Documentación código Inyección de Dependencias

Rodrigo Martínez Zambrano

La inyección de dependencias es un patrón de diseño que permite a un objeto recibir sus dependencias de una entidad externa en lugar de crearlas por sí mismo. Este patrón es muy útil para hacer que el código sea más modular, testable y fácil de mantener.

En este caso, vamos a implementar una aplicación simple que envía mensajes. Tendremos una interfaz *MessageService*, y las clases *EmailService*, *SMS_Service*, MessageProcessor que utiliza MessageService.

La interfaz MessageService define el método sendMessage que va a recibir dos String como parámetros, el mensaje y el medio por el que se manda.

```
//Se define la interfaz a implementar
public interface MessageService {

    void sendMessage(String message, String recipient);

}
```

Las clases EmailService y SMS_Service implementaran la interfaz y cada una desarrollará el método sendMessage como se requiera.

En este caso EmailService mandará un mensaje por medio de correo electrónico y SMS_Service lo hará por mensaje de texto con un número de celular.

```
//Clase que enviara un mensaje via mail
public class EmailService implements MessageService {

    @Override
    public void sendMessage(String message, String recipient) {
        System.out.println("Sending email to " + recipient + " with message: " + message);
    }
}
```

```
//Clase que enviara un mensaje vía SMS
public class SMS_Service implements MessageService{

    @Override
    public void sendMessage(String message, String recipient) {
        System.out.println("Sending SMS to " + recipient + " with message: " + message);
    }
}
```

La clase MessageProcessor se encarga de inicializar la inyección por medio del constructor recibiendo el tipo de servicio por el parámetro. Adicionalmente se encargará de enviar el mensaje correspondiente.

```java
//clase para realizar la inyección
public class MessageProcessor {
    private MessageService messageService;

    // Constructor para la inyección de dependencias
    public MessageProcessor(MessageService messageService) {
        this.messageService = messageService;
    }

    //Invocará el metodo de la interfaz que estará sobrescrito
    //por cada dependencia
    public void processMessage(String message, String recipient) {
        messageService.sendMessage(message, recipient);
    }
}
```

Finalmente, la clase Main será la encargada de ejecutar el código, como se observa en la imagen, primero se instancian los servicios, mail y sms, posteriormente se inyectan las dependencias en el parámetro de la instancia de MessageProcessor.

Finalmente dicho objeto se encarga de enviar el mensaje.

```java
public class Main {
    public static void main(String[] args) {
        // Crear la instancia de la implementación del servicio
        MessageService emailService = new EmailService();
        MessageService smsService = new SMS_Service();

        // Inyectar la dependencia en MessageProcessor a través del constructor
        MessageProcessor messageProcessorMail = new MessageProcessor(emailService);
        MessageProcessor messageProcessorSMS = new MessageProcessor(smsService);


        // Usar MessageProcessor para enviar un mensaje
        messageProcessorMail.processMessage("Hello, how are you?", "persona@gmail.com");
        messageProcessorSMS.processMessage("Hello Roy!","5512345678");
    }
}
```