



CICLO 3

Comunicación HTTP y Reglas de Negocio en **Java Spring Boot**



Agenda semana 3

- **Semana 3: Comunicación HTTP y reglas de negocio en Java Spring Boot**
 - **Protocolo HTTP**
 - **Controladores y servicios en Java Spring Boot**



Objetivo de aprendizaje

- Entender cómo funciona la comunicación HTTP en Java Spring Boot





Agenda 1

- Ejercicio práctico de calentamiento **Spring Boot** (repaso semana 2)
- **Teoría Protocolo HTTP**
- Ejercicio práctico **Protocolo HTTP**
- **Teoría comunicación HTTP**
- Ejercicio práctico **Teoría comunicación HTTP**
- **Teoría comunicación HTTP en Java Spring Boot**
- Ejercicio práctico **Spring Boot**

Ejercicio práctico de calentamiento (repaso semana 2)

- Spring Quickstart Guide
 - **Step 1:** Start a new Spring Boot Project
 - **Step 2:** Add your code
 - **Step 3:** Try it

Protocolo HTTP

¿Cómo funciona el internet?

URL

Ejemplo de una URL

`http://www.example.com/search?item=vw+beetle`

Protocol

Domain

Path

Parameters

Encabezados HTTP

- Primera linea
- Encabezados
- Cuerpo/Contenido

```
POST / HTTP/1.1
```

```
Host: localhost:8000
```

```
User-Agent: Mozilla/5.0 (Macintosh; ... ) Firefox/51.0
```

```
Accept: text/html,application/xhtml+xml,...,*/*;q=0.8
```

```
Accept-Language: en-US,en;q=0.5
```

```
Accept-Encoding: gzip, deflate
```

```
Connection: keep-alive
```

```
Upgrade-Insecure-Requests: 1
```

```
Content-Type: multipart/form-data; boundary=-12656974
```

```
Content-Length: 345
```

