

Estás aquí: [Inicio](#) / [Java SE](#) / [Eclipse](#) / Eclipse y el concepto de Delegación

# 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.

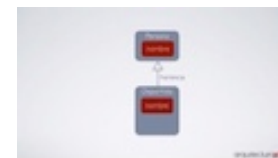
BUSCAR

Buscar en esta web

Mis Cursos de Java Gratuitos

Java Herencia

Java JDBC

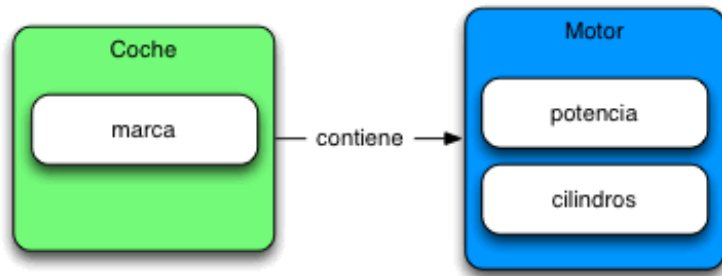


Servlets

Intro JPA



Mis Cursos de Java

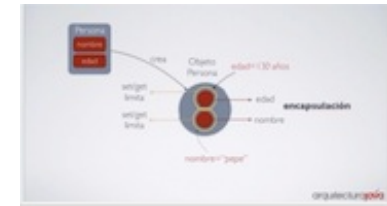


Vamos a ver su código Java :

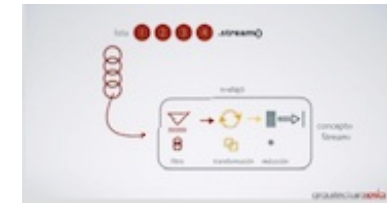
```

1  package com.arquitecturajava;
2
3  public class Coche {
4
5      private String marca;
6      private Motor motor;
7
8      public Motor getMotor() {
9          return motor;
10     }
11
12     public void setMotor(Motor motor) {
13         this.motor = motor;
14     }
15
16     public String getMarca() {
17         return marca;
18     }
19
20     public void setMarca(String marca) {
21         this.marca = marca;
22     }
  
```

## Programación Orientada a Objeto en Java



## Java APIS Core



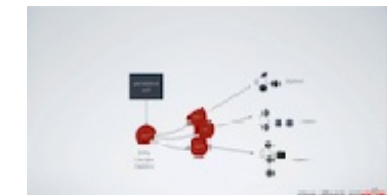
## Java Web



## Pack Java Core



## Arquitectura Java Solida con Spring



```

1 package com.arquitecturajava;
2
3 public class Motor {
4
5     private int potencia;
6     private int cilindros;
7     public int getPotencia() {
8         return potencia;
9     }
10    public void setPotencia(int potencia) {
11        this.potencia = potencia;
12    }
13    public int getCilindros() {
14        return cilindros;
15    }
16    public void setCilindros(int cilindros) {
17        this.cilindros = cilindros;
18    }
19 }

```

```

1 package com.arquitecturajava;
2
3 public class Principal {
4
5     public static void main(String[] args) {
6         Coche c= new Coche();
7         c.setMarca("toyota");
8         Motor m= new Motor();
9         m.setCilindros(6);
10        m.setPotencia(100);
11        c.setMotor(m);
12
13        System.out.println(c.getMotor().getPotencia())
14
15    }
16
17 }

```

## 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](mailto:contacto@arquitecturajava.com)

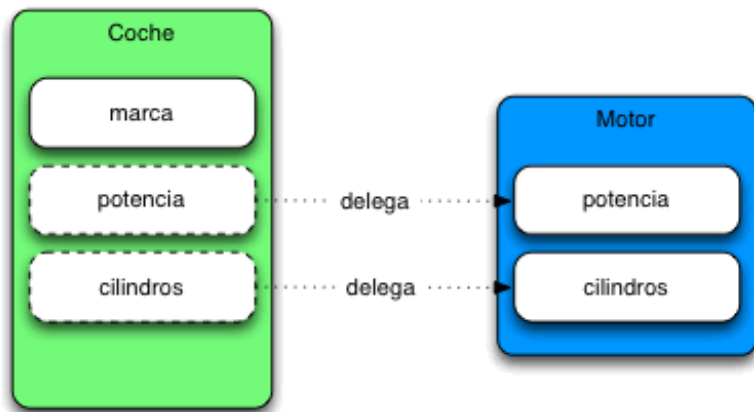
## LO MAS LEIDO

[¿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

---

Arquitecturas RESTful y agregados

---

Usando Java Session en aplicaciones web

---

Ejemplo de Java Singleton (Patrones y ClassLoaders)

---

Java Iterator vs ForEach

---

Introducción a Servicios REST

---

Java Assert librerías y enfoques

---

Comparando java == vs equals

---

Java Override y encapsulación

---

Usando el patron factory

---

REST JSON y Java

---

Ejemplo de JPA , Introducción (I)

---

¿Cuales son las certificaciones Java?

