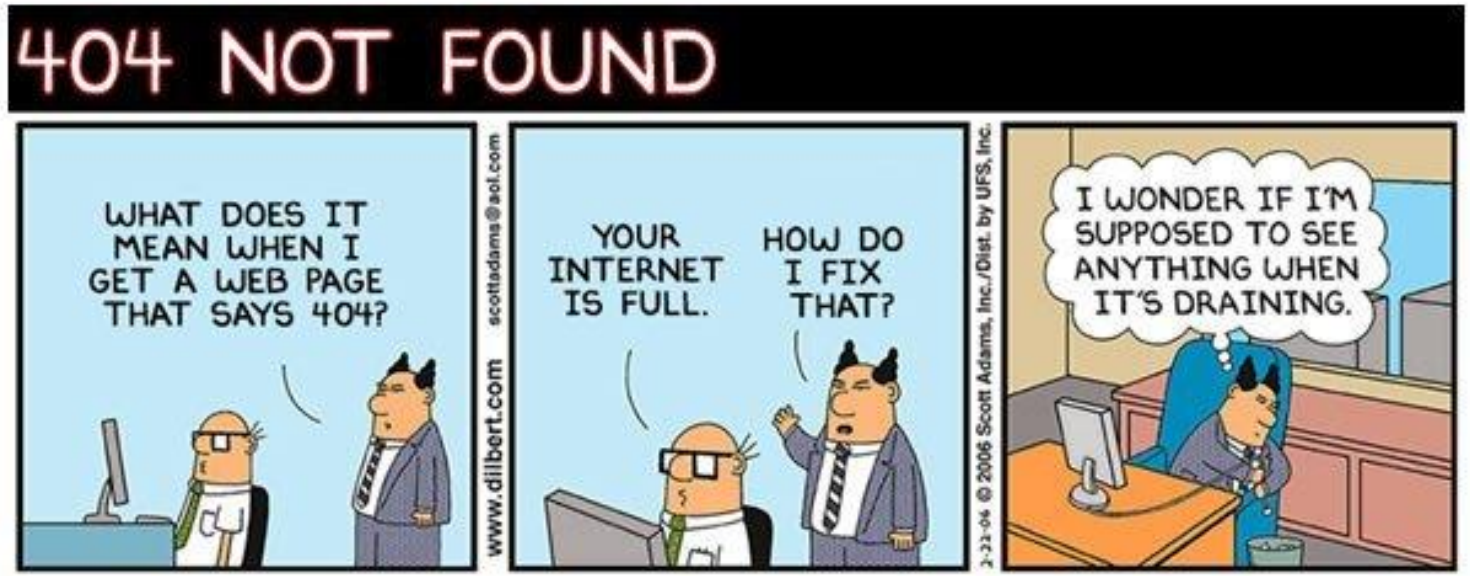
404 NOT FOUND
WHAT DOES IT MEAN WHEN I GET A WEB PAGE THAT SAYS 404?
www.dilbert.com scottadams@aol.com
YOUR INTERNET IS FULL.
HOW DO I FIX THAT?
2-22-04 © 2006 Scott Adams, Inc./Dist. by UFS, Inc.
I WONDER IF I'M SUPPOSED TO SEE ANYTHING WHEN IT'S DRAINING.

HTTP Web services

POST

Objectives

- Implement Java code that can send `POST`, `PUT`, and `DELETE` requests that send data to the server in JSON request body
- Handle an error in network communication properly
- Handle a response containing a 4xx status code
- Use Postman to make a `PUT`, `POST`, or `DELETE` request

Spring Framework

- Powerful, lightweight framework used for application development
- Supports various other frameworks
  - Struts, Hibernate, Tapestry, EJB, JSF, etc.
- Helps solve many technical problems
- Comprehensive tool for supporting applications using Java

Spring Framework

- POJO based
- Modular
- IoC (Inversion of Control)
- Integration with existing frameworks
- Testable
- Web MVC
- Central Exception handling
- Lightweight

More Request Types

- GET: reads the data
- POST: Ideally suited for inserting new data into the data source.
- PUT: Ideally suited for updating an existing record within a data source.
- DELETE: Ideally suited for removing an existing record from the data source.

For the POST & PUT requests we are converting an object to data

GET vs. POST

GET

- Can be cached
- Remain in browser history
- Can be bookmarked
- Should never be used for sensitive data
- Maximum length of 2048 characters
- Used to request data

POST

- Are never cached
- Do not remain in browser history
- Cannot be bookmarked
- Have no restrictions on length

Implementing a POST

Suppose the documentation for the API specifies POST as well :

(POST) *http://localhost:3000/hotels/{id}/reservations*

```
String API_BASE_URL = http://localhost:3000/ ;
RestTemplate restTemplate = new RestTemplate();
HttpHeaders headers = new HttpHeaders();
headers.setContentType(MediaType.APPLICATION_JSON);

// Where reservation is an object of type Reservation.
HttpEntity<Reservation> entity = new HttpEntity<>(reservation, headers);

reservation = restTemplate.postForObject(BASE_URL + "hotels/" + reservation.getHotelID() +
"/reservations", entity, Reservation.class);
```

Note that POST requests have a body and header as well!

Let’s Code!

