Lecture #5 RESTful Web Services #2 Spring 2024

Introduction to Designing URIs for Queries

- URI uniform resource identifier
- URL uniform resource locator
- URN uniform resource name



- Consistency
- Clarity
- Predictability
- Security
- Persistence



Consistency



- Consistency
- Clarity



- Consistency
- Clarity
- Predictability



- Consistency
- Clarity
- Predictability
- Security



- Consistency
- Clarity
- Predictability
- Security
- Persistence



- Consistency
- Clarity
- Predictability
- Security
- Persistence



- Usability
- Scalability
- RESTful Architecture





- Usability
 - Intuitive
 - Consistent
 - Easy to understand



- Usability
- Scalability



- Usability
- Scalability
- RESTful Architecture



https://example.com/api/books?author=Mark+Twain&page=2

Understanding Query Parameters

Parameters

• (

Optional

Required

Parameters

• Optional

https://example.com/api/products?category=electronics &price_range=100-500 &sort=price_asc &page=2 &limit=10



Required

https://example.com/api/products?product_id=12345

Parameters

Well-designed URI for a RESTful service

https://example.com/api/cars

Well-designed URI for a RESTful service

https://example.com/api/cars

- https:// protocol
- <u>example.com</u> domain name
- /api/cars endpoint

More expressive

https://example.com/api/cars?price_min=10000&price_max=20000

• Query Parameters

• price_min

• price_max

- Easy to read
- Easy to remember



• Expressive to convert the purpose

Define Query Responses

- HTTP Status Codes:
 - 200 OK
 - 400 Bad Request
 - 404 Not Found
 - 500 Internal Server Error

Responses Body

HTTP/1.1 200 OK Content-Type: application/json "data": [{ "id": 1, "name": "John Doe", "age": 30 $\},$ "id": 2, "name": "Jane Doe", "age": 25 }

- Caching improves performance by storing frequently accessed data in memory.
- Caching can reduce server load and improve response times.
- Web caching can be implemented using browser caching or server-side caching.



Setting Expiration Caching Headers

- is considered stale.
- content and whether it can be cached by intermediate proxies.
- changed before returning it.

• The Expires header specifies a date and time after which the cached content

 The Cache-Control header provides more granular control over caching by allowing you to specify caching behavior, such as the maximum age of the

• The ETag header allows the server to check whether the cached content has

Setting Cache-Control Header in Java

- Use the HttpServletResponse class to set the Cache-Control header.
- and the caching behavior as the second argument.
- following code:

response.setHeader("Cache-Control", "max-age=3600");

• Call the setHeader method and pass "Cache-Control" as the first argument

• For example, to set a maximum age of 3600 seconds (1 hour), use the

Extensibility and Versioning

- without breaking existing clients.
- breaking changes or add new features.
- Deciding when to version an API can be a challenging decision.

• Extensibility is the ability to add new features and functionality to an API

• Versioning is the practice of creating new versions of an API to introduce

When to Version an API

- When adding breaking changes that will affect existing clients.
- When adding new features that are not backwards-compatible.
- When the current API is no longer meeting the needs of users.

Versioning an API in Practice

- Use a version number in the API's URL to indicate the version of the API.
- Create a new version of the API when making breaking changes or adding new features.
- Use API documentation to help clients migrate to the new version.

- nabling discovery allows clients to easily find and understand the functionality of a RESTful API.
- The use of hypermedia links can be a powerful tool for enabling discovery.
- A well-designed API documentation can also greatly aid in the discovery process.



Using Hypermedia Links for Discovery

- point to.
- in the API.
- Clients can use hypermedia links to navigate the API and discover its functionality.

• Hypermedia links are links that contain information about the resource they

• Hypermedia links can be used to represent relationships between resources

Using Hypermedia Links in a RESTful API

- Use hypermedia links to represent relationships between resources.
- Include links to related resources in response payloads.
- Use link relation types to define the relationship between resources.

```
"books": [
  "id": 1,
  "author": "Douglas Adams",
  "links": [
     "rel": "self",
     "href": "/books/1"
    ٢,
     "rel": "delete",
     "href": "/books/1",
     "method": "DELETE"
  "id": 2,
  "title": "1984",
  "author": "George Orwell",
  "links": [
    "rel": "self",
     "href": "/books/2"
```

"title": "The Hitchhiker's Guide to the Galaxy",

```
"id": 2,
  "title": "1984",
  "author": "George Orwell",
  "links": [
    "rel": "self",
     "href": "/books/2"
    5,
    "rel": "delete",
     "href": "/books/2",
     "method": "DELETE"
,
"links": [
  "rel": "self",
  "href": "/books"
 $,
  "rel": "add",
  "href": "/books",
  "method": "POST"
```

aution. Ocorge Orwell, "links": ["rel": "self", "href": "/books/2" **}**, "rel": "delete", "href": "/books/2", "method": "DELETE" **_ >** "links": ["rel": "self", "href": "/books" **}**, "rel": "add", "href": "/books", "method": "POST"

Using API Documentation for Discovery

- Well-designed API documentation can greatly aid in the discovery process.
- API documentation should include information about the API's resources, operations, and relationships.
- API documentation should be easy to navigate and understand.

Using API Documentation for Discovery

GET /products Retrieve a list of all products in the collection.

GET /products/{id} Retrieve a specific product by ID.

POST /products Add a new product to the collection.

PUT /products/{id} Update an existing product.

DELETE /products/{id} Delete a product from the collection.

HATEOAS in RESTful Web Services

- What is HATEOAS?
- Why is it important?

• How can it be implemented in a RESTful API?

- A constraint of the REST architectural style
- Allows clients to navigate APIs through hypermedia links and controls
- Enables discovery and exploration of APIs

What is HATEOAS?

Implementing HATEOAS in a RESTful API

- Define resources and resource relationships
- Include hypermedia links and controls in responses
- Use a standardized format for links and controls

"product": { "id": "123", "name": "Example Product", "description": "This is an example product", "price": 19.99, "category": "electronics", "_links": { "self": { "href": "/products/123" }, "category": { "href": "/categories/electronics" }

Why is HATEOAS important?

- Makes APIs self-describing and easier to use
- Enables discovery and exploration of APIs
- Supports evolution of APIs over time

- Restful
 - Queries
 - Caching
 - Versioning
 - Discovery

Lecture outcomes

