REST API Design

Mark J Tyers

REST API Entry Point

There should only be one entry point

This should be provided to the client to enable the API to be discovered

This URL should return:

- Information on API version, supported features, etc.
- A list of top-level collections.
- A list of singleton resources.
- Statistics, etc.

Collections and Resources

These are the "things" that the client interacts with

Define the nouns in the system

Each should have a unique URL

These URLs should be defined by the API and provided to the client

Resources	Individual items that contain their own data such as a book
Collections	Groups of resources that share the same properties such as a set of books

Collections

A number of resources that share the same properties

Identified in the URL using the plural of the resource name, eg.

```
/books

/articles

/accounts
```

Notice that in each case the collection is a **plural noun**.

Resources

A unique "thing" that forms part of the collection

Belongs to a specific collection

The URL should include a suitable unique identifier

In each example the collection describes what type of resource it is.

`/books/1449336361`	the **ISBN number** is a good choice
`/articles/42`	here we use the database **auto-increment id**
`/accounts/doej`	the **username** makes sense

The URL Defines Ownership

Collections own resources

But some resources own collections

Consider the following examples:

```
/forums/42/posts

/accounts/doej/clients
```

Note the use of alternate collection / resource path segments.

URL Variables

Resources have unique IDs in the URL

By changing the ID we get different resources

/accounts/doej

/accounts/jonesw

These are called URL variables

We describe these variables using the "Rails Routes" syntax

/accounts/:username

HTTP Methods

HTTP Methods are the verbs that act on collections and elements.

CRUD operations require 4 methods.

CREATE	**POST**
RETRIEVE	**GET**
UPDATE	**PUT**
DELETE	**DELETE**

Additional Methods

OPTIONS	Returns an allow header that contains a list of supported methods
HEAD	Asks for a response identical to a GET request, but without the response body
PATCH	Allows for a partial update of a resource

Methods for Collections

Two actions:

GET	Retrieve the collection
POST	Add a new resource to the collection

Methods for Resources

Three actions:

GET	Retrieve the resource
PATCH/PUT	Update the resource
DELETE	Remove the resource

Handling Errors

API consumers see the API as a black box

Each response needs to include a status code

    401 UNAUTHORISED

Include a human-readable message in the body

```
{
  message: "Basic Authentication required"
}
```

Takeaway

Designing and implementing a REST API with a high level of maturity requires lots of planning

This section contains a lot of very useful advice that should be heeded when designing and implementing an API

Take time to map out the routes, methods, headers and body schemas before attempting to build.

Any Questions?

Coventry University

Test Your Understanding

10:00

Familiarise yourself with the features requested in your assignment topic.

Write down:

- The **collections** and **resources.**
- The **HTTP methods** that each supports.
- The possible **HTTP status codes** that each might return.