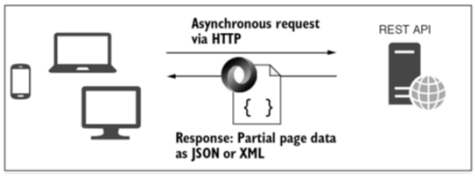
2. Minimal APIs

What is an HTTP API (Web API)? (一)

- An HTTP API exposes multiple URLs via HTTP that can be used to access or change data on a server.
- It typically returns data using the JSON format..



29

What is an HTTP API (Web API)? (二)

- When should you develop an HTTP API application?
  - Client-side SPA (single-page application)
  - Mobile apps
  - Other back-end service

30

Case 1: The SPA

- Client-side SPAs have become popular in recent years with the development of frameworks such as ***Angular, React, and Vue***.
- These frameworks typically use ***JavaScript*** running in a web browser to generate the HTML that users see and interact with.
- Once the SPA is loaded in the browser, communication with a server still occurs over HTTP, but instead of sending HTML directly to the browser in response to requests, the ***server-side application*** sends data in JSON format to the ***client-side application***.

31

Case 2: The Mobile Application

- These days, mobile applications are common and, from the server application's point of view, similar to client-side SPAs.
- A mobile application typically communicates with a server application by using an HTTP API, receiving data in JSON format, just like an SPA. Then it modifies the application's UI depending on the data it receives.

32

Case 3: Other Back-end Service

- The HTTP API application is designed to be partially or solely consumed by other backend services.
- Imagine that you've built a web application to send emails.
- By creating an HTTP API, you can allow other application developers to use your email service by sending you an email address and a message.
- Virtually all languages and platforms have access to an HTTP library they could use to access your service from code.

33

Defining minimal API endpoints

CodeLab 2.2

```
app.MapGet("/", () => "Hello World!");
app.MapGet("/person", () => new Person("Lee", "Lee");
```

- These two APIs correspond to the paths / and /person, respectively.
- This basic functionality is useful, but typically you need some of your APIs to be more dynamic.
- This goal is achieved by using ***parameterized routes*** for API definitions.

Parameterized Routes (一)

- Create a parameter in a minimal API route using the expression {someValue}, where someValue is any name you choose.
- The value will be extracted from the request URL's path and can be used in the lambda function endpoint.
- Example: /person/{name}
  - /person/Lee
  - The name parameter will have the value "Lee".

Parameterized Routes (二)

```
WebApplicationBuilder builder = WebApplication.CreateBuilder(args);
WebApplication app = builder.Build();

var people = new List<Person>
{
    new("Tom", "Hanks"),
    new("Denzel", "Washington"),
    new("Leondardo", "DiCaprio"),
    new("Al", "Pacino"),
    new("Morgan", "Freeman"),
};

app.MapGet("/person/{name}", (string name) =>
    people.Where(p => p.FirstName.StartsWith(name)));

app.Run();
```


Mapping verbs to endpoints (一)

- The MapGet( ) function matches requests that use the GET HTTP verb.
- In theory, each of the HTTP verbs has a well-defined purpose:
  - GET: get data from the server
  - POST: send data or change data on the server
  - ……
- Define endpoints for other verbs by using the appropriate Map* functions.

Mapping verbs to endpoints (二)

Method	HTTP verb
MapGet(path, handler)	GET
MapPost(path, handler)	POST
MapPut(path, handler)	PUT
MapDelete(path, handler)	DELETE
MapPatch(path, handler)	PATCH
MapMethods(path, methods, handler)	Multiple verbs
Map(path, handler)	All verbs

Defining Route Handlers with Functions (一)

- Using a lambda function as the handler for an endpoint is often the simplest approach, but you can take many approaches.
- Approach 1: A lambda expression

```
record Fruit(string Name, int Stock)
{
    public static readonly Dictionary<string, Fruit> All =
 new();
};
app.MapGet("/fruit", () => Fruit.All);
```

Defining Route Handlers with Functions (二)

■Approach 2: A function

```
void DeleteFruit(string id)
{    Fruit.All.Remove(id);  }
app.MapDelete("/fruit/{id}", DeleteFruit);
```

```
class Handlers
{
    public static void AddFruit(string id, Fruit fruit)
    {        Fruit.All.Add(id, fruit);    }
}
app.MapPost("/fruit/{id}", Handlers.AddFruit);
```