# Design Patterns - Prototype Pattern

Prototype pattern refers to creating duplicate object while keeping performance in mind. This type of design pattern comes under creational pattern as this pattern provides one of the best ways to create an object.

原型模式是指创建对象的副本，同时保持性能。这种类型的设计模式是创建型设计模式下创建对象最好的方式之一。

译注：原型模式为何会对性能有所提高？原型模式的本质就是对对象进行克隆，创建对象的副本，在创建对象以及初始化操作比较复杂的情况下，原型模式通过克隆原对象的运行时状态，减少了初始化，提高性能。

http://www.jb51.net/article/48201.htm

http://blog.csdn.net/zhengzhb/article/details/7393528

This pattern involves implementing a prototype interface which tells to create a clone of the current object. This pattern is used when creation of object directly is costly. For example, an object is to be created after a costly database operation. We can cache the object, returns its clone on next request and update the database as and when needed thus reducing database calls.

这个模式需要实现一个原型接口，并通知它创建一个当前对象的一个副本。这种模式在直接创建一个对象代价非常高昂的时候使用。例如，在执行代价较高的数据库操作后创建对象。我们可以缓存对象，在下次请求时返回该对象的一个副本，并更新数据库从而减少数据库的调用。

Implementation

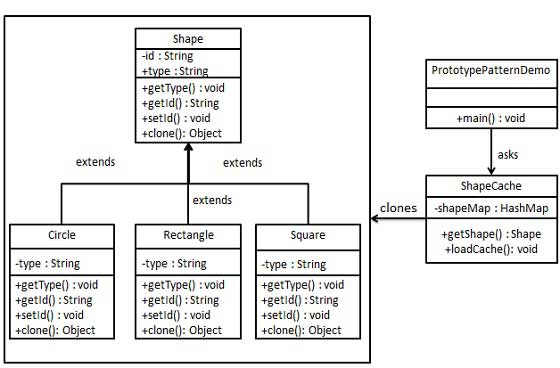
实现

We're going to create an abstract class *Shape* and concrete classes extending the *Shape* class. A class *ShapeCache* is defined as a next step which stores shape objects in a *Hashtable* and returns their clone when requested.

*PrototypPatternDemo*, our demo class will use *ShapeCache* class to get a *Shape*object.

我们将创建一个抽象的Shape类和继承这个抽象类的子类Shape。接下来声明一个类ShapeCache，该类把Shape对象在存储在HashTable中，当被请求时返回它们的副本。

PrototypPatternDemo，我们的democlass将使用ShapeCache类来获得Shape对象。



Step 1

第一步

Create an abstract class implementing *Clonable* interface.

创建一个抽象类Shape实现Cloneable接口。

*Shape.java*

public abstract class Shape implements Cloneable {

private String id;

protected String type;

abstract void draw();

public String getType(){

return type;

}

public String getId() {

return id;

}

public void setId(String id) {

this.id = id;

}

public Object clone() {

Object clone = null;

try {

clone = super.clone();

} catch (CloneNotSupportedException e) {

e.printStackTrace();

}

return clone;

}

}

Step 2

Create concrete classes extending the above class.

创建具体类继承Shape类。

*Rectangle.java*

public class Rectangle extends Shape {

public Rectangle(){

type = "Rectangle";

}

@Override

public void draw() {

System.out.println("Inside Rectangle::draw() method.");

}

}

*Square.java*

public class Square extends Shape {

public Square(){

type = "Square";

}

@Override

public void draw() {

System.out.println("Inside Square::draw() method.");

}

}

*Circle.java*

public class Circle extends Shape {

public Circle(){

type = "Circle";

}

@Override

public void draw() {

System.out.println("Inside Circle::draw() method.");

}

}

Step 3

Create a class to get concrete classes from database and store them in a*Hashtable*.

创建一个类从数据库中得到具体类，并且把它们存储在一个HashTable中。

*ShapeCache.java*

import java.util.Hashtable;

public class ShapeCache {

private static Hashtable<String, Shape> shapeMap = new Hashtable<String, Shape>();

public static Shape getShape(String shapeId) {

Shape cachedShape = shapeMap.get(shapeId);

return (Shape) cachedShape.clone();

}

// for each shape run database query and create shape

// shapeMap.put(shapeKey, shape);

// for example, we are adding three shapes

public static void loadCache() {

Circle circle = new Circle();

circle.setId("1");

shapeMap.put(circle.getId(),circle);

Square square = new Square();

square.setId("2");

shapeMap.put(square.getId(),square);

Rectangle rectangle = new Rectangle();

rectangle.setId("3");

shapeMap.put(rectangle.getId(), rectangle);

}

}

Step 4

*PrototypePatternDemo* uses *ShapeCache* class to get clones of shapes stored in a *Hashtable*.

PrototypePatternDemo 使用 ShapeCache 类从HashTable中获得诸多图形的克隆副本。

*PrototypePatternDemo.java*

public class PrototypePatternDemo {

public static void main(String[] args) {

ShapeCache.loadCache();

Shape clonedShape = (Shape) ShapeCache.getShape("1");

System.out.println("Shape : " + clonedShape.getType());

Shape clonedShape2 = (Shape) ShapeCache.getShape("2");

System.out.println("Shape : " + clonedShape2.getType());

Shape clonedShape3 = (Shape) ShapeCache.getShape("3");

System.out.println("Shape : " + clonedShape3.getType());

}

}

Step 5

Verify the output.

校验输出

Shape : Circle

Shape : Square

Shape : Rectangle