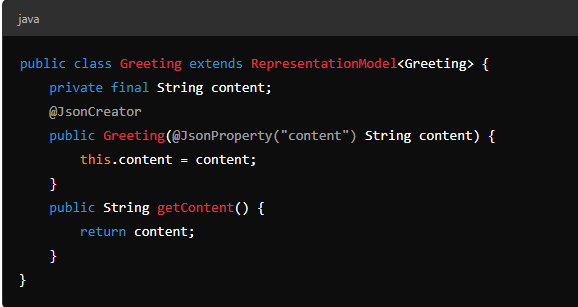
To build a **hypermedia-driven RESTful web service** with Spring, follow these key steps:

**1. Set Up the Project**

* Use **Spring Initializr** (<https://start.spring.io>) to create a project with the necessary dependencies (select **Spring HATEOAS**).
* Choose **Gradle** or **Maven** as the build tool, and Java as the language.
* Download and import the project into your IDE (like Spring Tool Suite, IntelliJ IDEA, or VSCode).

**2. Create a Resource Representation Class**

* Define a class for the resource, which extends RepresentationModel from Spring HATEOAS.
* Example: Create a Greeting class to hold the content and hypermedia links.



**3. Build the REST Controller**

* Create a controller to handle HTTP GET requests using @RestController.
* Example: A GreetingController handles requests at /greeting and uses @RequestParam to allow for an optional name query parameter.

java

@RestController

public class GreetingController {

private static final String TEMPLATE = "Hello, %s!";

@RequestMapping("/greeting")

public HttpEntity<Greeting> greeting(@RequestParam(value = "name", defaultValue = "World") String name) {

Greeting greeting = new Greeting(String.format(TEMPLATE, name));

greeting.add(linkTo(methodOn(GreetingController.class).greeting(name)).withSelfRel());

return new ResponseEntity<>(greeting, HttpStatus.OK);

}

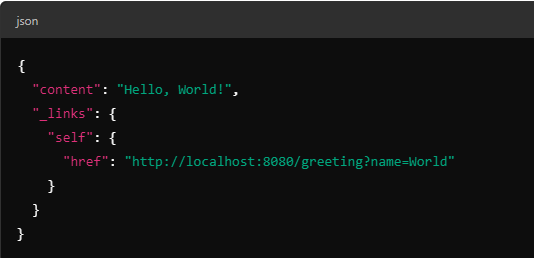
}

* The linkTo and methodOn methods dynamically build hypermedia links.

**4. Build and Run the Application**

* Run the app using either Gradle or Maven:
  + Gradle: ./gradlew bootRun
  + Maven: ./mvnw spring-boot:run
* You can also build an executable JAR file:
  + Gradle: ./gradlew build, then java -jar build/libs/app.jar
  + Maven: ./mvnw clean package, then java -jar target/app.jar

**5. Test the Service**

* Visit http://localhost:8080/greeting to get the default response:
* 

Try customizing the greeting with a query string, like ?name=User, and see the updated response.

**6. Summary**

* Use Spring Initializr for setup.
* Create a resource representation class that extends RepresentationModel.
* Develop a REST controller with hypermedia links.
* Build and test the application by running it locally and verifying the responses.