</talentlabs>

Express Lecture 7

Restful API

</talentlabs>

Agenda

- HTML Response vs JSON Response
- Restful API Basic
- More Examples
- Real World Example

HTML Response vs JSON Response

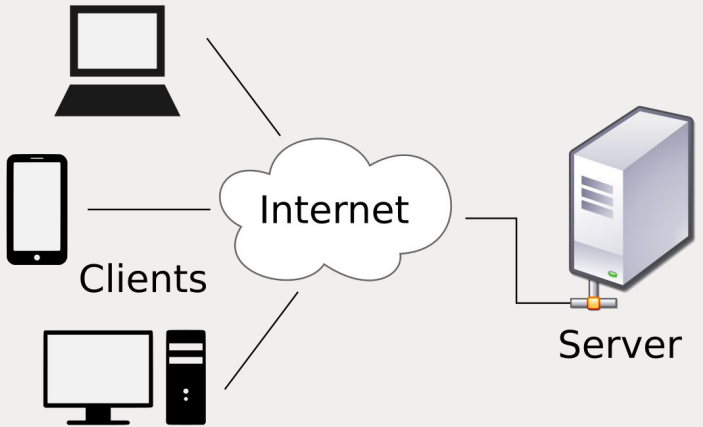
</talentlabs>

```
GET https://adventure-works.com/orders/1
```

```
{"orderId":1,"orderValue":99.90,"productId":1,"quantity":1}
```

Why JSON Response

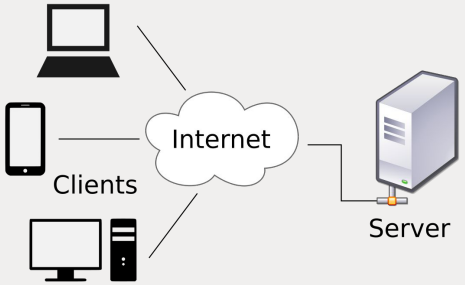
- Not all frontend applications can read HTML.
- Same set of APIs for
  - Browser
    - React
    - Pure JavaScript
  - Mobile Application
    - Android
    - IOS
  - IoT devices
  - etc.



Can the clients read HTML?
Most of the time, only browsers read HTML.

	Frontend Cost	Time Cost	Extensibility
HTML Server (Server-side Rendered)	The same developer can do both front and backend development.	Rendering HTML directly from Express is much easier than setting up both the APIs and a frontend app.	Only can serve a page to a web browsers.
JSON API Server (Client-side Rendered)	A Company usually need to hire **another frontend developer** for the frontend app.	Need extra time to setting up good and **reusable API standards. Communication between teams are needed.**	The same set of API can be used by different frontend applications

You can find out more discussion by google: “server side rendering vs client side rendering”



How to tell if a website Server-side rendered vs Client-side rendered?

- When you hit a link on a website, if it leads to a **page refresh** (go to the other page). The website has a high chance to be Server-side rendered.
  - Old Banking site
  - Old Government site
- If you see a website can switch to different views/contents without page refresh, the website has a high chance to be Client-side rendered.
  - Youtube
  - Facebook
- Actually some websites are hybrid as well, it is not a hard yes or no answer, you need to consider which method suit your problem better.

Restful API Basic

</talentlabs>

What is Restful API

- It is a popular guideline for API design
  - https://docs.microsoft.com/en-us/azure/architecture/best-practices/api-design
  - It has a strong opinion on using readable
    i. HTTP Path
    ii. HTTP Method
    iii. HTTP Status Code
- It is just a guideline, most people are not following it 100%.

Use meaningful HTTP Methods

HTTP Method	Description	Common
GET	**retrieves** a representation of the resource at the specified URI. The body of the response message contains the details of the requested resource	Very Common
POST	**creates** a new resource at the specified URI. The body of the request message provides the details of the new resource. Note that POST can also be used to trigger operations that don't actually create resources.	Very Common
PUT	either **creates or replaces** the resource at the specified URI. The body of the request message specifies the resource to be created or updated.	Common, but it can be replaced with POST
PATCH	performs a partial update of a resource. The request body specifies the set of changes to apply to the resource.	Can be replaced with POST
DELETE	**removes** the resource at the specified URI.	Very Common

These methods are not invented by Restful API, they are in the original HTTP protocol standards.

However before Restful API, people use GET or POST for all of their work.

Use meaningful HTTP Path

This one is easier, we have been doing like this.

- cats/1
- products/1

Restful API encourages us to have a meaningful HTTP Path design. Reader/User should know which resource it is about by looking at the path.

For example:

courses/abcd123: you know it is about a course with id/name abcd123
courses: you know it is about all or multiple courses

Meaningful HTTP Path and Method -> Meaningful Endpoint

A **combination of (Endpoint, HTTP method)** defines the API behaviour.

Endpoint	POST	GET	PUT	DELETE
/customers	**Bulk / Create** new customer(s)	**Retrieve** all customers	Bulk **update** of customers	**Remove** all customers
/customers/1 (/customers/:id)	**Create** a new customer with id = 1	**Retrieve** the details for customer 1	**Update** the details of customer 1 if it exists	**Remove** customer 1

Only an example, there is no 100% correct answer, the key here is to make them readable.

Meaningful HTTP Status code for response

https://en.wikipedia.org/wiki/List_of_HTTP_status_codes

- 200 OK
- 201 Created

Recall the HTTP status code in a HTTP Response. There is a list of HTTP status code. Restful API encourage us to pick a meaningful status code for a response.

More Examples

</talentlabs>

GET methods

Example:

- GET /customers/ (Retrieve all customers)
- GET /customers/1 (Retrieve the details for customer 1)

A successful GET method typically returns HTTP status code **200 (OK)**.
If the resource cannot be found, the method should return **404 (Not Found).**

POST methods

Example:

- POST /customers/ ((Bulk) Create a new customer)
- POST /customers/1 (Create a new customer)

If a POST method **creates a new resource, it returns HTTP status code 201 (Created).** The URI of the new resource is included in the Location header of the response. **The response body contains a representation of the resource.**

If the client puts **invalid data into the request,** the server should return **HTTP status code 400 (Bad Request).** The response body can contain additional information about the error or a link to a URI that provides more details.

PUT methods

Example:

- PUT /customers/1 (Update the details of customer 1 if it exists)

If a PUT method **updates an existing resource, it returns either 200 (OK)** or 204 (No Content).

In some cases, it **might not be possible to update an existing resource. In that case, consider returning HTTP status code 409 (Conflict).**
HTTP status code 400 (Bad Request).

DELETE methods

Example:

- DELETE /customers/1 (Remove customer 1)

If the **delete operation is successful**, the web server should respond with **HTTP status code 204**, indicating that the process has been successfully handled, but that the response body contains no further information.

If the resource doesn't exist, the web server can return HTTP 404 (Not Found).

Real World Example

</talentlabs>

For the lab today:
We will take a look at the Customer API of Stripe:
https://stripe.com/docs/api/customers