Rules of REST

These match most common REST implementations

- URL represents a "resource" to interact with
- HTTP method is the interaction with the resource
- HTTP Status code is interaction result

First Rule of REST

First Rule of REST:

- The URL represents a "resource" to interact with

Often a noun (the HTTP method is the verb)

- **Good** - `/student/`
- **Good** - `/grades/`
- **Good** - `/locations/`
- **Bad** - `/addStudent/`
- **Bad** - `/updateGrade/`
- **Bad** - `/searchLocations/`

URL as resource

- Parameters: in query, body, or path
- Often different based on method
    - `GET /students`
    - `GET /students?startsWith=Am`
    - `POST /students?givenName=Xiu&familyName=Li`
    - `POST /students/Li/Xui/`
    - `PATCH /students/34322/`
    - `DELETE /students?billingStatus=overdue`
- **the path of the URL identifies the "thing"**
    - the params do NOT identify the "thing" (resource)

Second Rule of REST

- HTTP method is the interaction with the resource

The URL is the "thing"

The method is what you "do" to it

Examples of the Second Rule of REST

The method shows the kind of interaction:

- `GET /students/` - read
- `POST /students/` - create
- `PUT /students/Naresh/Rajkumar` - overwrite
- `DELETE /students/Naresh/Rajkumar` - remove
- `PATCH /students/Naresh/Rajkumar` - partial update

These have passed params, but

- Method and the URL alone say what is happening

Third Rule of REST

- HTTP Status code is interaction result

There are many Status codes!

- With meaningful names
- Use them!
- but confirm the meaning (MDN)

Add details in body

Status Codes

Some general "classes" of status codes

- 100-199 (1xx): Informational (very rare)
- 200-299 (**2xx**): Successful
- 300-399 (**3xx**): Redirection
- 400-499 (**4xx**): Error (client-caused)
- 500-599 (**5xx**): Error (server-side)

https://developer.mozilla.org/en-US/docs/Web/HTTP/Status

REST Status Code Examples

Some common scenarios

- **200 (OK)** - Means real success
- **400 (Bad Request)** - bad input
    - Provide detail in body of response
- **404 (Not Found)**
- **500 (Internal Server Error)** - server had issue
    - Not user's fault
    - Not expected!

REST Response Body

- Services shouldn't give error messages for display
    - That moves UI changes to services (yuck)
    - Give error **codes** translated by client code
        - Example: `missing-name`
- JSON is common, even from non-JS services
    - Upside: very portable, very readable
    - Downside: No built-in schema validation

Basic REST Example

```
const people = {};

app.get('/people/', (req, res) => {
  res.json(Object.keys(people));
});

app.get('/people/:name', (req, res) => {
  const name = req.params.name;
  if(people[name]) {
    res.json(people[name]);
  } else {
    res.status(404).json({ error: `Unknown user: ${name}`});
  }
});
```

- `:name` syntax (express) sets the `req.params.name`
- `.json(...)` does `res.send(JSON.stringify(...))`
- AND sets the `content-type` header

More REST Example

```
app.post('/people/', express.json(), (req, res) => {
  const name = req.body.name;
  if(!name) {
    res.status(400).json({ error: "'name' required" });
  } else if(people[name]) {
    res.status(409).json({ error: `duplicate: ${name}`});
  } else {
    people[name] = req.body;
    res.sendStatus(200);
  }
});
```

`express.json()` middleware

- request MUST be `content-type: application/json`
- populates `req.body`

No `content-type` = no `body` value.

Considerations

- JSON for error messages?
- POST data needs to return new identifier
    - `POST /people/` - what is url for new person?
- Slow requests need a "polling" setup
    - A slow query will timeout
    - Return a url to check that responds quickly
- Versioning of services!
    - `/v1/people`
- path to services might conflict with pages
    - `/api/v1/people`

Write a REST service to track people

- **GET** /people - JSON array of names
- **POST** /people/:name - Adds name, returns array
    - Status 409 (Conflict), `{error: "duplicate"}`
    - 400 (Bad Request), `{error: "missing-name"}`
- **DELETE** /people/:name - removes, returns array
    - 400 (Bad Request), `{error: "missing-name"}`

Consider:

- Are you looping through an array many times?
- Why these HTTP methods/verbs?
- Why return the array for each?

Thinking ahead

How would you add authorization requirements?

- pass a parameter that the service checks
- have a cookie that the service checks
- pass a special header that the service checks

What kinds of responses can this add?

- 401 - Authorization required
    - the thing to check wasn't there
- 403 - Forbidden
    - it was there but didn't allow access

Sample Authentication endpoint

- **POST** `/api/v1/session` - sets cookie ("logged in")
- **GET** `/api/v1/session` - client can see if logged in
- **DELETE** `/api/v1/session` - clears cookie ("logout")
- **GET** `/api/v1/people`
    - Requires the cookie be set
    - ...with a value the server knows is valid
    - Returns a 401 value if cookie not set
    - Returns a 403 value if cookie is bad value
    - Other endpoints also make these checks