**Please Complete The Socrative Pulse Survey!**

URL: gosocrative.com
Room Name: JAVAGOLD

Review:

Name 4-5 things wrong with this code snippet:

```
@RequestMapping(path = "/hotels/id", method = RequestMethod.POST)
public void get(@RequestParam("hotelId") int id) {
     return hotelDAO.get(id);
}
```

1. In an MVC application, what does the M stand for and what are its responsibilities?

2. In an MVC application, what does the V stand for and what are its responsibilities?

3. In an MVC application, what does the C stand for and what are its responsibilities?

4. Clients send a _____ and Servers return a _____.

5. What annotation marks a class as a REST controller?

# Server Side APIs Part 2

Module 2: 14

# Objectives

1. Dependency Injection
2. RESTful Service Design with CRUD
3. Bean Validation
4. HTTP Response Codes
    a. Error Handling
        i. 400 errors from Validation
        ii. Other errors like 404
5. PUT
6. DELETE

# Dependency Injection

**Inversion of Control (IoC)** is a object oriented principle that inverts the flow of normal flow of control of an application from a class to a higher layer.  It is commonly used in Java Frameworks.

**Dependency Injection** is a design pattern that implements IoC.  It *inverts control* of the instantiation of dependencies from the class using them to an dependency injection container. Rather than instantiating classes with “new” in our class, they will be *injected* into our class at runtime.

**Dependency Injection Container** is code that creates and injects the dependencies into our class.  Spring has a Dependency Injection Container built into it.

# Dependency Injection in Spring Boot

1. Create an Interface with an Implementation class  (CityDao)
2. Add the @Component annotation to the implementation class  (JdbcCityDao)
3. Declare an argument of a constructor using the Interface

```
public CityController( CityDao cityDao)
```

4. Or a variable can be created with the @Autowired annotation, but can make Integration/Unit testing more difficult.

```
@Autowired
private CityDao cityDao;
```

# RESTful Service Design - CRUD

Define the CRUD

1. What do you want your user to be able to Create?  (POST)
2. What do you want your user to be able to Read?  (GET)
3. What do you want your user to be able to Update? (PUT)
4. What do you want your user to be able to Delete?  (DELETE)

Not all CRUD needs to be included, only what you want the user to be able to access.  Some CRUD operations may have multiple routes that do different things:

