ASP.NET Core Web API 5

Shailendra Chauhan

Microsoft MVP, Consultant and Corporate Trainer

DotNetTricks

Agenda

- REST Architecture
- ASP.NET Core Web API
- Web API2 vs. ASP.NET Core Web API
- Creating ASP.NET Core Web API
- Http Methods
- Parameter Binding
- Content Negotiation
- Versioning

REST (REpresentational State Transfer) Architecture

- REST is an architectural style (concept) to describe a hypermedia distributed system, such as the Web. REST is based upon the following principles
  - **Addressable Resources** - Each resource should be identified by a URI.
  - **Simple and Uniform Interfaces** - Keep your interfaces simple and uniform. This can be achieved by using the HTTP/HTTPS protocol uniform methods.
  - **Representation Oriented -** Representation of resources are exchanged. Representation may be in many formats like XML, JSON etc.
  - **Communicate Stateless** - Any session specific data should be held and maintained by the client and transferred to the server with each request.
  - **Cacheable** - Clients should be able to cache the responses for further use.

ASP.NET Core Web API

- Based on RESTful Architecture.
- Supports ASP.NET Core MVC features.
- Provides productivity features for writing APIs:
  - Model binding Convert incoming request objects into models
  - Model validation Makes sure the bound models are valid
  - Formatters Read and write objects from/to the request/response
  - Filters Run custom logic on code that runs before/after application logic
  - Content negotiation
- OpenAPI Support Using Swagger

Web API2 vs. ASP.NET Core Web API

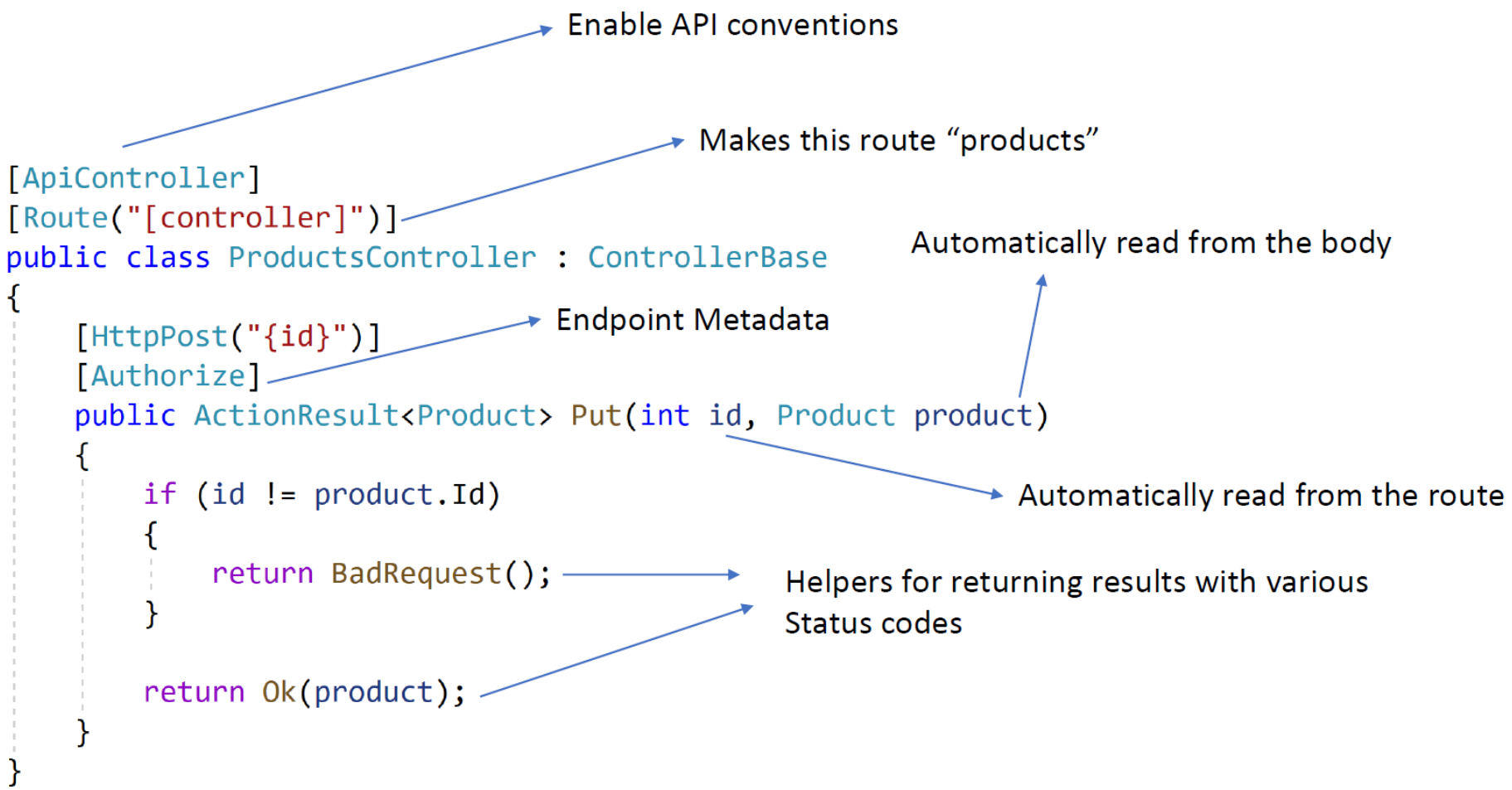
- An ideal platform for building RESTful services.
- Modules and handlers based pipeline.
- Supports development only by using Windows OS.
- Supports deployment to Windows, and windows based containers.
- Supports IIS and Self-Hosting.

