Generating Responses with IResult (一)

- What is the return type of the following endpoint handlers?

```
void DeleteFruit(string id)
{    Fruit.All.Remove(id);       }
public void ReplaceFruit(string id, Fruit fruit)
{    Fruit.All[id] = fruit;      }
public static void AddFruit(string id, Fruit fruit)
{    Fruit.All.Add(id, fruit);   }
record Fruit(string Name, int Stock)
{    public static readonly Dictionary<string, Fruit> All = new();};
```

41

Generating Responses with IResult (二)

- What happens when sending a request to /fruit/apple before you create a fruit with the id apple?
  - You'll get an exception.

- Throwing an exception whenever a user requests an id that doesn't exist clearly makes for a poor experience all round.
- The most declarative way to do this with minimal APIs is to return an IResult instance.

42

Generating Responses with IResult (三)

- The endpoint middleware handles each return type as below:

void	Returns a 200 response with no body
string	Returns a 200 response with the string serialized to the body as text/plain.
IResult	Executes the IResult.ExecuteAsync method. Depending on the implementation, **this type can customize the response, returning any status code**.
T	Serialized to JSON and returned in the body of a 200 response as application/json

43

Generating Responses with IResult (四)

- A well-designed API uses status codes to:
  - Indicate to a client what went wrong when a request failed.
  - Provide more descriptive codes when a request is successful.
- Helper types Results and TypedResults can create a response with common status codes, optionally including a JSON body.

```
app.MapGet("/fruit/{id}", (string id) =>
    _fruit.TryGetValue(id, out var fruit)
        ? TypedResults.Ok(fruit)
        : Results.NotFound());
```

44

3. Routing

What is Routing? (一)

- One crucial aspect of minimal APIs that we touched on only lightly is how ASP.NET Core selects a specific endpoint from all the handlers defined, based on the incoming request URL.
- This process is called routing.

46

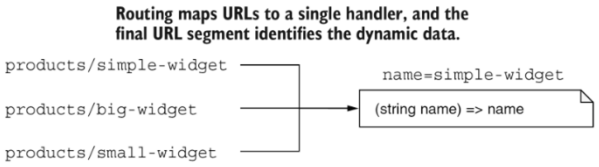
What is Routing? (二)

- Routing is the process of mapping an incoming request to a method that will handle it.
- You can use routing to:
  - Control the URLs exposed in the application;
  - Enable powerful features such as mapping multiple URLs to the same handler;
  - Automatically extracting data from a request's URL.

47

What is Routing? (三)

- Consider an e-commerce application that sells multiple products. Each product needs to have its own URL.



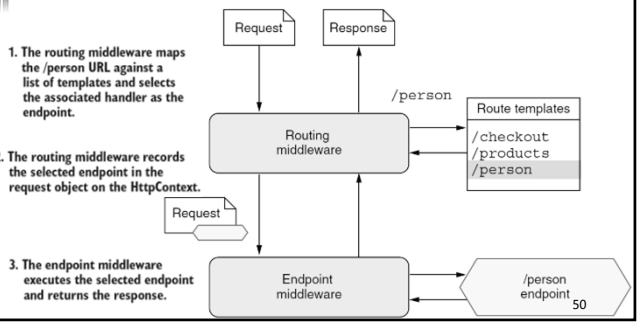
48

Endpoint Routing (一)

- Endpoint routing is the routing system introduced by ASP.NET Core 3.
- It's implemented with two pieces of middleware:
  - The **RoutingMiddleware** chooses which registered endpoints execute for a given request at runtime.
  - The **EndpointMiddleware** which is placed at the end of pipeline executes the endpoint selected by the RoutingMiddleware.

49

Endpoint Routing (二)



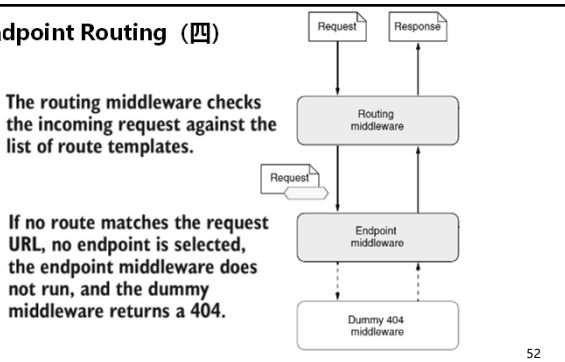
50

Endpoint Routing (三)

- What happens if the request URL doesn't match a route template?
  - The RoutingMiddleware doesn't select an endpoint.
  - As no endpoint is selected, the EndpointMiddleware silently ignores the request and passes it to the next middleware in the pipeline.
  - the "next" middleware is normally the dummy middleware that always returns a 404 Not Found response.

