Building API’s

What is an API?

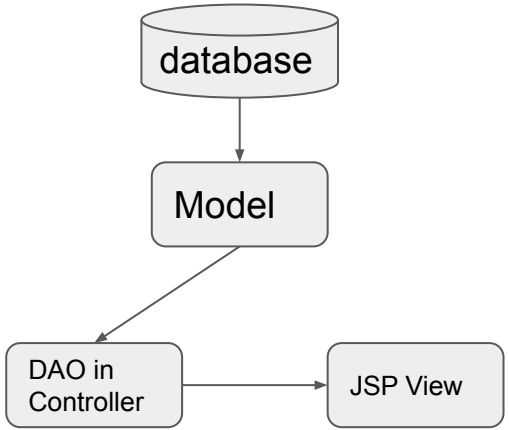
- A set of functions and/or procedures designed to interact with an external system.
- Modern cloud architecture relies heavily on API’s.

API’s as a source of data

- We have explored various ways of obtaining data, starting from having Java read a text file, to building a sophisticated relational database like PostgreSQL.
- API’s could potentially be yet another source of data for other applications to consume.

API’s as a source of data

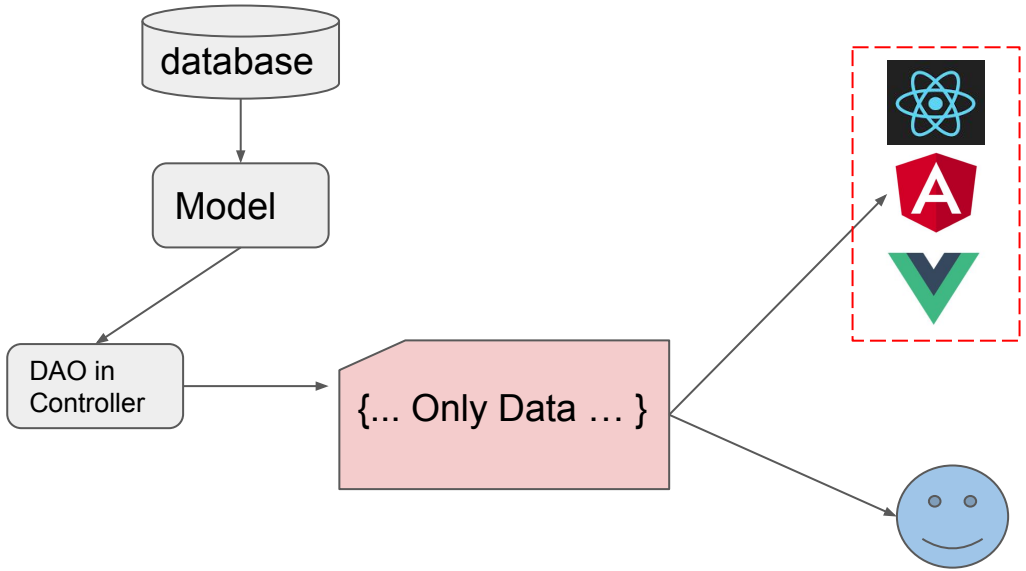
Consider what happens after a controller receives a GET request:



This is the direction data flows in a MVC application

API's as a source of data

Now consider what happens if instead of having a fixed JSP front end, the controllers return only data.



This data can now be sent and used by JS applications.

Building these "front end" apps is the focus of Mod 4!

… or just for a user to analyze the raw data. (analysts and data scientists perhaps?)

REST Controllers

- In order to send back data instead of a view, we need to change our controllers to REST controllers.
  - REST is short for **Re**presentational **S**tate **T**ransfer.