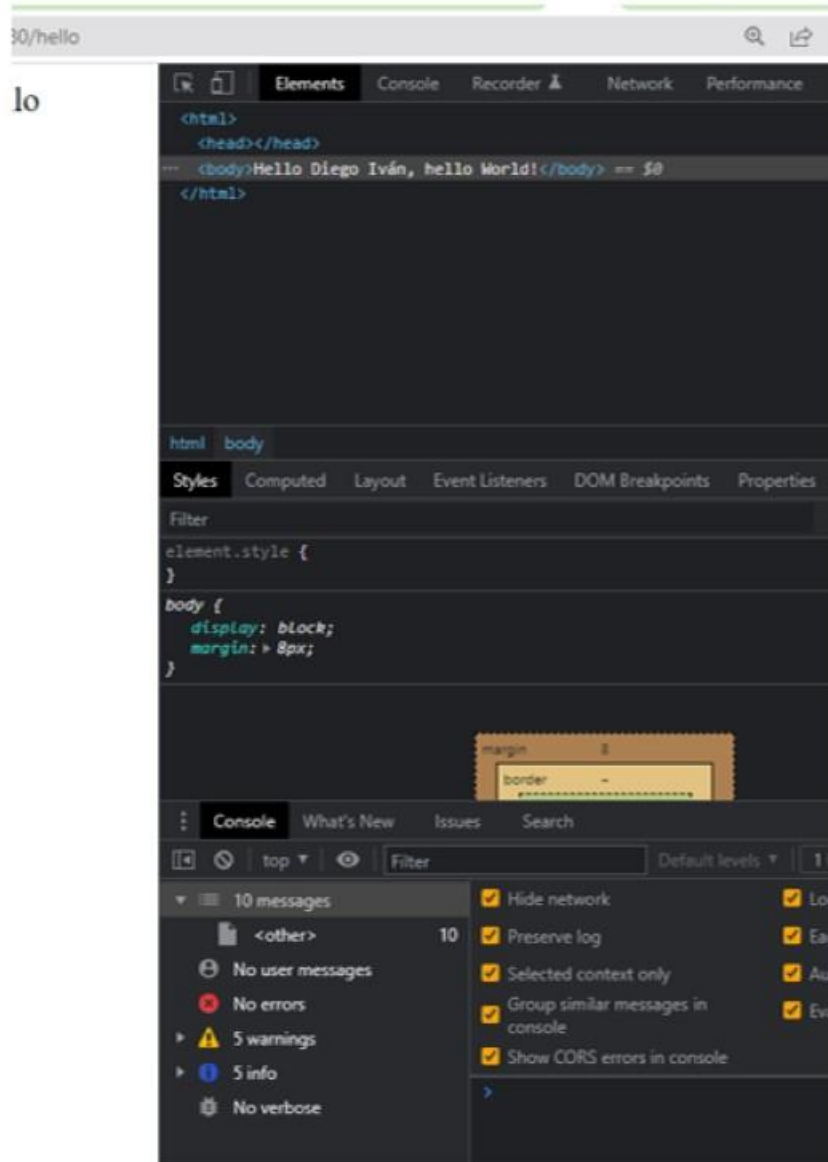
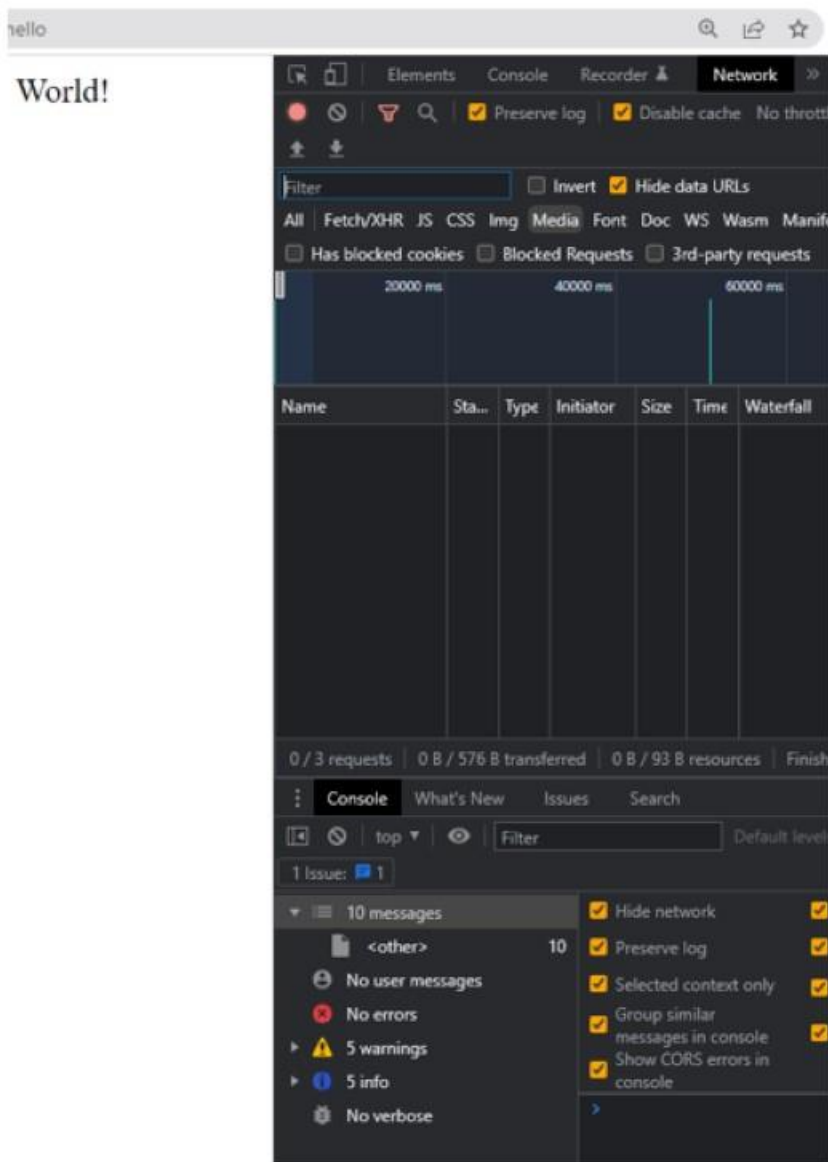
Request headers

General headers

Entity headers

```
-12656974
```

```
(more data)
```

Ejercicio práctico Teoría comunicación HTTP

The background is a vibrant blue digital landscape. It features a curved horizon line that resembles the Earth's surface, with a bright light source on the right creating a lens flare and illuminating the scene. Below the horizon, there are layers of glowing binary code (0s and 1s) and abstract, flowing lines that suggest data movement or network connections.