51

Endpoint Routing (四)



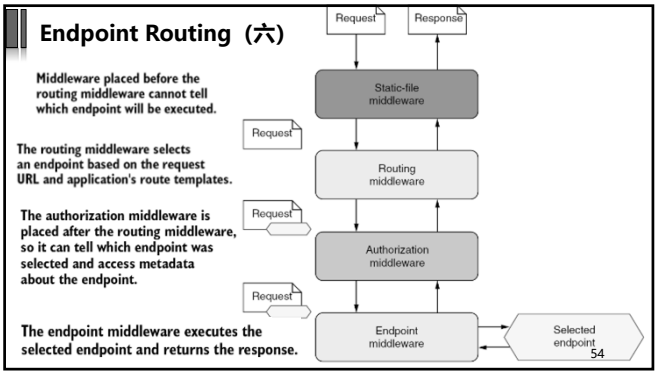
52

Endpoint Routing（五）

- Why we need two separate pieces of middleware to handle the routing process?
  - Only middleware placed after the RoutingMiddleware can see which endpoint is going to be executed.

53

Endpoint Routing（六）



54

Endpoint Routing（七）

- How to define an endpoint?
  - An endpoint in ASP.NET Core is a handler that returns a response.
  - Endpoints are registered by calling Map* functions.

```
WebApplication app = builder.Build();
app.MapGet("/test", () => "Hello world!");
```

Registers a minimal API endpoint that returns "Hello World!" at the route /test

55

Endpoint Routing（八）

- How to define an endpoint?
  - Each endpoint is associated with a route template that defines which URLs the endpoint should match.
  - A route template is a URL pattern that is used to match against request URLs.

```
WebApplication app = builder.Build();
app.MapGet("/test", () => "Hello world!");
```

Registers a minimal API endpoint that returns "Hello World!" at the route /test

56

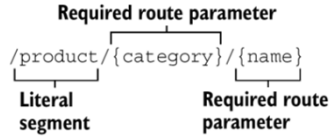
Exploring the Route Template Syntax（一）

- The routing middleware parses a route template by splitting it into segments.
- A segment is typically separated by the / character.

```
/product/{category}/{name}
```

57

Exploring the Route Template Syntax（二）



- Each segment is either:
  - A literal value
    - Literal segments in ASP.NET Core aren't case-sensitive
  - A route parameter
    - Route parameters are sections of a URL that may vary
- The request URL must match literal values exactly (ignoring case).

58

Route Template Example

Exercise

- Consider this route template, "/About/Contact", which of the following URL will match?
  - /about
  - /about-us/contact
  - /about/contact/email
  - /about/contact-us

59

Route Template Example

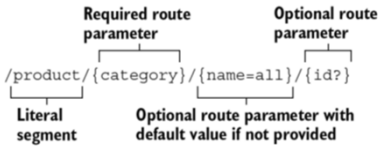
Exercise

- Consider this route template, "/{category}/{name}", which of the following URL will match?
  - /bags/rucksack-a
  - /shoes/black-size9
  - /socks/
  - /trousers/mens/formal

60

Using Optional and Default Values

- {id?} defines an optional route parameter called id. This segment of the URL is optional.
- Using default values allows multiple ways to call the same URL.
  - /product/shoes
  - /product/shoes/all



61

Route Values

- Route values are the values extracted from a URL based on a given route template.
- Each route parameter in a template has an associated route value.

62

Route Values

Exercise

- Suppose the route template is /product/{category}/{name=all}/{id?}
- Complete the route values

URL	Route values
/product/shoes/formal/3	category=shoes, name=formal, id=3
/product/shoes/formal	category=shoes, name=formal
/product/shoes	category=shoes, name=all
/product/computers/laptops/ABC-123	category=computers, name=laptops, id=ABC-123

63

4. Model Binding

Model Binding (一)

- In previous section, you've learned route parameters can be extracted from the request's path and used to execute minimal API handlers.

```
app.MapPost("/square/{num}", (int num) => num * num);
```

- Let's look in more detail at the process of extracting route parameters and the concept of model binding.

65

Model Binding (二)

- Endpoint handlers are normal C# methods, so the ASP.NET Core framework needs to be able to call them in the usual way.
- When handlers accept parameters as part of their method signature, the framework needs a way to generate those objects.
- Where do they come from, and how are they created?
  - These values come from the request itself
  - ASP.NET Core turn that into a .NET object

```
app.MapPost("/square/{num}", (int num) => num * num);
```

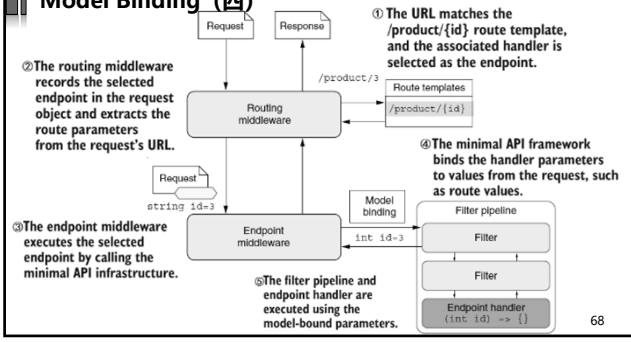
66

Model Binding (三)

- Model binding extracts values from a request and uses them to create .NET objects.
- These objects are passed as method parameters to the endpoint handler being executed.
- Model binding happens before the endpoint handler execute in the EndpointMiddleware.

67

Model Binding (四)



68

Binding Source of the Parameters of Endpoint Handlers

- Minimal APIs can use six different binding sources to create the handler arguments:
  - Route values
  - Query string values
  - Header values
  - Body JSON
  - Dependency injected services
  - Custom binding

69