

# Configuración

# Índice

- 01** [Configuración de valores en Git](#)
- 02** [Configuración servidor de configuraciones](#)
- 03** [Configuración de clientes del servidor de configuraciones](#)

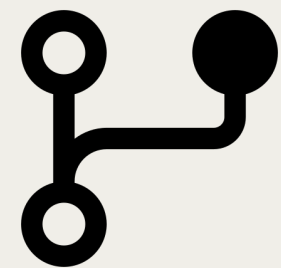


01

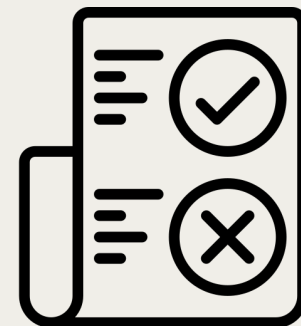
# Configuración de valores en Git

# Configuración de valores en Git

Las configuraciones que tengamos de nuestros microservicios estarán especificadas como archivos YML dentro de un repositorio GitHub. Allí existirá un archivo YML por cada microservicio que necesitemos configurar de forma externa. El uso de Git es una práctica habitual porque, al igual que con el código, permite:



**Control de versiones.**



**Establecer las reglas de aprobación y rechazo de forma automática.**











**Facilidad para analizar, aplicar y/o revertir cambios.**



**Trazabilidad de los cambios.**


En Git se visualizan los siguientes archivos:


- **application.yml:** contiene las configuraciones generales para todos los servicios.
- **dh-cuenta-service.yml:** contiene las configuraciones para el microservicio llamado “dh-cuenta-service”.
- **dh-transaccion-service.yml:** contiene las configuraciones para el microservicio llamado “dh-transaccion-service”

|  |                            |                  |
|--|----------------------------|------------------|
|  master ▾  1 branch  0 tags |                            |                  |
|  duccisg Update dh-cuenta-service-qa.yml ...  |                            |                  |
|    | README.md                  | Initial commit   |
|   | application.yml            | properties cloud |
|   | dh-cuenta-service.yml      | properties cloud |
|   | dh-transaccion-service.yml | properties cloud |

En este ejemplo, estamos agregando las propiedades:

- **server.port:** puerto de escucha del microservicio “dh-cuenta-service”.
- **message:** propiedad personalizada con un valor.

 **duccisg** Update dh-cuenta-service.yml ...

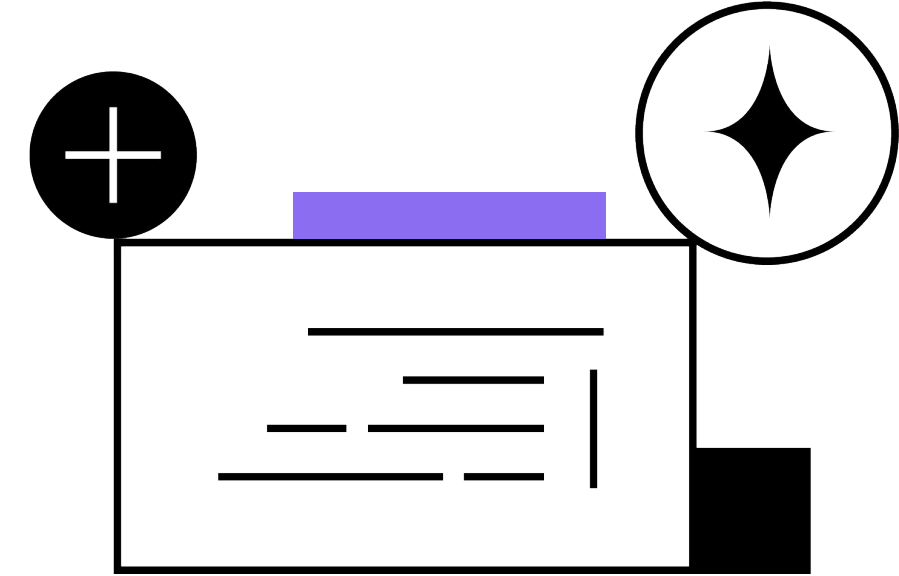
 1 contributor

4 lines (3 sloc) | 87 Bytes

```
1  server:
2    port: ${PORT:8889}
3
4  message: configuración custom con spring cloud config !
```

02

# Configuración servidor de configuraciones



Ahora que ya tenemos nuestro repositorio Git con las configuraciones que consideramos pertinentes, debemos crear nuestro servidor Cloud Config para que tome la base de Git y lo deje disponible por servicios REST para los microservicios que lo necesiten.  
Los pasos que tendremos que hacer son:

