Proxy Pattern

Благоја Евкоски

15 декември 2014

Што е proxy?

- PROXY == ЗАСТАПНИК
- Застапникот во објектно-ориентираниот свет ќе застапува некој друг објект со цел да го заштити на некој начин.
- Наместо да комуницира директно со целниот објект, клиентот ќе комуницира со неговиот застапник.

Интерфејсот ICar

```
1 package edu.finki.das.simpleproxy;
2
3 public interface ICar {
4
5     void drive(int age);
6
7 }
```

Класата Car

```
1  package edu.finki.das.simpleproxy;
2
3  public class Car implements ICar {
4
5      @Override
6      public void drive(int age) {
7          System.out.println("Car has been driven " +
8                  "by a " + age + " year old person.");
9      }
10
11 }
```

Класата Driver

```
1  package edu.finki.das.simpleproxy;
2
3  public class Driver {
4
5      private int age;
6      private ICar car;
7
8      public Driver(int age) {
9          this.age = age;
10     }
11
12     public void setCar(ICar car) {
13         this.car = car;
14     }
15
16     public void driveCar() {
17         car.drive(age);
18     }
19
20 }
```

Тест

```
1  package edu.finki.das.simpleproxy;
2
3  public class Test {
4
5      public static void main(String[] args) {
6          ICar car = new Car();
7          Driver driver = new Driver(15);
8          driver.setCar(car);
9          driver.driveCar();
10     }
11
12 }
```

Излез:
Car has been driven by a 15 year old person.

Класата - застапник

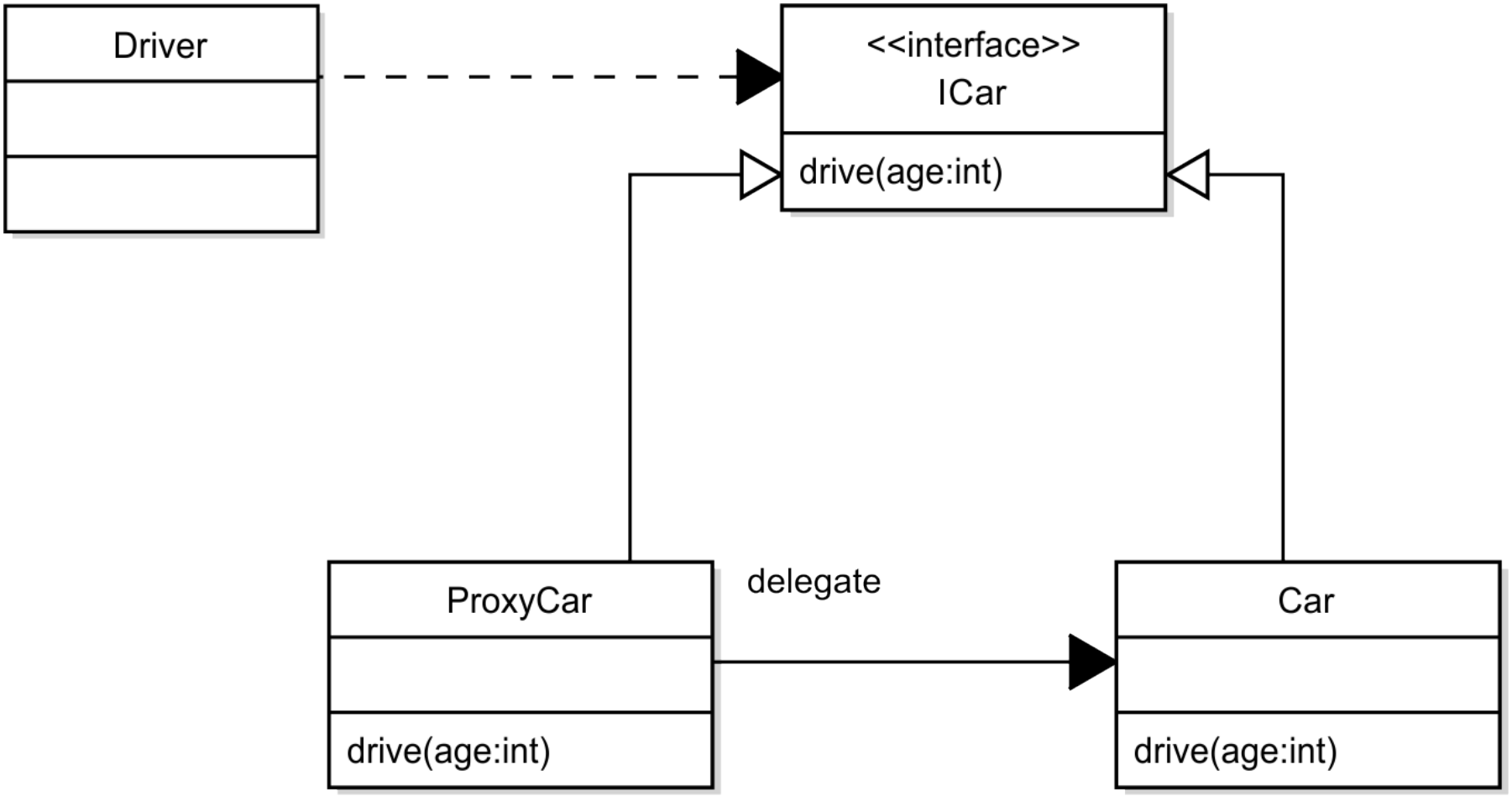
```java
package edu.finki.das.simpleproxy;

public class ProxyCar implements ICar {

    private ICar realCar;

    public ProxyCar() {
        realCar = new Car();
    }

    @Override
    public void drive(int age) {
        realCar.drive(age);
    }

}
```

