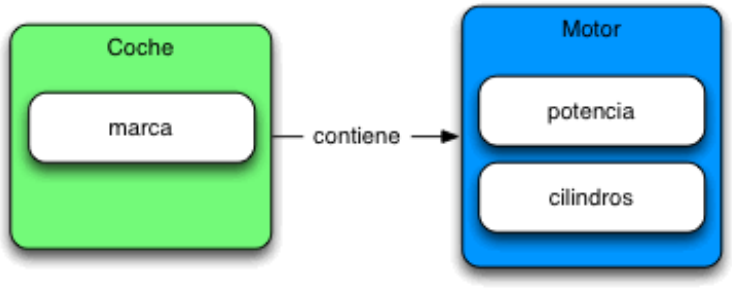
Eclipse y el concepto de Delegación

24 octubre, 2014 por Cecilio Álvarez Caules — 5 comentarios

Cuando programamos en Java en muchas ocasiones nos encontramos con la necesidad de usar el concepto de delegación . Este conceptos es muy habitual cuando tenemos estructuras de clase de composición. Es decir un objeto A contiene un Objeto B. Por ejemplo supongamos que tenemos el diagrama de clases de Coche y Motor.



Vamos a ver su código Java :

```java
package com.arquitecturajava;

public class Coche {

private String marca;
 private Motor motor;

public Motor getMotor() {
 return motor;
 }

public void setMotor(Motor motor) {
 this.motor = motor;
 }

public String getMarca() {
 return marca;
 }

public void setMarca(String marca) {
 this.marca = marca;
 }
```

```java
package com.arquitecturajava;

public class Motor {

 private int potencia;
 private int cilindros;
 public int getPotencia() {
 return potencia;
 }
 public void setPotencia(int potencia) {
 this.potencia = potencia;
 }
 public int getCilindros() {
 return cilindros;
 }
 public void setCilindros(int cilindros) {
 this.cilindros = cilindros;
 }
}
```

```java
package com.arquitecturajava;

public class Principal {

 public static void main(String[] args) {
 Coche c= new Coche();
 c.setMarca("toyota");
 Motor m= new Motor();
 m.setCilindros(6);
 m.setPotencia(100);
 c.setMotor(m);

 System.out.println(c.getMotor().getPotencia())

 }

}
```


POPULAR

Framework vs Libreria dos conceptos importantes

JDBC Prepared Statement y su manejo

Los Frameworks y su lado oscuro

Spring Boot JPA y su configuración

¿Qué es un Java Maven Artifact ?

Static Method vs instance method y su uso correcto

Java new String y la creación de objetos

Spring REST CORS y su configuración

Single Page Application y REST

Mi Nuevo Curso de Typescript

CONTACTO

contacto@arquitecturajava.com

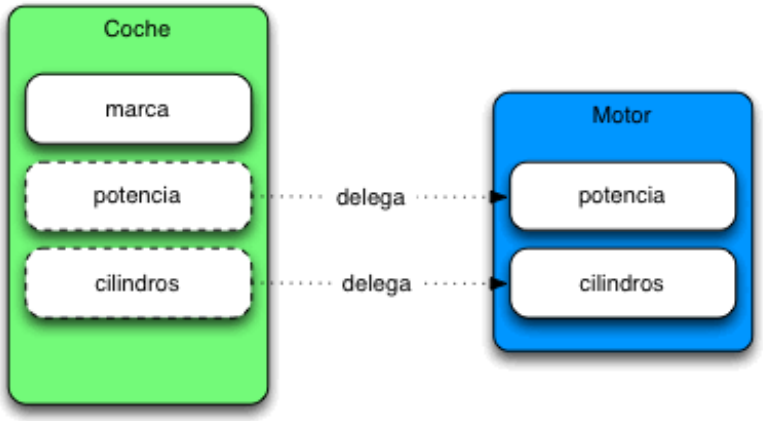
LO MAS LEIDO

¿Qué es Spring Boot?

Java Constructores this() y super()