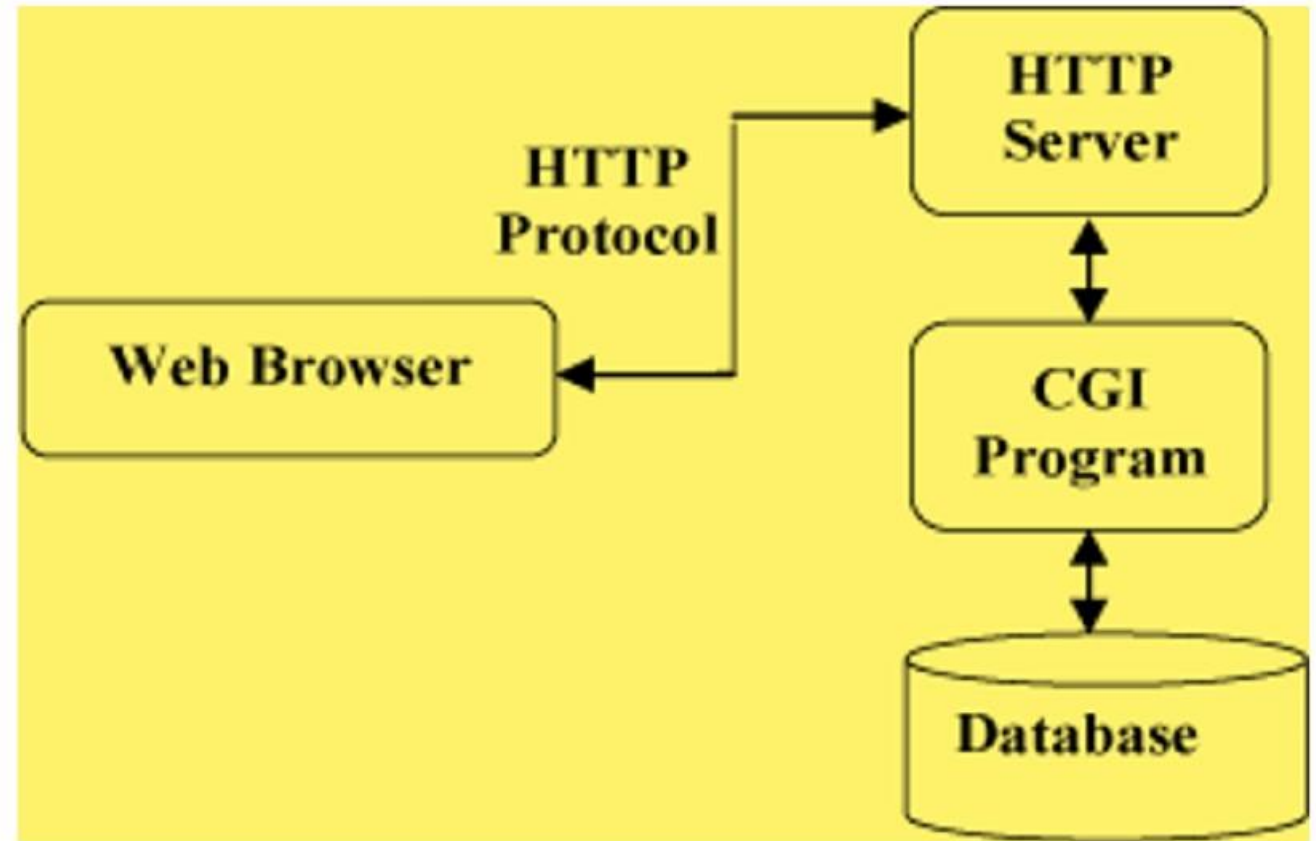
Comunicación HTTP

Agenda

- HTTP Profile
- HTTP message structure
- HTTP request method
- HTTP response headers
- HTTP status codes
- HTTP content-type

¿Cómo funciona Http?

- HTTP no tiene conexión
- HTTP es la independencia de los medios
- HTTP es un protocolo sin estado



HTTP message structure

Client requests:

- GET /hello.txt HTTP/1.1
- User-Agent: curl/7.16.3 libcurl/7.16.3 OpenSSL/0.9.7l zlib/1.2.3
- Host: www.example.com
- Accept-Language: en, mi

Output:

- Hello World! My payload includes a trailing CRLF.

