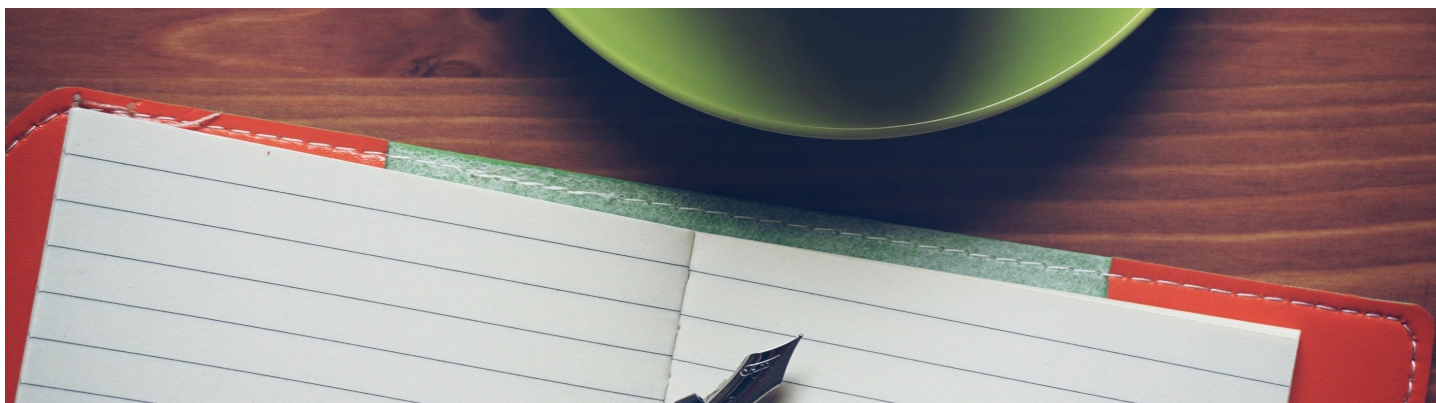








# devs4j

EL MEJOR SITIO WEB SOBRE PROGRAMACIÓN EN ESPAÑOL.

[HOME](#)[ABOUT](#)[CONTACT](#)

Anuncios



```
<?php find_developers( [ 'language' => 'PHP',  
    'specialty' => 'scaling', 'location' => ANYWHERE ] )
```

[APPLY](#)

[REPORT THIS AD](#)

# Spring framework 5 : Uso de la anotación @Prim ary

📅 [HACE 2 DÍAS](#)    💬 [DEJA UN COMENTARIO](#)

1 Vote

En el ejemplo anterior vimos como utilizar la anotación @Qualifier y como nos ayuda cuando tenemos multiples implementaciones de una interfaz. El problema de esa solución es cuando se utiliza @Autowired sin especificar el qualifier dado que Spring falla con el siguiente error:

```
Field userService in com.devs4j.qualifier.UserContr
```



Esto se resuelve fácilmente utilizando un qualifier o utilizando la anotación **@Primary**, esta anotación nos permite indicar el bean default a inyectar en caso de que no se especifique ningún qualifier, veamos el siguiente ejemplo:

## UserService.java

```
/**
 * @author raidentrance
 *
 */
public interface UserService {

    public boolean authenticate(String username

}
```

## UserServiceDatabaseImpl.java

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.context.annotation.Primary;
import org.springframework.stereotype.Service;

/**
 * @author raidentrance
 *
 */
@Service
@Primary
public class UserServiceDatabaseImpl implements UserService {

    private static Logger log = LoggerFactory.getLogger(UserServiceDatabaseImpl.class);

    @Override
    public boolean authenticate(String username
```

```
        log.info("authenticating against th  
        return false;  
    }  
  
}
```



#### UserServiceActiveDirectoryImpl.java

```
import org.slf4j.Logger;  
import org.slf4j.LoggerFactory;  
import org.springframework.stereotype.Service;  
  
/**  
 * @author raidentrance  
 *  
 */  
@Service  
public class UserServiceActiveDirectoryImpl implements  
  
    private static Logger log = LoggerFactory.g  
  
    @Override  
    public boolean authenticate(String username  
        log.info("authenticating against th  
        return false;  
    }  
  
}
```



Como podemos ver tenemos las siguientes clases:

- UserService : Interfaz que cuenta con 2 implementaciones
- UserServiceDatabaseImpl : Implementación de UserService que utiliza la anotación @Primary
- UserServiceActiveDirectoryImpl : Implementación de UserService

Una vez que definimos nuestra interfaz y sus implementaciones el siguiente paso será utilizarlo en otro bean, veamos el bean UserController :

```
import org.springframework.beans.factory.annotation
import org.springframework.stereotype.Controller;

/**
 * @author raidentrance
 *
 */
@Controller
public class UserController {

    @Autowired
    private UserService userService;

    public boolean authenticate(String username
        return userService.authenticate(use
    }
}
```

Como podemos ver es posible inyectar UserService sin especificar ningún qualifier sin recibir ningún error, esto es gracias a que definimos la anotación @Primary en la clase UserServiceDatabaseImpl.

Para estar al pendiente sobre nuestro contenido nuevo síguenos en nuestras redes sociales <https://www.facebook.com/devs4j/> (<https://www.facebook.com/devs4j/>) y <https://twitter.com/devs4j> (<https://twitter.com/devs4j>).

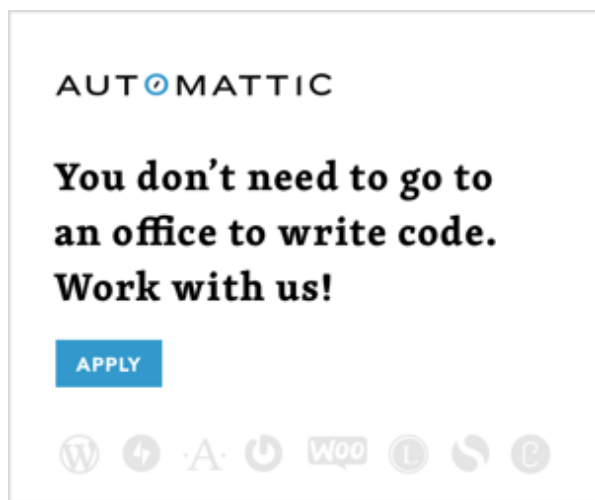
*Autor: Alejandro Agapito Bautista*

*Twitter: [@raidentrance](https://twitter.com/raidentrance)*

*(<https://geeksjavamexico.wordpress.com/mentions/raidentrance/>)*

*Contacto: [raidentrance@gmail.com](mailto:raidentrance@gmail.com)*

#### Anuncios



**AUTOMATIC**

**You don't need to go to an office to write code. Work with us!**

**APPLY**

Logos for various technologies: WordPress, Laravel, Angular, React, WooCommerce, Magento, Joomla, Drupal.

REPORT THIS AD



**Earn money from your WordPress site**



REPORT THIS AD