# Example CRUD for Hotel Reviews

### All Reviews

1. List Reviews
2. Add a Review

### Specific Review

1. Get Review
2. Update Review
3. Delete Review
4. Get Comments for Review
5. Add Comments to Review

### Comment

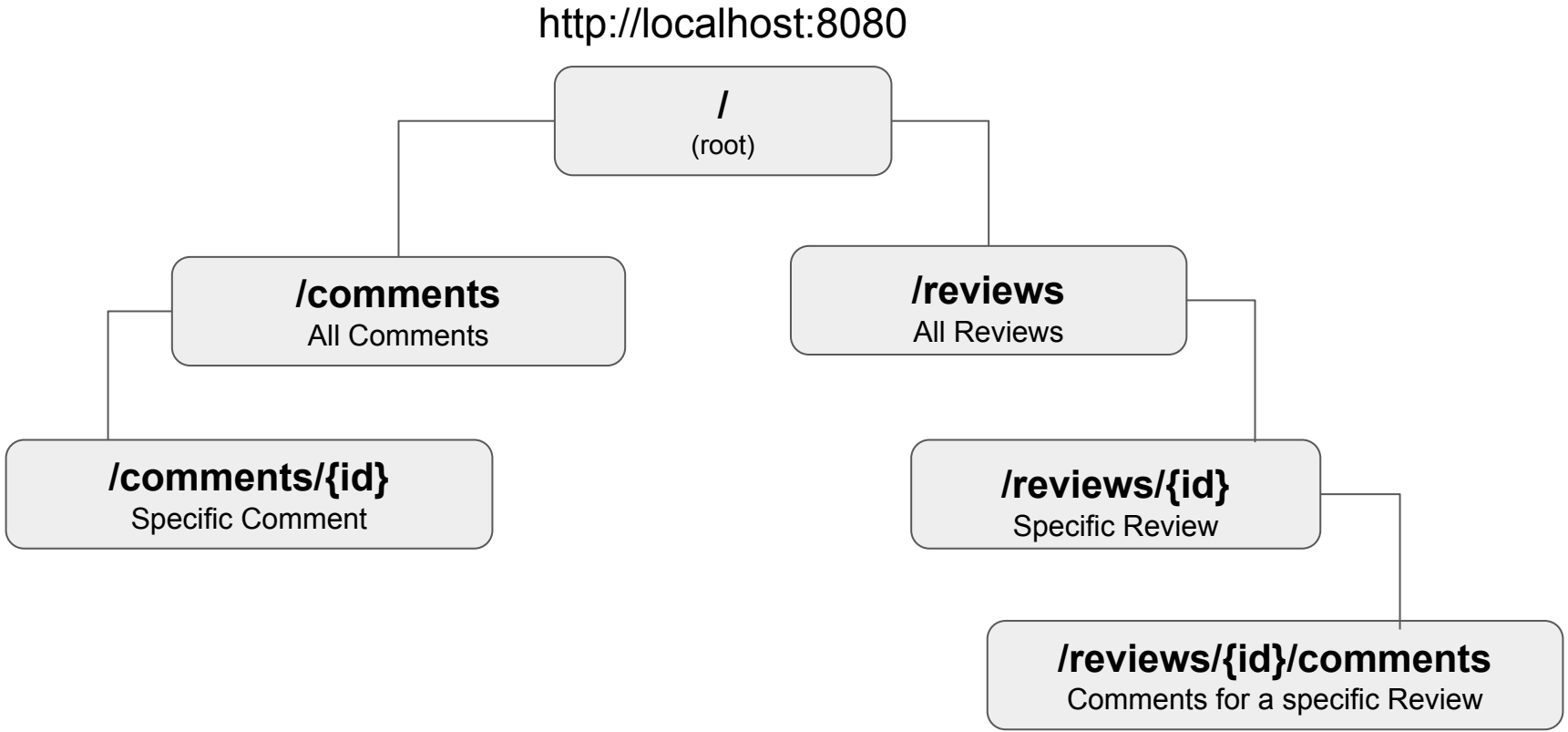
1. Get Comment
2. Update Comment

# RESTful Service Design - End Points

Designing endpoints

1. Endpoints should be plural names that define the entity (what is being accessed)
    a. Hotels, Reservations, Cities, etc.
2. IDs should be sent as Path Variables
3. Filters and Sorting should be sent as Query Parameters
4. Endpoints should reflect the hierarchy of the entities

# Endpoint Hierarchy - Reviews



# Example Endpoint Design for Reviews

| Endpoint | Method | Description | Success | Error |
|---|---|---|---|---|
| **/api/reviews** | GET | List All Reviews | List of Reviews | |
| **/api/reviews** | POST | Add a Review | 201 - CREATED | |
| **/api/reviews/{id}** | GET | Get a specific Review | Review | 404 - Review Not Found |
| **/api/reviews/{id}** | PUT | Update a Review | 200 - OK | 404 - Review Not Found |
| **/api/reviews/{id}** | DELETE | Delete a Review | 204 - No Content | 404 - Review Not Found |
| **/api/reviews/{id}/comments** | GET | List Comments for a Review | List of Comments | 404 - Review Not Found |
| **/api/reviews/{id}/comments** | POST | Add a comment to a Review | 201 - CREATED | 404 - Review Not Found |
| **/api/comments/{id}** | GET | Get a specific Comment | Comment | 404 - Comment Not Found |
| **/api/comments/{id}** | PUT | Update a Comment | 200 - OK | 404 - Comment Not Found |

# Java Bean

A standard for Classes that define data objects to ensure they can be consistently used by libraries and frameworks and correctly serialized/deserialized.  Java Beans have the following characteristics:

1. All properties are private with getters and setters
2. Has a public no-argument constructor
3. Implements Serializable

# Java Bean Validation

1. Add Constraint Annotations to the Data Class member variables

```
@Min( value = 1, message = "The field 'hotelID' is required.")
private int hotelID;
@NotBlank( message = "The field 'fullName' is required.")
private String fullName;
@NotBlank( message = "The field 'checkinDate' is required.")
private String checkinDate;
@NotBlank( message = "The field 'checkoutDate' is required.")
private String checkoutDate;
@Min( value = 1, message = "The minimum number of guests is 1")
@Max( value = 5, message = "The maximum number of guests is 5")
private int guests;
```

List of Constraints

2. Add @Valid Annotation to the Controller Method receiving the object.

```
public Reservation addReservation(@Valid @RequestBody Reservation
reservation
```

# Returning a 2xx Status

200 - OK will return automatically if an exception is not thrown during the execution of the controller method.

A more specific 2xx Status Codes (201-Created, 204-No Content) can be returned from a controller method when no exception is thrown by applying the @ResponseStatus annotation to the method

```
@ResponseStatus(HttpStatus.CREATED)
```

List of 2xx Status Codes

# Returning a 4xx Status

4xx Status Codes can be returned by a controller method by throwing an Exceptions from the the Controller Method that has the @ResponseStatus annotation applied.

**Exception subclass:**

```
@ResponseStatus( code = HttpStatus.NOT_FOUND, reason = "Hotel not found.")
 public class HotelNotFoundException extends Exception
```

**Controller method:**

```
public Reservation addReservation(@Valid @RequestBody Reservation
reservation, @PathVariable("id") int hotelID throws HotelNotFoundException
```

List of 4xx Status Codes