Server response:

- HTTP/1.1 200 OK
- Date: Mon, 27 Jul 2009 12:28:53 GMT
- Server: Apache
- Last-Modified: Wed, 22 Jul 2009 19:15:56 GMT
- ETag: "34aa387-d-1568eb00"
- Accept-Ranges: bytes
- Content-Length: 51
- Vary: Accept-Encoding
- Content-Type: text/plain

HTTP request method

- HTTP standard
- HTTP1.0
- HTTP1.1

o.	method
1	GET
2	HEAD
3	POST
4	PUT
5	DELETE
6	CONNECT
7	OPTIONS
8	TRACE

- El encabezado de solicitud HTTP proporciona información sobre la solicitud, la respuesta u otras entidades de transmisión.
- En esta sección, presentaremos encabezados de respuesta HTTP específicos.

sponse header

Allow

Content-Encoding

Content-Length

Content-Type

Date

Expires

Last-Modified

Location

Refresh

Server

Set-Cookie

WWW-Authenticate

HTTP status codes



200-- request was successful



301-- resources (web pages, etc.) is permanently transferred to another URL



404 - resources (web pages, etc.) requested does not exist



500 - Internal Server Error

HTTP status code classification

classification	Category Description
1**	Information, the server receives the request, the requester needs to proceed
2**	Successful, the operation is successfully received and processed
3**	Redirection, further action is required to complete the request
4**	Client error, the request contains syntax errors or not complete your request
5**	Server error, a server error occurred during the processing of the request

100- 200

- Status Code Status Code English name Chinese description
- 100 Continue carry on. Client should continue its request
- 101 Switching Protocols Switching protocols. Server switching protocol based on the client's request. Can only switch to a more advanced protocol, for example, to switch to the new version of the HTTP protocol
- 200 OK Request was successful. Generally used for GET and POST requests
- 201 Created It has been created. Successful requests and created a new resource
- 202 Accepted accepted. We have accepted the request, but did not complete the process
- 203 Non-Authoritative Information Unauthorized information. Request was successful. But not in the original meta information returned by the server, but a copy of the
- 204 No Content Empty. The server successfully processed, but did not return content. In the absence of updated pages to ensure the browser continues to display the current document
- 205 Reset Content Reset content. Server processing is successful, the user terminal (for example: a browser) should reset document view. This return code can clear your browser's form fields
- 206 Partial Content Part. The server successfully processed a partial GET request
- 300 Multiple Choices multiple choices. Resource request may include a plurality of positions, corresponding to return a list of resource characteristics and address for the user terminal (for example: a browser) Select

300

- 301 Moved Permanently Moved Permanently. The requested resource has been permanently moved to a new URI, will return information including the new URI, the browser will automatically be directed to the new URI. Any future new request should be replaced with a new URI
- 302 Found Temporary move. Similar to 301. But the resource only temporarily moved. The client should continue to use the original URI
- 303 See Other View other address. Similar to 301. Use GET and POST requests View
- 304 Not Modified Unmodified. The requested resource unmodified, the server returns this status code, it will not return any resources. Client typically caches visited resources by providing a header indicates that the client wish to return only after the specified date modified resource
- 305 Use Proxy Use a proxy. The requested resource must be accessed through a proxy
- 306 Unused It has been abandoned HTTP status code
- 307 Temporary Redirect Temporary redirect. Similar to 302. Use GET request is redirected

400

- 400 Bad Request Syntax error in client requests, the server can not understand
- 401 Unauthorized Request requires user authentication
- 402 Payment Required Reserved for future use
- 403 Forbidden The server understood the request to the client's request, but refused to implement this request
- 404 Not Found The server could not find the resources (web) at the request of the client. With this code, site designers can set "resource you have requested could not be found" personalized page
- 405 Method Not Allowed Client requests prohibited methods
- 406 Not Acceptable The server could not complete your request based on the content characteristics requested by the client
- 407 Proxy Authentication Required Request requires proxy authentication, similar to 401, but the requestor should use proxy authorization
- 408 Request Time-out The server waits for a client to send a request for too long, a timeout
- 409 Conflict Clashes server to complete the client's PUT request is likely to return this code when the server processes the request

400

410	Gone	Resource requested by the client does not already exist. Unlike 410 404, if the resource has now been permanently deleted before you can use the 410 code, website designer can specify resources through a new location code 301
411	Length Required	The server was unable to process the request message sent by the client without the Content-Length
412	Precondition Failed	Prerequisites client requests information errors
413	Request Entity Too Large	Since the request entity is too large, the server can not handle, so the request is denied. To prevent the continuous request of the client, the server may close the connection. If the server is temporarily unable to process only, it will contain information about the response of a Retry-After
414	Request-URI Too Large	URI is too long request (URI typically, a URL), the server can not handle
415	Unsupported Media Type	The server was unable to process the request that came with media formats
416	Requested range not satisfiable	Range client request is invalid
417	Expectation Failed	The server could not meet the request header Expect

