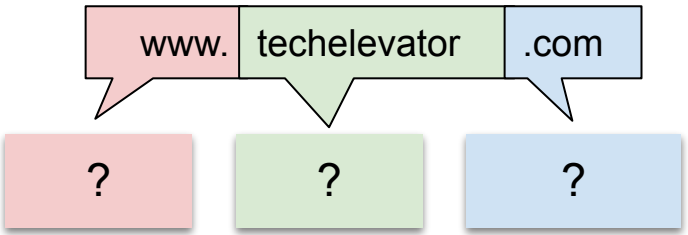
Please Complete The Socrative Pulse Survey!

URL: gosocrative.com
Room Name: JAVAGOLD

Name the 3 parts of this of a domain name:



Review:

1. What is the difference difference between stateless and stateful?
2. Name 4 parts of a HTTP Request.
3. What is the difference between HTTP and HTTPS?
4. For a server error, what range of HTTP status codes would you expect to see in the response?
5. For a client error, what range of HTTP status codes would you expect to see in the response?
6. What is the purpose of a DNS?
7. What does API stand for and what does an API do?

HTTP Web Services POST

Module 2: 12

Today’s Objectives

1. Services
2. HTTP Method Types
3. HTTP Method: GET
4. Object Serialization and Deserialization
5. Handling API Errors
6. HTTP Method: POST
7. HTTP Method: PUT
8. HTTP Method: DELETE
9. Server side API tutorial

Service Class

A service class encapsulates the functionality of code that works as a unit or provides a specific business purpose. This allows other code in the application to interact with the Service Class without needing to know or understand the inner workings of the functionality.

For example: An API service class would encapsulate the functionality of an API so the other code could interact with it as a business unit rather than needing to understand the functionality of the API.

Types of HTTP Methods

Safe - a request that does not change the server.

Safe	**Not Safe**
GET	POST, PUT, DELETE

Idempotent - a request that has the same result regardless of how many times it is completed.

Idempotent	**Not Idempotent**
GET, PUT, DELETE	POST

HTTP Methods - GET

Usage

- HTTP GET is generally used to retrieve web pages to display
- It also is included for
  - images
  - documents
  - stylesheets, script files
- Search Pages
- HTTP GET requests are easily bookmarked because the parameters are in the url.

HTTP GET should never modify any data on the server. Get does not change the state on the server and has the same result each time it is repeated so it is ***Safe Idempotent***

Parameters for a GET request travel as either Path Parameters or in the **Query String**.

http://localhost:3000/hotels/2/reviews?stars=4

HTTP Methods - POST

Usage

- HTTP POST is used when
  - Data must be secure (credit card number, password, etc.)
  - Data is too large for the URL
  - When the request is asking to add something on the server
- HTTP POST requests cannot be bookmarked or sent with the browser directly.

.

HTTP POST indicates that the request will add new data to the the server. Post modifies the server and leaves the server in a different state each time the same request is repeated, so it is ***Not Safe Not Idempotent.***

POST transfers data in the message body instead of the URL.
While HTTPS encrypts the message body, it cannot encrypt the URL.

HTTP Methods - PUT

Usage

- HTTP PUT is used to update existing data
- HTTP PUT requests cannot be bookmarked or sent with the browser directly.
- HTTP PUT requests are meant to overwrite the entire record with the new values.

.

HTTP PUT indicates that the request will update existing data on the the server. Put changes the state on the server but the state remains the same if the same request is repeated multiple times, so it is ***Not Safe Idempotent.***

PUT transfers data in the message body instead of the URL.

While HTTPS encrypts the message body, it cannot encrypt the URL.

HTTP Methods - DELETE

Usage

- HTTP DELETE is used to remove existing data from the server
- HTTP DELETE requests cannot be bookmarked or sent with the browser directly.
- Usually only requires the entities id, so parameters are sent in the query string.

.

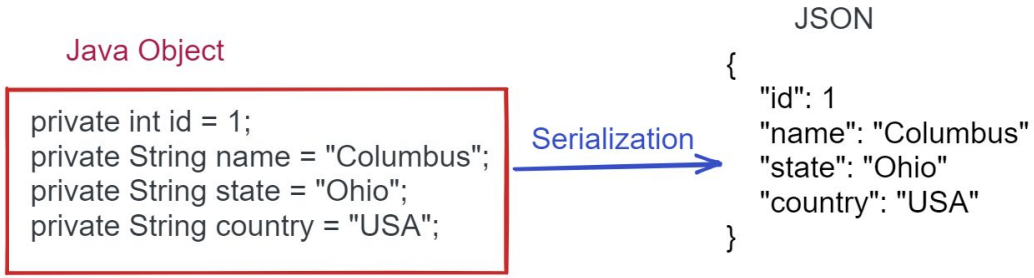
HTTP DELETE indicates that the request will remove existing data from the the server.

Delete changes the state of the server but the state remains the same if the request is repeated multiple times, so it is ***Not Safe Idempotent***.

Object Serializing and Deserializing

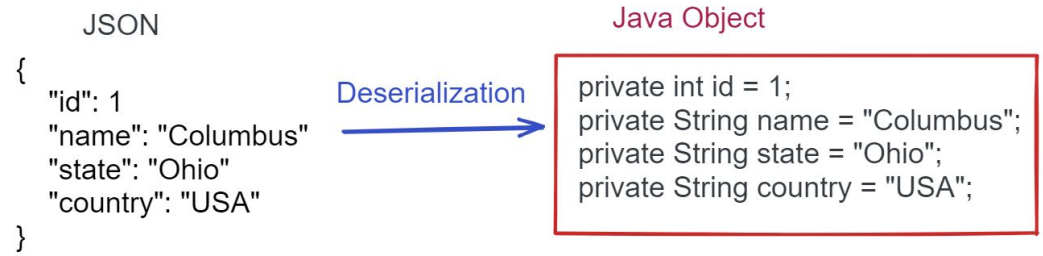
Transforming a Java Object to a string representation of the object, JSON, is called **Serialization.**

Java Object → JSON is **Serialization**



Transforming a string representation of an object, JSON, into an Java Object is called **Deserialization**.

JSON → Java Object is **Deserialization**



Handling API Errors

`RestClientResponseException`

Catches common error status codes, like 401 (Unauthorized), 404 (Not Found) , or 500 (Server Exception).

`ResourceAccessException`

Catches connection errors when the server cannot be reached.

These can be handled using the Java Exception try...catch

```
try {
      hotels = restTemplate.getForObject(BASE_URL + "hotels", Hotel[].class);
catch (RestClientResponseException e) {
      // handle common errors
} catch (ResourceAccessException e) {
      // handle connection errors
}
```

Coding a POST

Setting Headers

```
HttpHeaders headers = new HttpHeaders();

headers.setContentType(MediaType.APPLICATION_JSON);

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

POST with postForObject()

```
restTemplate.postForObject(url, entity, Reservation.class);
```

The **HttpEntity** object contains the *Headers* and message *Body*.

Since we want the reservation to be in the message body instead of the Query String, we set it in the entity object and pass it to the restTemplate.

Coding a PUT

Setting Headers

```
HttpHeaders headers = new HttpHeaders();

headers.setContentType(MediaType.APPLICATION_JSON);

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

POST with put()

```
restTemplate.put(url, entity);
```

Coding a DELETE

POST with put()

```
restTemplate.delete(url);
```