Outline

- What is a REST API
- Resource Design
  - Workflow
  - Resource naming
  - Relationships and sub-resources
- URI Design: Rules & Guidelines

What Is a REST API?

- A REST API is modeled as ***collections*** of individually-addressable resources:
  - The ***nouns*** of the API: e.g., books, channels, players, etc.

- Resources are referenced with their resource names

- Resources are manipulated via a small set of ***methods***
  - These methods are known as ***verbs*** or operations.
  - Standard *HTTP methods* for REST APIs are: GET, POST, PUT, PATCH, and DELETE

Resource Design: Workflow

1) Determine what types of resources an API provides
2) Determine the **relationships** between resources
3) Decide the resource name schemes based on types and relationships
4) Decide the resource schemas (that is, resource representations)
5) Assign minimum set of **methods** to resources

REST API: Resources

- REST Web services are designed around resources
- A resource-oriented API is generally modeled as a resource hierarchy (a tree-like structure)
- Each node is either:
  1. Simple (or singleton) resource
  2. Collection resource
     - A resource can be seen as a directory: it is a container that can contain files and subdirectories

REST API: Resources Cont'd

- Resources can be grouped into ***collections***
- A collection resource contains a list of resources (items) of the ***same type***
- A collection resource can be paged, sorted, and filtered
- A resource:
  - Has a state: that is, a representation
  - And may have a zero or more sub-resources
- Each ***sub-resource*** can be either:
  1. An instance resource (i.e., a leaf)
     - E.g., /books/1, /books/1/authors/3
  2. Or a collection resource (i.e., a branch)
     - E.g., /books/1/authors

REST API: Relationships and Sub-Resources

- Resources model objects from the application data model
- Resources almost always have relationships to other resources
- Relationships are often modeled by a sub-resource

- Modeling relationships
  - ✓ Use the following pattern for **sub-resources**

```
GET  /{resource}/{resource-id}/{sub-resource}
GET  /{resource}/{resource-id}/{sub-resource}/{sub-resource-id}
POST /{resource}/{resource-id}/{sub-resource}
```

REST API: Resources Relationships Cont'd

- Examples of resource relationships:

Relation Type	Resources	Meaning
Independent	/projects, /tasks	Tasks can exist with or without a project
Dependent	/projects, /projects/{id}/tasks	Tasks must belong to a project instance
Associative	/users, /projects/{id}/collaborators	Users assigned to a project become collaborators