Тест класата... повторно

```
1  package edu.finki.das.simpleproxy;
2
3  public class Test {
4
5      public static void main(String[] args) {
6          ICar car = new ProxyCar();
7          Driver driver = new Driver(15);
8          driver.setCar(car);
9          driver.driveCar();
10     }
11
12 }
```

Излез:
Car has been driven by a 15 year old person.

Дијаграм


Driver
<<interface>>
ICar
drive(age:int)
ProxyCar
drive(age:int)
delegate
Car
drive(age:int)

Промена кај застапникот

```
1  package edu.finki.das.simpleproxy;
2
3  public class ProxyCar implements ICar {
4
5      private ICar realCar;
6
7      public ProxyCar() {
8          realCar = new Car();
9      }
10
11     @Override
12     public void drive(int age) {
13         if (age < 16)
14             System.out.println("The driver is " +
15                     " too young to drive...");
16         else
17             realCar.drive(age);
18     }
19
20 }
```

Повторно истиот тест... со различен излез

```
1  package edu.finki.das.simpleproxy;
2
3  public class Test {
4
5      public static void main(String[] args) {
6          ICar car = new ProxyCar();
7          Driver driver = new Driver(15);
8          driver.setCar(car);
9          driver.driveCar();
10     }
11
12 }
```

Излез:
The driver is too young to drive...

3-те најбитни подвидови на proxy шаблонот

Proxy шаблонот се користи за создавање на застапник објект кој го контролира пристапот кон некој друг објект, кој може да биде:

- Скап за создавање (Virtual Proxy)
- Во потреба од заштита од пристап (Protection Proxy)
- Оддалечен, на друга машина (Remote Proxy)

Virtual Proxy

Виртуелниот застапник застапува објект кој е скап за креирање. Тој често го одложува неговото создавање сè додека тој објект е навистина потребен. Откако ќе го креира, застапникот ја игра улогата на делегат за сите барања кон вистинскиот објект.

Интерфејсот Image

```
1 package edu.finki.das.virtualproxy;
2
3 public interface Image {
4
5     public void display();
6
7 }
```

Класата RealImage

```
1  package edu.finki.das.virtualproxy;
2
3  public class RealImage implements Image {
4
5      private String filename;
6
7      public RealImage(String filename) {
8          this.filename = filename;
9          loadImageFromDisk();
10     }
11
12     private void loadImageFromDisk() {
13         System.out.println("Loading " + filename);
14     }
15
16     @Override
17     public void display() {
18         System.out.println("Displaying " + filename + "\n");
19     }
20
21 }
```