500

500	Internal Server Error	Internal server error and could not complete request
501	Not Implemented	The server does not support the requested function, can not fulfill the request
502	Bad Gateway	As a gateway or proxy server received from the remote server to an invalid request
503	Service Unavailable	Because it is overloaded or system maintenance, the server is temporarily unable to handle the client's request. The length of the delay may be included in the Retry-After header information server
504	Gateway Time-out	Acting as a gateway or proxy server, not timely access request from a remote server
505	HTTP Version not supported	The server does not support the requested HTTP protocol version not finish processing

HTTP content-type


xtensión de archivo	Tipo de contenido (tipo Mime)	Extensión de archivo	Tipo de contenido (tipo Mime)
* (Un flujo binario, no sé el tipo de archivo para descargar)	aplicación/flujo de octetos	.tif	imagen / tiff
.001	aplicación / x-001	.301	aplicación / x-301
.323	texto / h323	.906	aplicación / x-906
.907	dibujo / 907	.a11	aplicación / x-a11
.acp	audio / x-mei-aac	.ai	solicitud / posdata
.aif	audio / aif	.aifc	audio / aif
.aiff	audio / aif	.anv	aplicación / x-anv
.como un	texto / asa	.asf	video / x-ms-asf
.áspid	texto / áspid	.asx	video / x-ms-asf

Ejercicio práctico

Postman on the web


You can now access Postman through your web browser. Simply create a free Postman account, and you're in.

Try the Web Version



Why sign up?

- Organize all your API development within Postman Workspaces
- Sync your Postman data across devices
- Backup your data to the Postman cloud
- It's free!



Create Postman Account

[Sign in instead?](#)

Email
ceo@scalapp.co

Username
DiegoIvánOliverosAcosta

Password
SHOW


☒ Sign up to get product updates, news, and other marketing communications.

☒ Stay signed in for 30 days

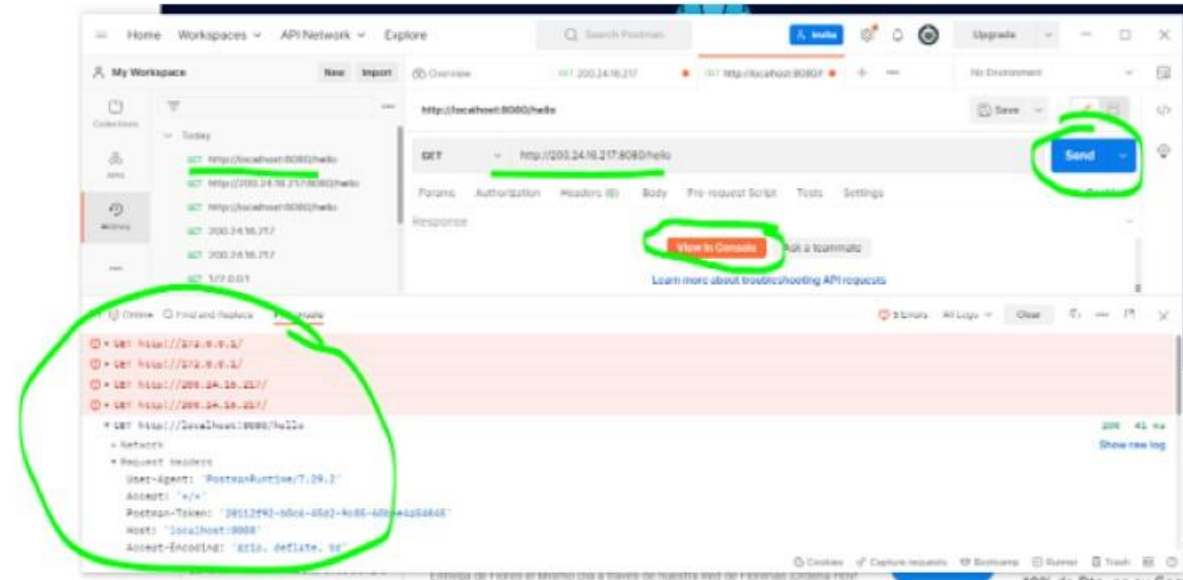
By creating an account, I agree to the [Terms](#) and [Privacy Policy](#).