01

Agregar la dependencia de  
Spring Cloud Config

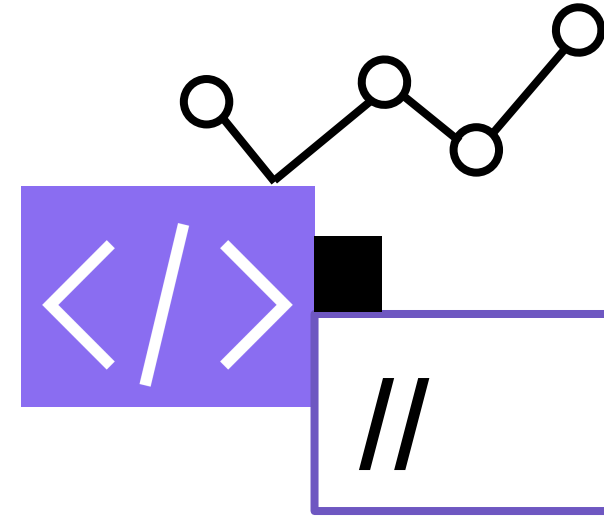
02

Configurar el microservicio  
como servidor de  
configuraciones

03

Configurar el servidor para  
tomar los valores de Git



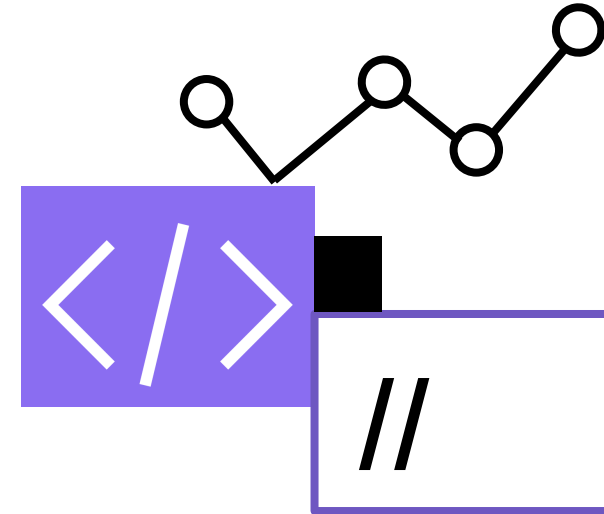


# 1) Agregar la dependencia de Spring Cloud Config

Para esto debemos agregar, en nuestro microservicio con Spring Boot, la siguiente dependencia dentro del **pom.xml**:

```
<dependency>
  <groupId>org.springframework.cloud</groupId>
  <artifactId>spring-cloud-config-server</artifactId>
</dependency>
```

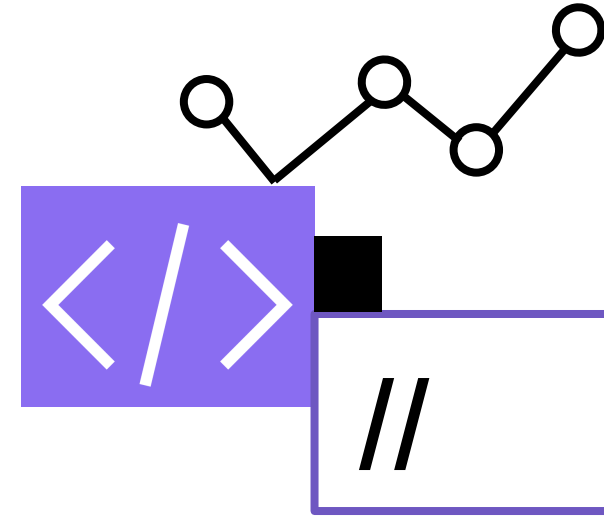
{JSON}



## 2) Configurar el microservicio como servidor de configuraciones

En nuestra clase de start-up de Spring Boot debemos especificar que dicha aplicación será un servidor de configuraciones mediante la anotación **@EnableConfigServer**:

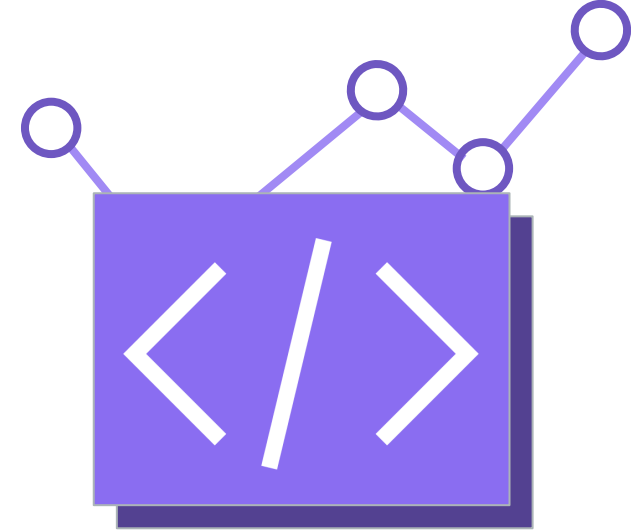
```
@EnableConfigServer
@SpringBootApplication
public class ConfigServerApplication {
    public static void main(String[] args)
    {SpringApplication.run(ConfigServerApplication.class, args);}
}
```



### 3) Configurar el servidor para tomar los valores de Git

En nuestro YML de aplicación debemos especificar el Git de origen de configuración para el servidor de configuraciones dentro de la clave: `spring.cloud.config.server.git.url`.

```
server:
  port: ${PORT:8888}
spring:
  application:
    name: config-server
  cloud:
    config:
      server:
        git:
          url: https://github.com/duccisg/spring-cloud-example
```



# Validación

Luego de estos tres pasos, podemos validar si hemos configurado correctamente el servidor mediante la consulta de alguna de las configuraciones de los servicios declarados vía HTTP. La consulta la realizaremos bajo el siguiente patrón:

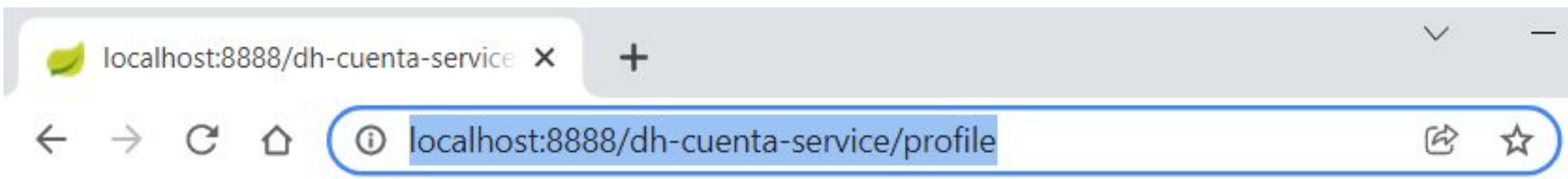
`http://{url-servidor-config}/{nombre-servicio}/{profile-name}`

Es la dirección `{url-servidor-config}` del servidor que proporcionará las configuraciones, bajo Spring Cloud Config

Es el nombre del microservicio del cual se consulta la configuración definida.

Establece el nombre del perfil de configuración a consultar. Este se utilizará junto al nombre del servicio para determinar las configuraciones a utilizar (más adelante veremos esto en profundidad).

Si ponemos en nuestro navegador la URL <http://localhost:8888/dh-cuenta-service/profile>, veremos el siguiente resultado:



```
{"name":"dh-cuenta-service","profiles":
["profile"],"label":null,"version":null,"state":null,"propertySources":
[{"name":"https://github.com/duccisg/spring-cloud-example/dh-cuenta-service.yml","source":
{"server.port":"${PORT:8889}","message":"configuración custom con spring cloud config !"}},
{"name":"https://github.com/duccisg/spring-cloud-example/application.yml","source":{"global-
message":"Configuration server is alive!"}}]}
```

03

# Configuración de clientes del servidor de configuraciones

# Para crear nuestro cliente de configuraciones de Spring Cloud Config debemos:

01

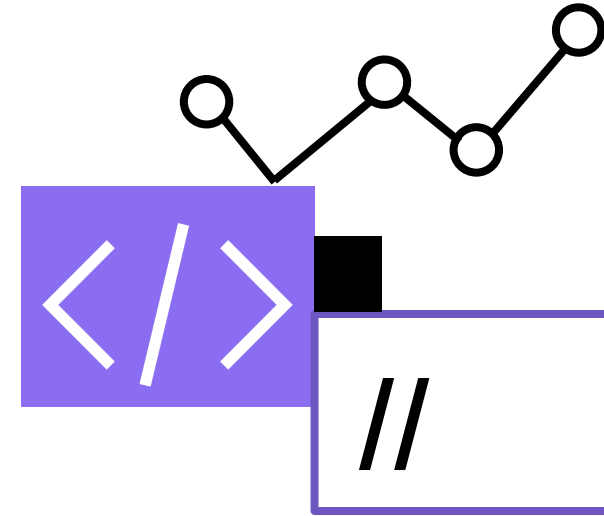
Agregar la dependencia de Spring Cloud Config

