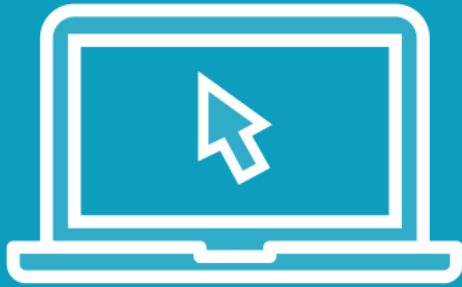
Demo



Returning Resources (part 2)



Demo



Improving the Architecture with Model Classes



The Importance of Status Codes



Part of the response

Provide information on

- Whether or not the request worked out as expected
- What is responsible for a failed request

The Importance of Status Codes

Level 200 Success

200 – OK

201 – Created

204 – No Content

Level 400 Client Error

400 – Bad Request

401 – Unauthorized

403 – Forbidden

404 – Not Found

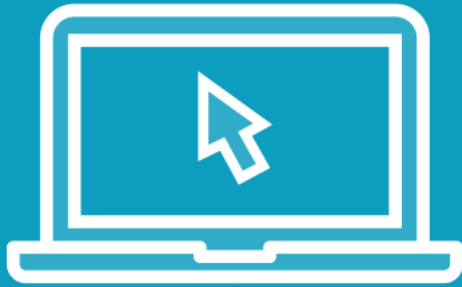
409 – Conflict

Level 500 Server Error

500 – Internal
Server Error



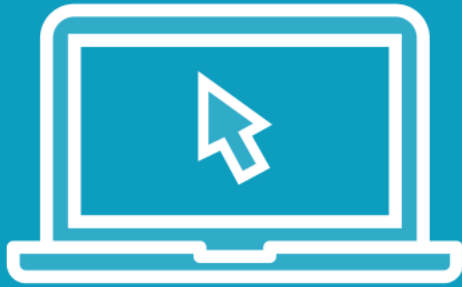
Demo



Returning Correct Status Codes



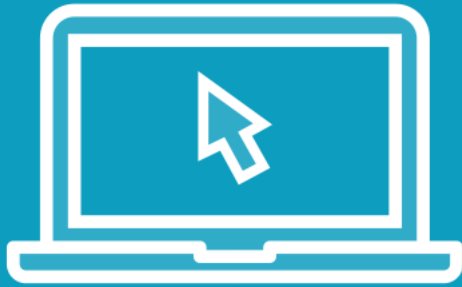
Demo



Returning Child Resources



Demo



Working with Serializer Settings



Formatters and Content Negotiation



Selecting the best representation for a given response when there are multiple representations available

Media type is passed via the Accept header of the request

- application/json
- application/xml
- ...

Formatters and Content Negotiation



Output formatter

Deals with output

Media type: accept header



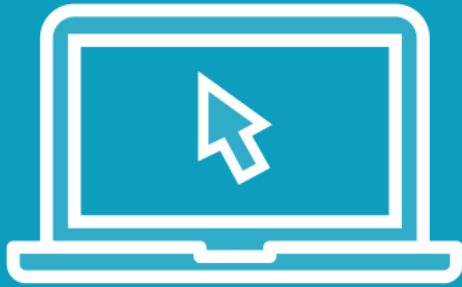
Input formatter

Deals with input

Media type: content-type header



Demo



Formatters and Content Negotiation



Summary



Model-View-Controller

- Model: application data logic
- View: display data
- Controller: interaction between View and Model
- More reuse, better testability

Routing: maps URI to controller method

Summary



Content negotiation: selecting the best representation for a given response

- Output formatters (accept header)
- Input formatters (content-type header)