Create free account

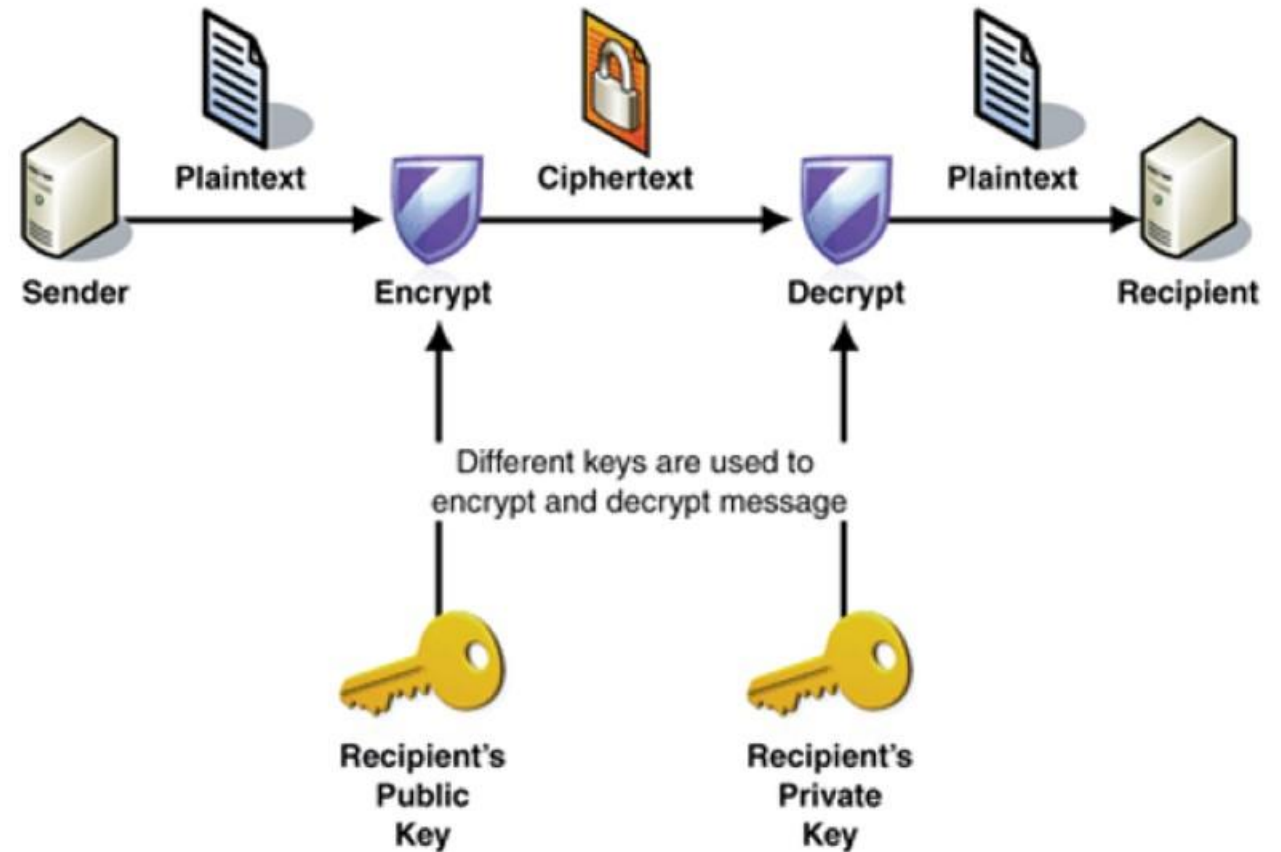
or

 Sign up with Google

Sign in with SSO



HTTPS (Hypertext Transfer Protocol Secure)



Comunicación HTTP en Java Spring Boot

¿Qué son los servicios REST?



Flights

¿Cómo se realizan reservas?