Класата ProxyImage

```
1  package edu.finki.das.virtualproxy;
2
3  public class ProxyImage implements Image {
4
5      private Image image;
6      private String filename;
7
8      public ProxyImage(String filename) {
9          this.filename = filename;
10     }
11
12     private void displayLoadingAnimation() {
13         System.out.println("Displaying loading animation");
14     }
15
16     @Override
17     public void display() {
18         if (image == null) {
19             displayLoadingAnimation();
20             image = new RealImage(filename);
21         }
22         image.display();
23     }
24
25 }
```

Тест

```
1  package edu.finki.das.virtualproxy;
2
3  public class Test {
4
5      public static void main(String[] args) {
6          Image image1 = new ProxyImage("HiRes_10MB_Photo1");
7          Image image2 = new ProxyImage("HiRes_10MB_Photo2");
8
9          image1.display();
10         image1.display();
11
12         image2.display();
13         image2.display();
14
15         image1.display();
16     }
17
18 }
```

Излез

```
<terminated> Test (13) [Java Application] C:\Program Files\Java\jre6\bin\javaw.exe
Displaying loading animation
Loading HiRes_10MB_Photo1
Displaying HiRes_10MB_Photo1

Displaying HiRes_10MB_Photo1

Displaying loading animation
Loading HiRes_10MB_Photo2
Displaying HiRes_10MB_Photo2

Displaying HiRes_10MB_Photo2

Displaying HiRes_10MB_Photo1
```

Protection Proxy

Застапникот-заштитник го контролира пристапот до ресурсите на објектот кој го застапува. Се користи кај системи со различни профили на корисници, кога е потребно да се авторизира пристап до различни делови од системот во зависност од корисничкиот профил.

Интерфејсот IPerson

```
1 package edu.finki.das.protectionproxy;
2
3 public interface IPerson {
4
5     public void setName(String name);
6
7     public void setHotOrNotRating(int hotOrNotRating);
8
9 }
```

Класата Person

```
1  package edu.finki.das.protectionproxy;
2
3  public class Person implements IPerson {
4
5      private String name;
6      private int ratingSum = 0;
7      private int ratingCount = 0;
8
9      public Person(String name) {
10         this.name = name;
11     }
12
13     @Override
14     public void setName(String name) {
15         this.name = name;
16     }
17
18     @Override
19     public void setHotOrNotRating(int hotOrNotRating) {
20         ratingSum += hotOrNotRating;
21         ratingCount++;
22     }
23
24     public int getHotOrNotRating() {
25         if (ratingCount == 0)
26             return 0;
27         else
28             return (int) Math.ceil((double) ratingSum / ratingCount);
29     }
30
31 }
```

Класата ProxyOwner

```
1  package edu.finki.das.protectionproxy;
2
3  public class ProxyOwner implements IPerson {
4
5      private IPerson realPerson;
6
7      public ProxyOwner(IPerson realPerson) {
8          this.realPerson = realPerson;
9      }
10
11     @Override
12     public void setName(String name) {
13         System.out.println("Setting name to " + name);
14         realPerson.setName(name);
15     }
16
17     @Override
18     public void setHotOrNotRating(int hotOrNotRating) {
19         System.out.println("You cannot set your own rating...");
20     }
21
22 }
```

Класата ProxyNonOwner

```
1  package edu.finki.das.protectionproxy;
2
3  public class ProxyNonOwner implements IPerson {
4
5      private IPerson realPerson;
6
7      public ProxyNonOwner(IPerson realPerson) {
8          this.realPerson = realPerson;
9      }
10
11     @Override
12     public void setName(String name) {
13         System.out.println("You cannot set the name "
14                 + "of another person...");
15     }
16
17     @Override
18     public void setHotOrNotRating(int hotOrNotRating) {
19         System.out.println("Setting rating to " + hotOrNotRating);
20         realPerson.setHotOrNotRating(hotOrNotRating);
21     }
22
23 }
```

Тест

```
1  package edu.finki.das.protectionproxy;
2
3  public class Test {
4
5      public static void main(String[] args) {
6          Person person = new Person("Brad Pitt");
7
8          ProxyOwner owner = new ProxyOwner(person);
9          owner.setName("William Bradley Pitt");
10         owner.setHotOrNotRating(10);
11         System.out.println();
12
13         ProxyNonOwner nonOwner = new ProxyNonOwner(person);
14         nonOwner.setName("Brad Pitt");
15         nonOwner.setHotOrNotRating(10);
16     }
17
18 }
```