---

Uso de Java Generics (I)

---

¿Qué es Gradle?

---

¿Qué es un Microservicio?

---

HttpSessionListener un concepto importante

---

Mis Libros

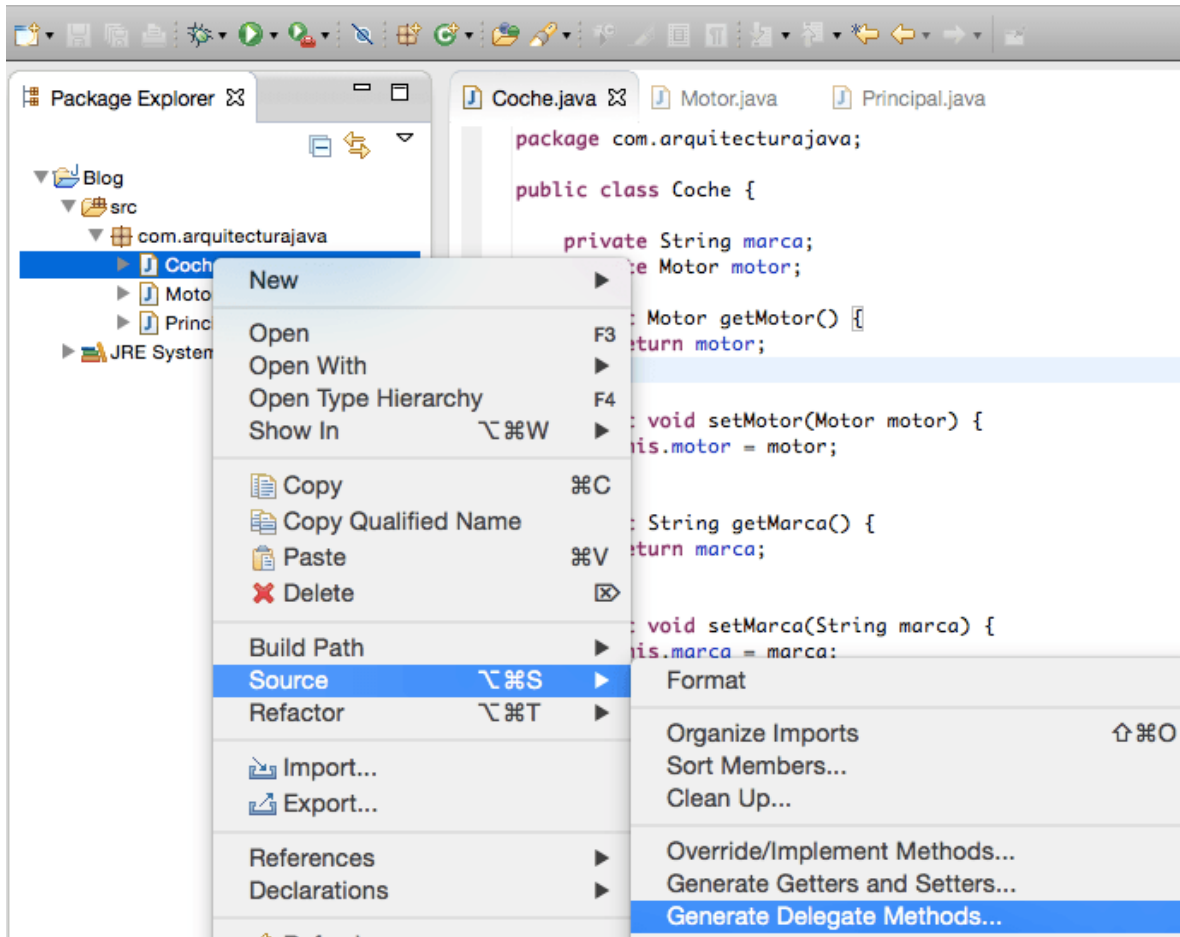
---

Spring MVC Configuración (I)