- In theory, Spring makes this easy. We have at our disposal an annotation called **@RestController**.

```
@RestController

public class ProductReviewsController {
… }
```

Let’s Try This Out

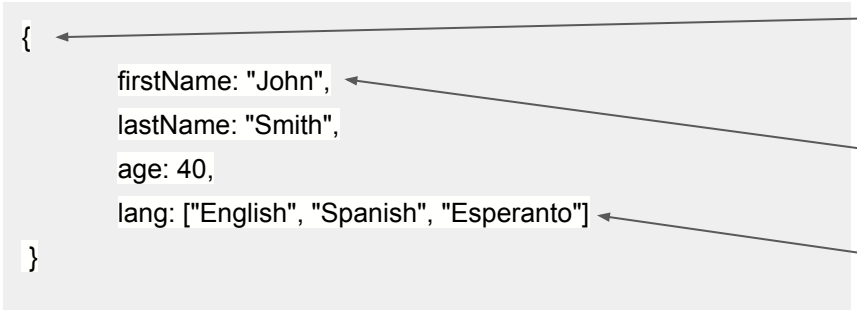
JSON

- The data generated by the controllers are in a format called JSON.

- JSON is simple! Only three rules:
  - Objects in JSON’s are delimited by curly braces { … }
  - Arrays in JSON’s are delimited by brackets [...]
  - Data is listed in key-value pairs (key : value)

JSON Example

Here is an example of JSON data:



- The object itself is enclosed with a set of curly braces.
- Each property of the object is listed as a key value pair.
- An array is enclosed with square brackets.

Back to Rest Controllers: PUT / DELETE

- We are now familiar with GET and POSTS. While working with API's you will likely encounter two additional request types:
  - **PUT**: When we want to update the value of 1 JSON object.
  - **DELETE**: When we want to remove a JSON object.

```
@PutMapping("/{id}")
public ProductReview update(@RequestBody ProductReview productReview, @PathVariable int id) {
        ...
}


@DeleteMapping("/{id}")
public ProductReview delete @PathVariable int id) {
        ...
}
```

Rest Controllers: @PathVariable

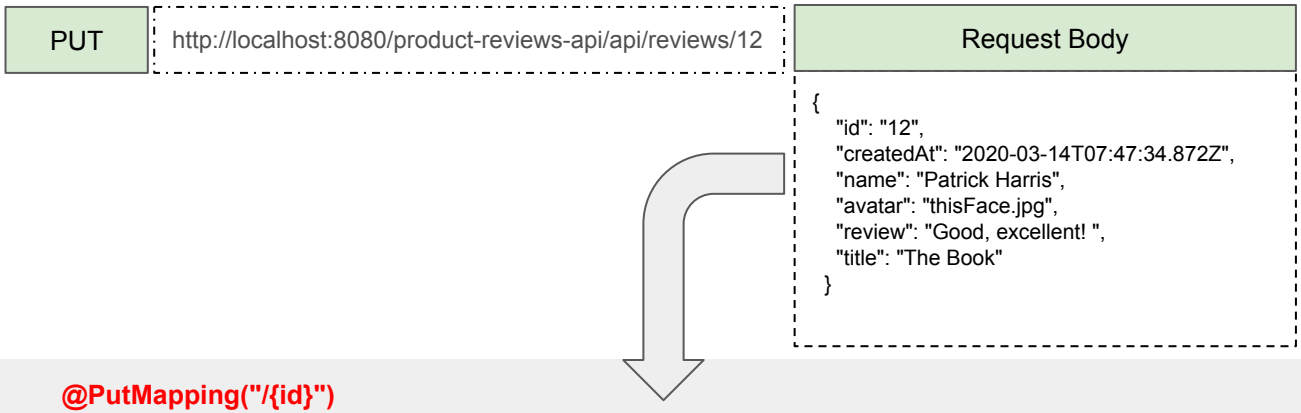
Let’s discuss two additional annotations:

- **@PathVariable**: This is somewhat similar to @RequestMapping. It indicates that part of the URL is a variable the controller needs. Consider this example:


```
@PutMapping("/{id}")
public ProductReview update(@RequestBody ProductReview productReview, @PathVariable int id) {
        System.out.println(id); // output will be 12.
        ...
}
```

Rest Controllers: @RequestBody

@RequestBody: This states that the request is expecting a payload (body) in the form of a JSON object that matches the model definition.




```
@PutMapping("/{id}")
public ProductReview update(@RequestBody ProductReview productReview, @PathVariable int id) {
        System.out.println(id); // output will be 12.
        ...
}
```

Let’s Create All the Endpoints!

Let’s Familiarize Ourselves with Postman