- ASP.NET Core is built for RESTful services with high performance and scalability.
- Much leaner middleware based pipeline.
- Supports cross-platform development using Windows, Mac, Linux.
- Supports deployment to Windows, Linux and containers.
- Support hosting using various web server like IIS, Kestrel, Apache Tomcat, Nginx, self-hosted, etc.

DEMO:

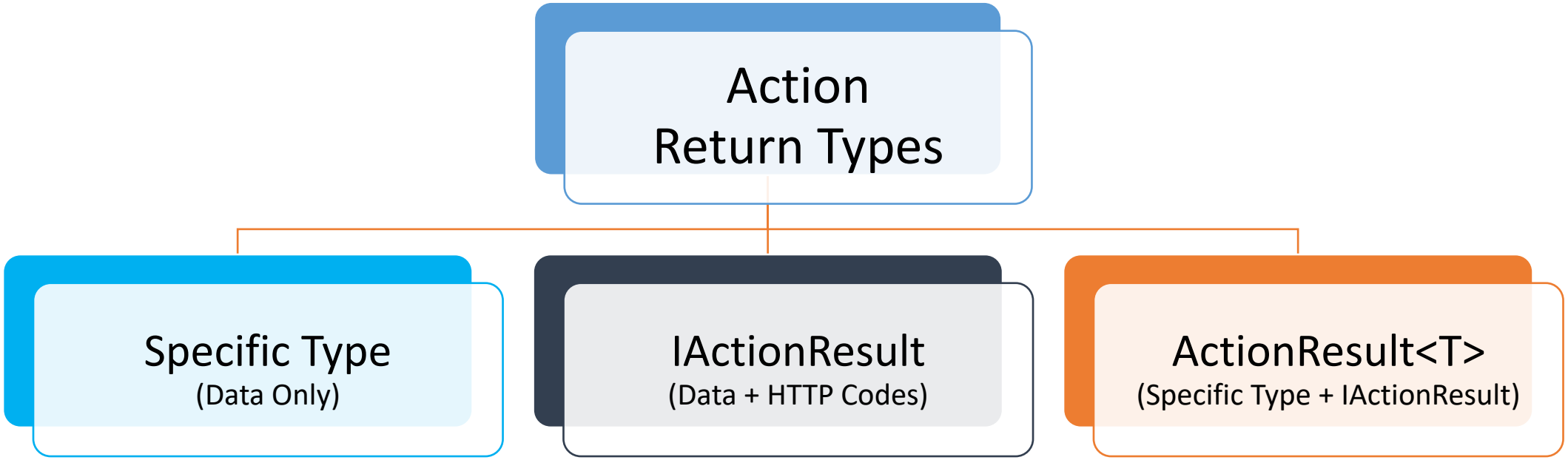
Creating ASP.NET Core Web API

ASP.NET Core Web API Conventions



DotNetTricks

Web API Action Return Types



Web API Action Results

Web API Action Results

Status Code Results
(Return Status Code)

- OkResult
- CreatedResult
- NoContentResult
- BadRequestResult
- UnauthorizedResult
- NotFoundResult
- UnsupportedMediaTypeResult
- StatusCode (Others)

Object Results
(Return Status Code + Data)

- OkObjectResult
- CreateObjectResult
- BadRequestObjectResult
- NotFoundObjectRequest
- ObjectResult
- StatusCode

DotNetTricks

DEMO:

Testing Web API Using Postman

DotNetTricks

DEMO:

Consuming Web API in ASP.NET Core

DotNetTricks

DEMO:

CRUD Operations using ASP.NET Core

Http Methods

- **GET** : Retrieve data/resource from the server.
- **HEAD**: Identical to *GET*, without the response body.
- POST: Send data to the server to create a resource.
- **PUT**: Send data to the server to update or replace a resource.
- **PATCH**: Send only changed data to server to partial update or modify a resource.
- **DELETE**: Delete the resource at the specified URI.
- **Idempotent Methods (GET, HEAD, PUT and DELETE):** The term idempotent describe an operation that will produce the same results if executed once or multiple times.

Parameter Binding

- A process to get the value from the request *URI* or *body* and map with action receiving parameters.
- Receiving parameters can be primitive type or complex type
- Web API support parameter binding or model binding by using *HttpParameterBinding* class

Parameter Binding Rules

- By default, primitive type parameters (value types, strings, Guids, DateTimes and so on) read from URI
- By default, complex types or collection read from the body
- A single model can't be composed based on input from both URI and request body

HTTP Method	Request URI (Query String)	Request Body
GET	Primitive Type, Complex Type	NA
POST	Primitive Type	Complex Type
PUT	Primitive Type	Complex Type
PATCH	Primitive Type	Complex Type
DELETE	Primitive Type, Complex Type	NA

Content Negotiation

- A process of selecting the best representation for a response when multiple representations are available.
- JSON is default representation.
- Content negotiation is handled by **Accept** header with values as *"application/json," "application/xml,"* or a custom media type.
- Enable XML serialization:

```
services.AddControllers().AddXmlSerializerFormatters()
                .AddXmlDataContractSerializerFormatters();
```

Specify a format

- [Produces] filter is used to restrict the response formats i.e. forces all actions within the controller to return specified-format responses.
- [Produces] can be applied at the action, controller, or global scope.

```
[ApiController]
[Route("[controller]")]
[Produces("application/json")]
public class UserController : ControllerBase
{

}
```

Versioning

- Url
  - /api/v2/users/
- Query string
  - /api/users?api-version=2
- Custom request header
  - api-version: 2
- Accept header
  - Accept: application/json;v=2