Servicio de reservas basado en la web

- Miembros Premier
- Miembros de viajero frecuente
- Miembros Regulares

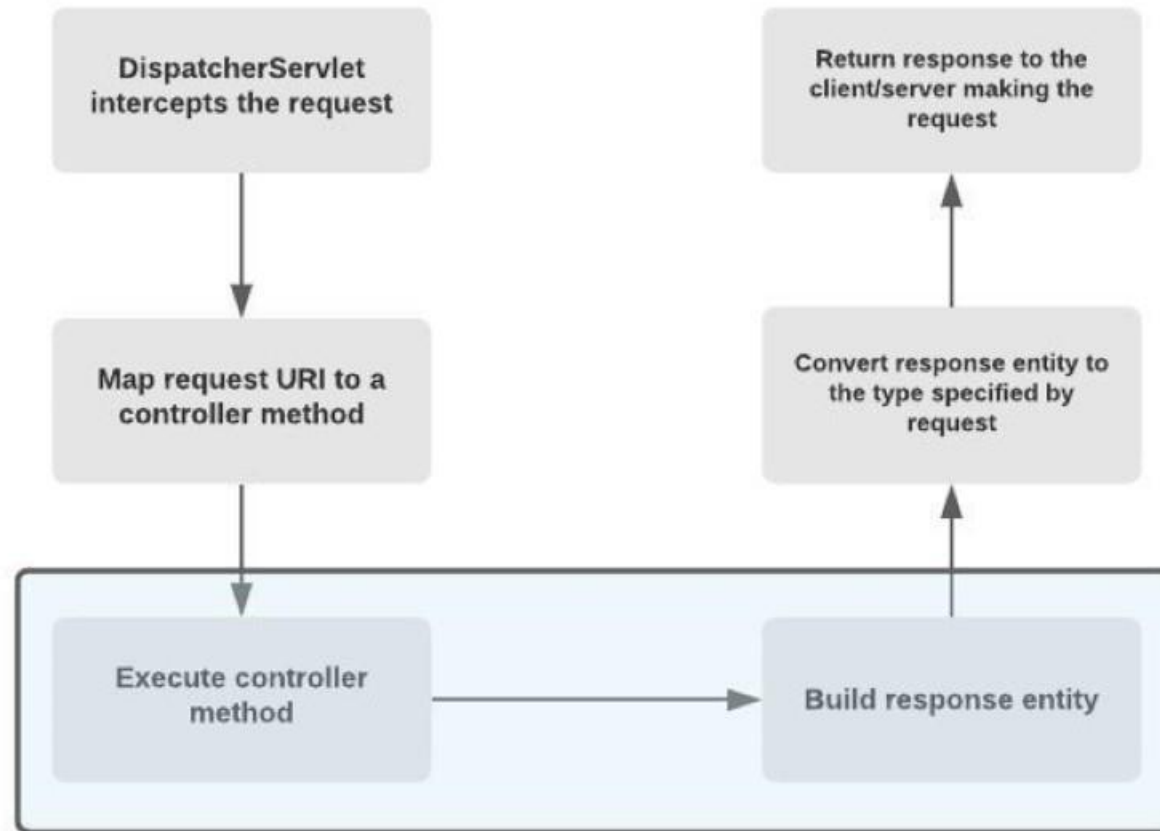
- Enfoque 1
- "Presione 1 para Premier, presione 2 para..."
- Enfoque 2:
- ¡las URL son baratas!
¡Usalos, usalos a ellos!

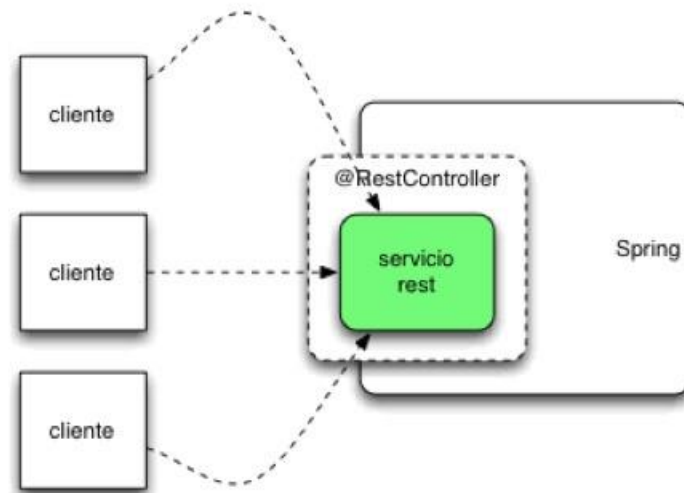


Patrón de diseño REST

- Dos Aspectos Fundamentales
 - **Recursos**
 - **Las URL identifican los recursos**

Flujo de trabajo de la API REST de Spring Boot





```
package com.example.restservice;
```

```
import java.util.concurrent.atomic.AtomicLong;
```

```
import org.springframework.web.bind.annotation.GetMapping;  
import org.springframework.web.bind.annotation.RequestParam;  
import org.springframework.web.bind.annotation.RestController;  
import java.util.ArrayList;  
import java.util.List;
```

```
@RestController
```

```
public class GreetingController {
```

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

```
    private final AtomicLong counter = new AtomicLong();
```

```
    @GetMapping("/greeting")
```

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

```
        //ArrayList<Coche> coches = new ArrayList();
```

```
        return new Greeting(counter.incrementAndGet(), String.format(template, name));
```

```
    }
```

```
    @GetMapping("/greeting2")
```

```
    public List < Greeting > listaGreeting() {
```

```
        List < Greeting > lista = new ArrayList < Greeting > ();
```

```
        Greeting p = new Greeting(1,"Hola este es el saludo 1");
```

```
        lista.add(p);
```