02

Especificar la ruta del servidor de configuraciones a utilizar

03

Utilizar las configuraciones obtenidas

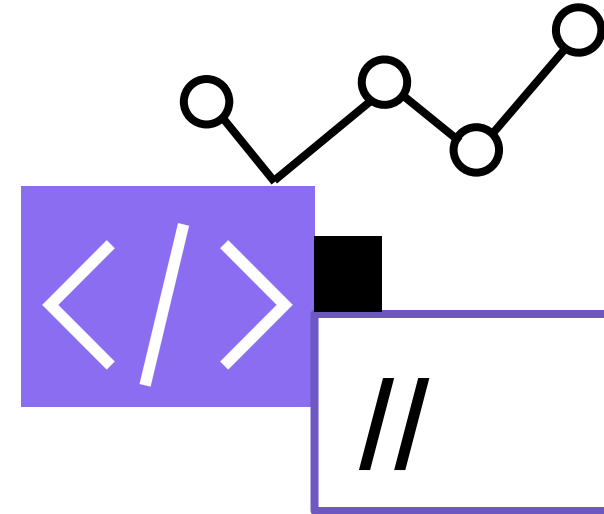


# 1) Agregar la dependencia de Spring Cloud Config

Para esto debemos agregar la siguiente dependencia dentro del **pom.xml**:

```
<dependency>  
  <groupId>org.springframework.cloud</groupId>  
  <artifactId>spring-cloud-config-starter-config</artifactId>  
</dependency>
```



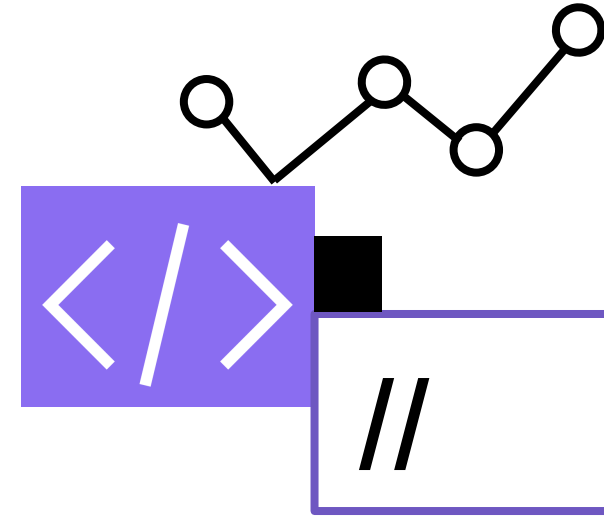


## 2) Especificar la ruta del servidor de configuraciones a utilizar

Debemos crear el archivo de configuraciones **bootstrap.yml** especificando el nombre del microservicio, nombre que se utilizará para obtener la configuración del servidor de acuerdo con los nombres de los servicios definidos en Git, como vimos al inicio del uso de este framework.

También dentro de este YML, estableceremos la URL del servidor de configuración:

```
spring:
  application:
    name: dh-cuenta-service
  cloud:
    config:
      url: ${CONFIG_SERVER:http://localhost:8888}
```



### 3) Utilizar las configuraciones obtenidas

Debemos agregar la dependencia de **spring-web** a nuestro proyecto, agregando un endpoint REST donde mostraremos las configuraciones que obtuvimos de nuestro servidor de configuraciones.

En nuestro **pom.xml** incorporamos:

```
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-config-starter-web</artifactId>
</dependency>
```

Luego, crearemos una clase con un endpoint inyectando los valores de configuración previamente vistos en la configuración de Git:

```
@RestController
class AccountService {
    @Value("${message}")
    private String message;
    @Value("${global-message}")
    private String globalMessage;

    @RequestMapping(method = RequestMethod.GET, path="service")
    public Map<String,String> message() {
        Map<String,String> response = new HashMap<>();

        response.put("message", message);
        response.put("global-message", globalMessage);

        return response;
    }
}
```

Por último, accedemos al servicio de “cuentas” de acuerdo con el puerto de uso definido en Git e ingresamos en el navegador a la URL <http://localhost:8889/service>.



¡Muchas gracias!