---

El patrón de inyección de dependencia y su

## 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.

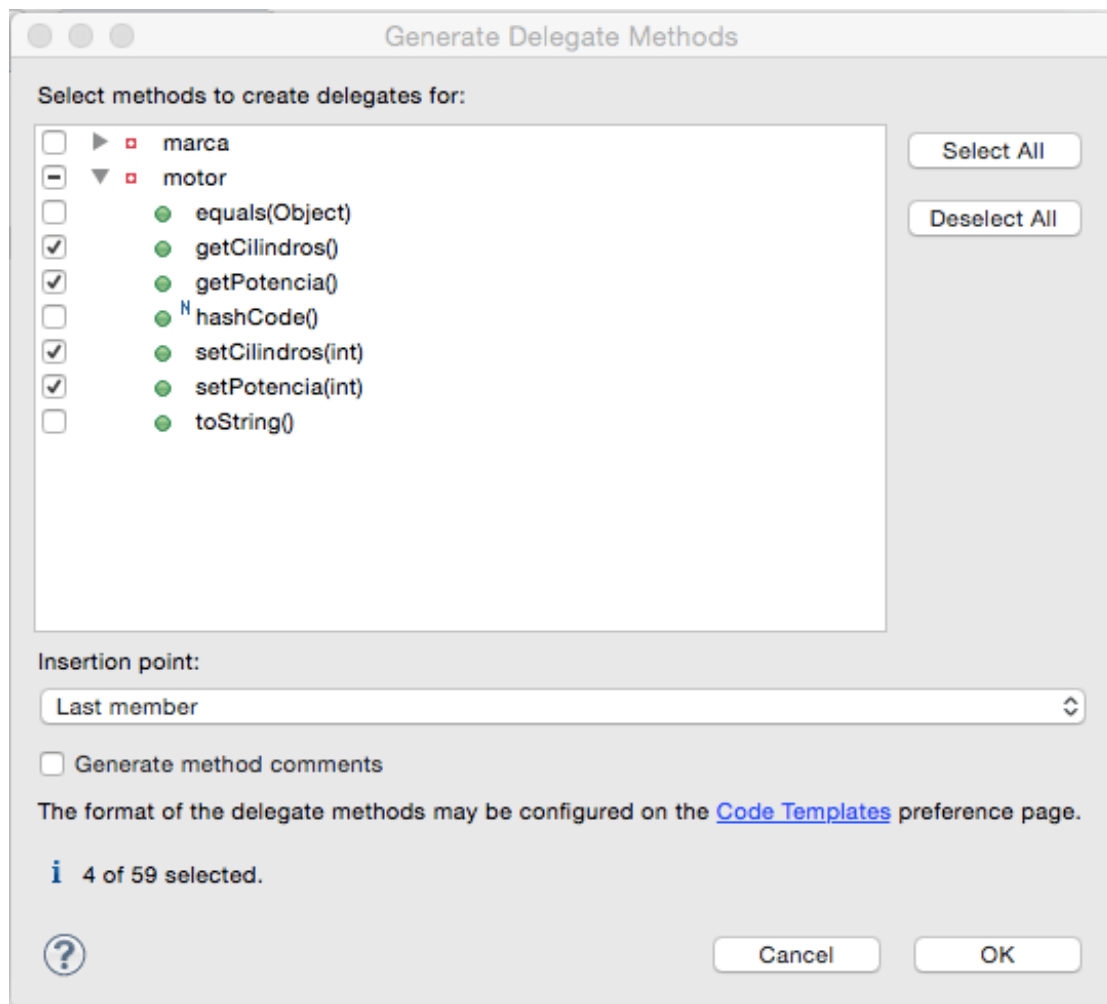
utilidad

¿ Que es REST ?

Angular ngFor la directiva y sus opciones

Java Stream forEach y colecciones

Entiendo los métodos de Java equals y hashCode



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

```

1  public int getPotencia() {
2  return motor.getPotencia();
3  }
4
5  public void setPotencia(int potencia) {
6  motor.setPotencia(potencia);
7  }
8
9  public int getCilindros() {
10 return motor.getCilindros();
11 }
12
13 public void setCilindros(int cilindros) {
14 motor.setCilindros(cilindros);
15 }

```

Una vez hecho esto podemos modificar nuestro programa principal :

```

1  package com.arquitecturajava;
2
3  public class Principal {
4
5  public static void main(String[] args) {
6  Coche c= new Coche();
7  c.setMarca("toyota");
8  Motor m= new Motor();
9  m.setCilindros(6);
10 m.setPotencia(100);
11 c.setMotor(m);
12
13 System.out.println(c.getPotencia());
14
15 }
16
17 }

```

Hemos implementado el concepto de delegación de una forma

automática

Otros artículos relacionados: [Eclipse y plantillas](#) , [Eclipse Utility Projects](#) , [Eclipse y Organización](#)

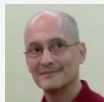


---

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.](#)

---

Este sitio web utiliza cookies para que usted tenga la mejor experiencia de usuario. Si continúa navegando está dando su consentimiento para la aceptación de las mencionadas cookies y la aceptación de nuestra [política de cookies](#), pinche el enlace para mayor información.

[plugin cookies](#)

ACEPTAR