El concepto de Delegación

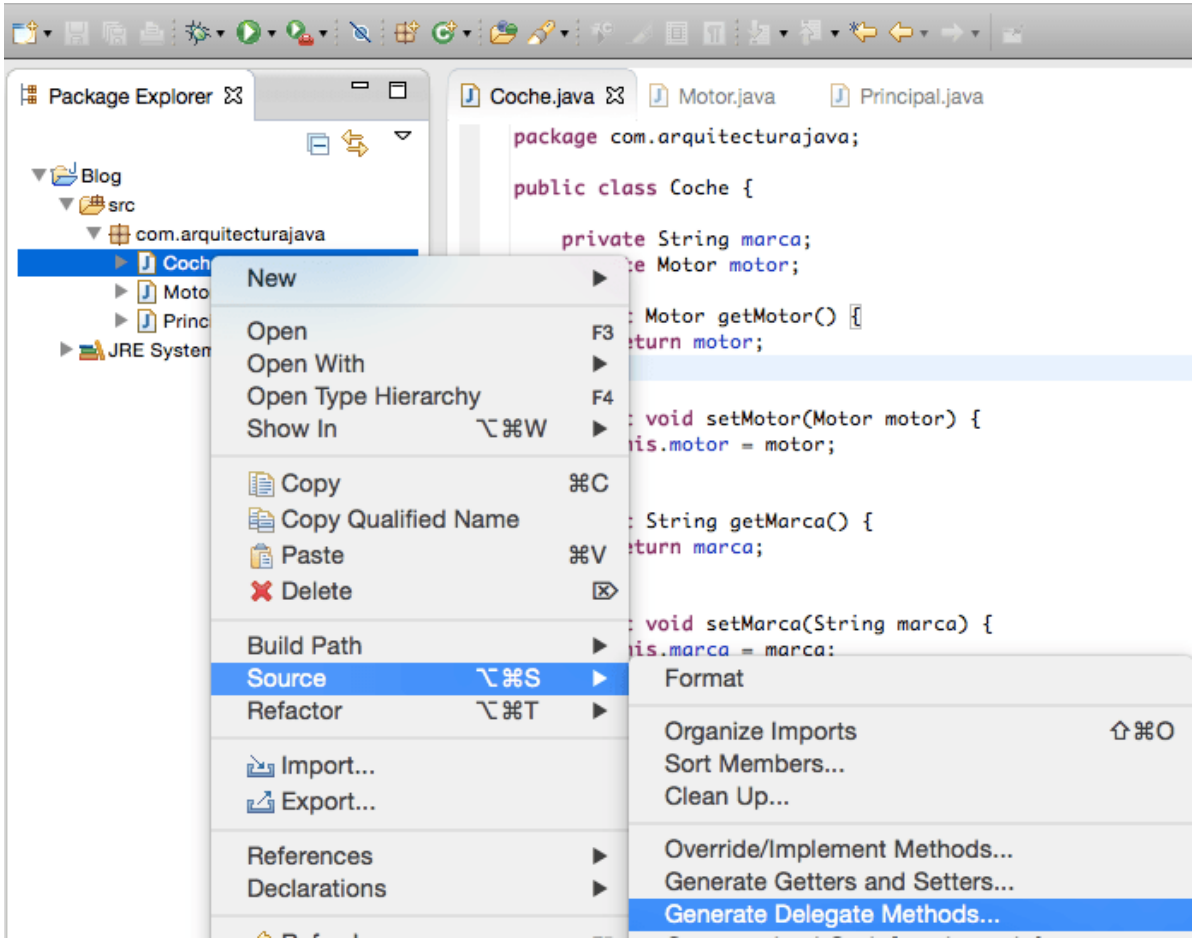
Como podemos ver acceder en nuestro código a la potencia es un poco enrevesado. Podemos apoyarnos en el concepto de delegación y generar nuevos métodos a nivel de la clase Coche para que "delegen" en los de motor y todo sea más sencillo.



Eclipse y Refactorings

Para construir estos nuevos métodos nos vamos a apoyar en el eclipse y en sus capacidades de refactoring. Nos posicionamos en la clase Coche y pulsamos boton derecho Source –>Generate Delegate

Methods.



Una vez seleccionada esta opción Eclipse nos permitirá seleccionar que métodos queremos generar como delegados. Marcamos los de potencia y cilindros que pertenecen al Motor.