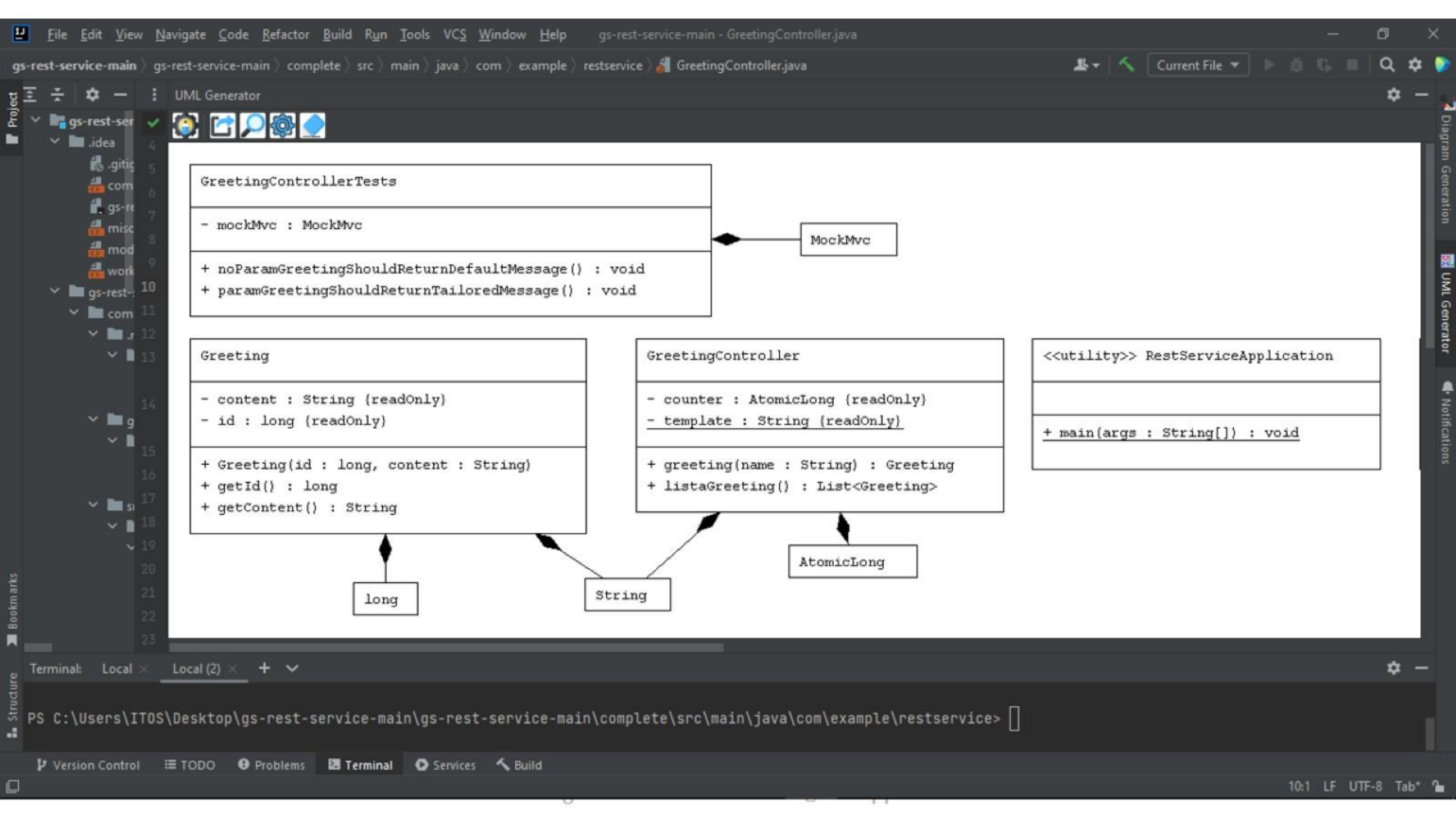
```
        Greeting p1 = new Greeting(2,"Hola este es el saludo 2");
```

```
        lista.add(p1);
```

```
        return lista;
```

```
    }
```

```
}
```

Gracias



Referencias

- <https://web.postman.co/home>
- <https://spring.io/>
- <https://www.freecodecamp.org/news/http-and-everything-you-need-to-know-about-it/>
- [https://www.tutorialspoint.com/spring boot/spring boot introduction.htm](https://www.tutorialspoint.com/spring_boot/spring_boot_introduction.htm)
- <https://stackabuse.com/controller-and-restcontroller-annotations-in-spring-boot/>
- <https://spring.io/guides/tutorials/rest/>