Излез

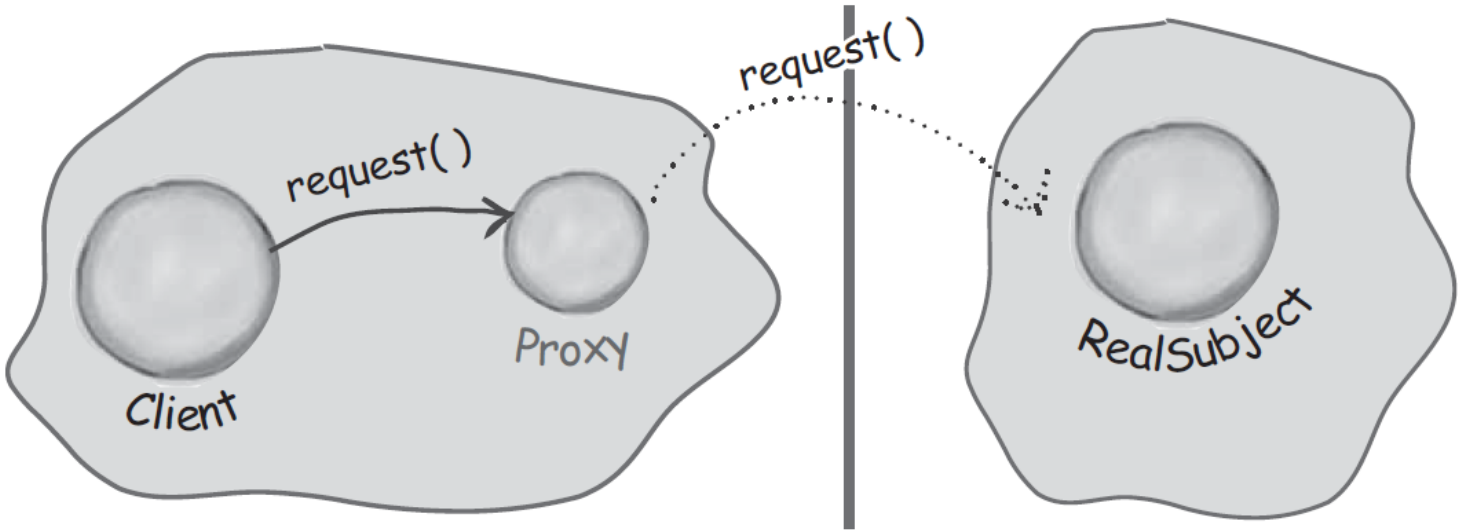
```
<terminated> Test (14) [Java Application] C:\Program Files\Java\jre6\bin\javaw.exe
Setting name to William Bradley Pitt
You cannot set your own rating...

You cannot set the name of another person...
Setting rating to 10
```

Remote Proxy

Овој тип на застапник се однесува како локална репрезентација на објект кој живее во друга Java виртуелна машина. Повик на метод кај застапникот резултира со:

1. Пренос на повикот преку мрежа
2. Разрешување на повикот во оддалечената машина
3. Враќање на резултатот назад на застапникот, и преку застапникот до клиентот

Илустрација



Што е RMI?

- RMI е акроним од Remote Method Invocation
- Ни овозможува да пронајдеме објекти кои живеат во друга Java виртуелна машина и да ги повикуваме нивните методи
- Функционалностите доаѓаат во пакетот java.rmi
- RMI ги гради помошните класи, без програмерот да мора да пишува код за комуникацијата преку мрежа

java.lang.reflect

- Java нуди поддршка за изведба на застапник шаблонот во пакетот java.lang.reflect
- Овој пакет овозможува креирање на застапник on the fly
- Поради тоа што застапник-класата се генерира за време на извршување, оваа технологија се нарекува dynamic proxy (или динамички застапник)

Користена литература

1. Freeman, Eric, Elisabeth Robson, Bert Bates, and Kathy Sierra. Head first design patterns. "O'Reilly Media, Inc. 2004.
2. Wikipedia contributors, "Proxy pattern,"Wikipedia, The Free Encyclopedia (accessed December 14, 2014)
3. Wikipedia contributors, "Java remote method invocation,"Wikipedia, The Free Encyclopedia (accessed December 14, 2014)