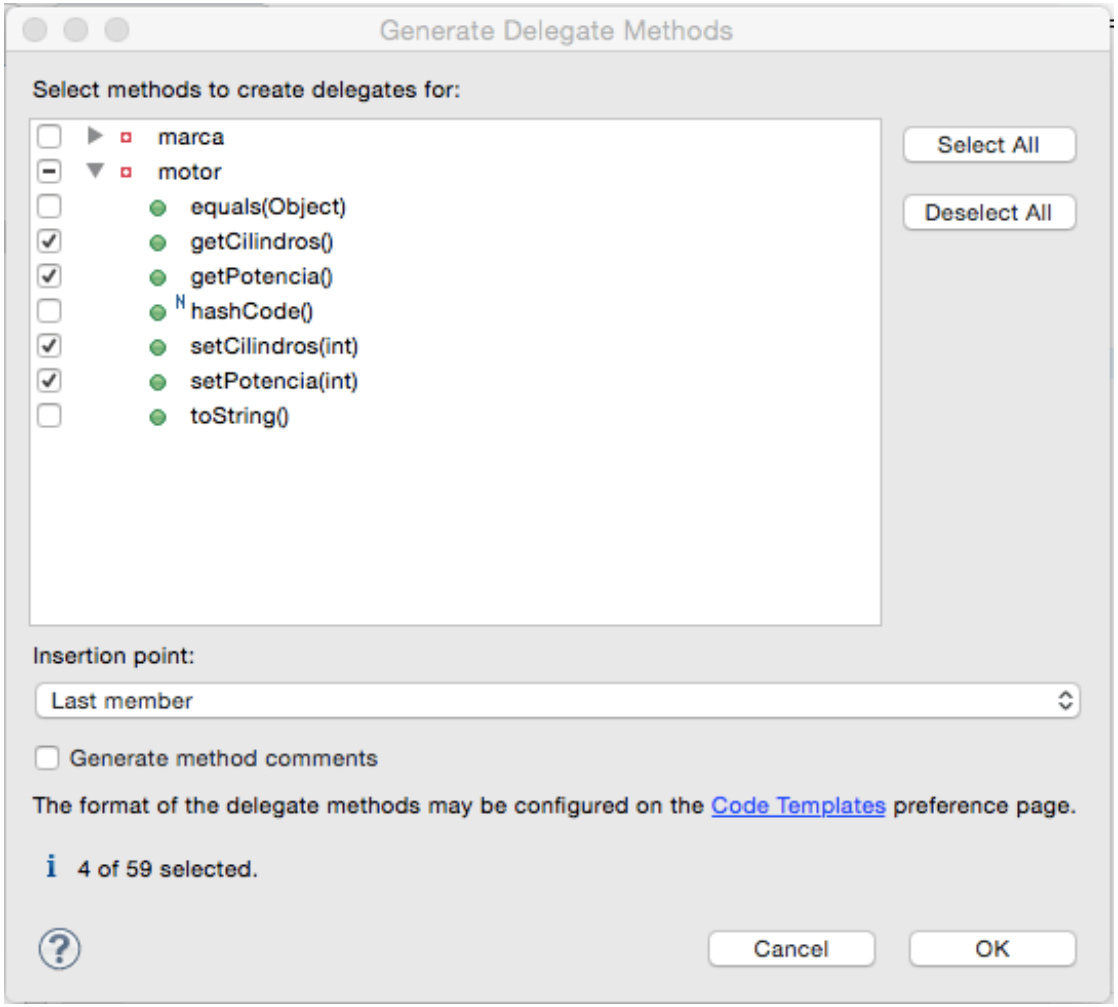
utilidad

¿ Que es REST ?

Angular ngFor la directiva y sus opciones

Java Stream forEach y colecciones

Entiendo los métodos de Java equals y hashcode



Eclipse automaticamente nos generará los nuevos métodos en la clase Coche:

```
public int getPotencia() {
return motor.getPotencia();
}

public void setPotencia(int potencia) {
motor.setPotencia(potencia);
}

public int getCilindros() {
return motor.getCilindros();
}

public void setCilindros(int cilindros) {
motor.setCilindros(cilindros);
}
```

Una vez hecho esto podemos modificar nuestro programa principal :

```
package com.arquitecturajava;

public class Principal {

public static void main(String[] args) {
Coche c= new Coche();
c.setMarca("toyota");
Motor m= new Motor();
m.setCilindros(6);
m.setPotencia(100);
c.setMotor(m);

System.out.println(c.getPotencia());

}

}
```

Hemos implementado el concepto de delegación de una forma

automática

Otros artículos relacionados: Eclipse y plantillas , Eclipse Utility Proyects , Eclipse y Organización

PDF

0 COMPART

Archivado en: Eclipse
Etiquetado como: Eclipse

Comentarios

pete dice
18 abril, 2017 en 20:07

Hola. Creo que en el párrafo titulado “El concepto de delegación” se te ha escapado una clase “Persona” donde debería poner “Coche”, seguramente de otro ejemplo práctico.

Responder

Cecilio Álvarez Caules dice
21 abril, 2017 en 8:16

gracias , corregido 🙂

Responder

Juan dice
27 octubre, 2014 en 10:10

Hola,

genial ejemplo. Esta es la forma de solucionar lo que dice la Ley de Demeter? Tenia entendido que no es recomendable nunca hacer mas de un .getXXX() y nose si con esto es la manera de evitarlo

Gracias!

Responder

Cecilio Álvarez Caules dice
27 octubre, 2014 en 15:48

Si porque si modificaras el constructor y pasaras la potencia etc el motor quedaría oculto

Responder

Trackbacks

Usando el patron factory - Arquitectura Java dice:
18 noviembre, 2014 a las 12:16
[...] Otros artículos relacionados: Singleton , Delegación ,Adaptadores [...]

Responder

Deja un comentario

Tu dirección de correo electrónico no será publicada. Los campos obligatorios están marcados con *

Comentario

Nombre *

Correo electrónico *

Web

PUBLICAR COMENTARIO

Este sitio usa Akismet para reducir el spam. Aprende cómo se procesan los datos de tus comentarios.