Implementing a PUT

Suppose the API's documentation states that there is a PUT endpoint: (PUT) *http://localhost:3000/reservations/{id}*

Using a REST template we can implement the following:

```
String API_BASE_URL = http://localhost:3000/;
RestTemplate restTemplate = new RestTemplate();
HttpHeaders headers = new HttpHeaders();
headers.setContentType(MediaType.APPLICATION_JSON);

// Where reservation is an object of type Reservation.
HttpEntity<Reservation> entity = new HttpEntity<>(reservation, headers);

restTemplate.put(BASE_URL + "reservations/" + reservation.getId(), entity);
```

Implementing a DELETE

Suppose the API's documentation states that there is a DELETE endpoint: (DELETE) *http://localhost:3000/reservations/{id}*

Using a REST template we can implement the following:

```
String API_BASE_URL = http://localhost:3000/;
RestTemplate restTemplate = new RestTemplate();
// Where id is an int:
restTemplate.delete(BASE_URL + "reservations/" + id);
```

Let’s Create the PUTs & DELETEs requests

Successful status codes

Successful POST, PUT and Delete return successful status codes:

POST – returns 201

PUT – 200 or 204

DELETE – 202 or 204

Exceptions and Error Handling

There are 2 exceptions to be aware of when dealing with APIs:

- RestClientResponseException – for when a status code other than a 2XX is returned.
- ResourceAccessException – for when there was a network issue that prevented a successful call

```
try {
    restTemplate.put(API_URL + "users/remove/23");
}
catch (ResourceAccessException ex) {
    // Handle network I/O errors
    System.out.println(ex.getMessage());
}
catch (RestClientResponseException ex) {
    // Handle response status codes: 1xx, 3xx, 4xx, 5xx
    System.out.println(ex.RawStatusCode());
}
```

We should use try catch blocks to handling these exceptions

```
private Reservation makeReservation(String CSV) {
  String[] parsed = CSV.split(",");

  // invalid input (
  if (parsed.length < 5 || parsed.length > 6 ) {
    return null;
  }

  // Add method does not include an id and only has 5 strings
  if (parsed.length == 5) {
    // Create a string version of the id and place into an array to be concatenated
    String[] withId = new String[6];
    String[] idArray = new String[]{new Random().nextInt(1000) + ""};
    // place the id into the first position of the data array
    System.arraycopy(idArray, 0, withId, 0, 1);
    System.arraycopy(parsed, 0, withId, 1, 5);
    parsed = withId;
  }
  return new Reservation(
          Integer.parseInt(parsed[0].trim()),
          Integer.parseInt(parsed[1].trim()),
          parsed[2].trim(),
          parsed[3].trim(),
          parsed[4].trim(),
          Integer.parseInt(parsed[5].trim())
  );
}
```

false

need to add an id

(set parsed to hold the reference of withId

parsed

1	Minnie Mouse	04/23/21	04/30/21	4
hotelID	fullName	checkInDate	checkoutDate	guests

withId

"899"	1	Minnie Mouse	04/23/21	04/30/21	4

idArray

"899"

copy from the parsed array starting in position 0
to the withId array starting in position 1 - copy 5 elements

copy from the idArray starting in position 0
to the withId array starting in position 0 - copy 1 element

```
private HttpEntity<Reservation> makeEntity (Reservation reservation) {
        HttpHeaders headers = new HttpHeaders();
        headers.setContentType(MediaType.APPLICATION_JSON);
        HttpEntity<Reservation> entity = new HttpEntity<>(reservation, headers);
        return entity;
}
```

Let’s Code!

Objectives

- Implement Java code that can send `POST`, `PUT`, and `DELETE` requests that send data to the server in JSON request body

```
String API_BASE_URL = http://localhost:3000/ ;
RestTemplate restTemplate = new RestTemplate();
HttpHeaders headers = new HttpHeaders();
headers.setContentType(MediaType.APPLICATION_JSON);

// Where reservation is an object of type Reservation.
HttpEntity<Reservation> entity = new HttpEntity<>(reservation, headers);

reservation = restTemplate.postForObject(BASE_URL + "hotels/" +
reservation.getHotelID() + "/reservations", entity, Reservation.class);
```

```
String API_BASE_URL = http://localhost:3000/;

RestTemplate restTemplate = new RestTemplate();

// Where id is an int:

restTemplate.delete(BASE_URL + "reservations/" +
id);
```

```
String API_BASE_URL = http://localhost:3000/;

RestTemplate restTemplate = new RestTemplate();

HttpHeaders headers = new HttpHeaders();

headers.setContentType(MediaType.APPLICATION_JSON);

// Where reservation is an object of type Reservation.

HttpEntity<Reservation> entity = new HttpEntity<>(reservation,
headers);

restTemplate.put(BASE_URL + "reservations/" +
reservation.getId(), entity);
```

Objectives

- Implement Java code that can send `POST`, `PUT`, and `DELETE` requests that send data to the server in JSON request body
- Handle an error in network communication properly

```
try {
    restTemplate.put(API_URL + "users/remove/23");
}
catch (ResourceAccessException ex) {
    // Handle network I/O errors
    System.out.println(ex.getMessage());
}
catch (RestClientResponseException ex) {
    // Handle response status codes: 1xx, 3xx, 4xx, 5xx
    System.out.println(ex.RawStatusCode());
}
```

Objectives

- Implement Java code that can send `POST`, `PUT`, and `DELETE` requests that send data to the server in JSON request body
- Handle an error in network communication properly
- Handle a response containing a 4xx status code

```
try {
    restTemplate.put(API_URL + "users/remove/23");
}
catch (ResourceAccessException ex) {
    // Handle network I/O errors
    System.out.println(ex.getMessage());
}
catch (RestClientResponseException ex) {
    // Handle response status codes: 1xx, 3xx, 4xx, 5xx
    System.out.println(ex.RawStatusCode());
}
```

Objectives

- Implement Java code that can send `POST`, `PUT`, and `DELETE` requests that send data to the server in JSON request body
- Handle an error in network communication properly
- Handle a response containing a 4xx status code
- Use Postman to make a `PUT`, `POST`, or `DELETE` request