转载：<https://www.cnblogs.com/yunche/p/9530927.html>

[java创建对象的四种方式](https://www.cnblogs.com/yunche/p/9530927.html)

**1.使用new创建对象**

　　使用new关键字创建对象应该是最常见的一种方式，但我们应该知道，使用new创建对象会增加耦合度。无论使用什么框架，都要减少new的使用以降低耦合度。

[复制代码](javascript:void(0);)

package yunche.test;

/\*\*

\* @ClassName: Hello

\* @Description: 待创建的类

\* @author: yunche

\* @date: 2018/08/24

\*/

public class Hello

{

public void sayWorld()

{

System.out.println("Hello world!");

}

}

[复制代码](javascript:void(0);)

[复制代码](javascript:void(0);)

package yunche.test;

/\*\*

\* @ClassName: NewClass

\* @Description: 使用new关键字创建对象

\* @author: yunche

\* @date: 2018/08/24

\*/

public class NewClass

{

public static void main(String[] args)

{

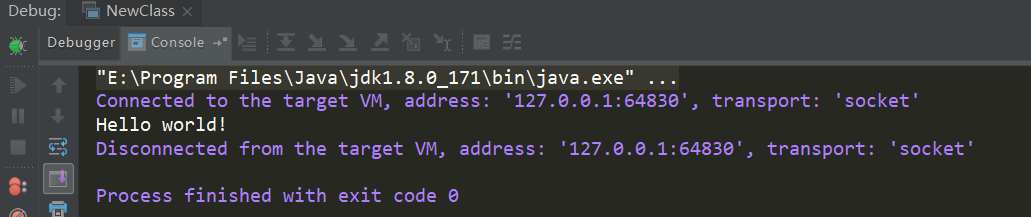
Hello h = new Hello();

h.sayWorld();

}

}

[复制代码](javascript:void(0);)



**2.使用反射的机制创建对象**

**使用Class类的newInstance方法**

　　Hello类的代码不变，NewClass类的代码如下：

[复制代码](javascript:void(0);)

package yunche.test;

/\*\*

\* @ClassName: NewClass

\* @Description: 使用Class类的newInstance方法

\* @author: yunche

\* @date: 2018/08/24

\*/

public class NewClass

{

public static void main(String[] args)

{

try

{

Class heroClass = Class.forName("yunche.test.Hello");

Hello h =(Hello) heroClass.newInstance();

h.sayWorld();

}

catch (ClassNotFoundException e)

{

e.printStackTrace();

}

catch (IllegalAccessException e)

{

e.printStackTrace();

}

catch (InstantiationException e)

{

e.printStackTrace();

}

}

}

[复制代码](javascript:void(0);)

**使用Constructor类的newInstance方法**

[复制代码](javascript:void(0);)

package yunche.test;

import java.lang.reflect.Constructor;

import java.lang.reflect.InvocationTargetException;

/\*\*

\* @ClassName: NewClass

\* @Description: 使用Constructor类的newInstance方法

\* @author: yunche

\* @date: 2018/08/24

\*/

public class NewClass

{

public static void main(String[] args)

{

try

{

//获取类对象

Class heroClass = Class.forName("yunche.test.Hello");

//获取构造器

Constructor constructor = heroClass.getConstructor();

Hello h =(Hello) constructor.newInstance();

h.sayWorld();

}

catch (NoSuchMethodException e)

{

e.printStackTrace();

}

catch (InvocationTargetException e)

{

e.printStackTrace();

}

catch (IllegalAccessException e)

{

e.printStackTrace();

}

catch (InstantiationException e)

{

e.printStackTrace();

}

catch (ClassNotFoundException e)

{

e.printStackTrace();

}

}

}

[复制代码](javascript:void(0);)

**3.采用clone**

　　clone时，需要已经有一个分配了内存的源对象，创建新对象时，首先应该分配一个和源对象一样大的内存空间。

　　要调用clone方法需要实现*Cloneable*接口，由于clone方法是protected的，所以修改Hello类。

[复制代码](javascript:void(0);)

package yunche.test;

/\*\*

\* @ClassName: Hello

\* @Description: 待创建的类

\* @author: yunche

\* @date: 2018/08/24

\*/

public class Hello implements Cloneable

{

public void sayWorld()

{

System.out.println("Hello world!");

}

public static void main(String[] args)

{

Hello h1 = new Hello();

try

{

Hello h2 = (Hello)h1.clone();

h2.sayWorld();

}

catch (CloneNotSupportedException e)

{

e.printStackTrace();

}

}

}

[复制代码](javascript:void(0);)

**4.采用序列化机制**

　　使用序列化时，要实现实现Serializable接口，将一个对象序列化到磁盘上，而采用反序列化可以将磁盘上的对象信息转化到内存中。

[复制代码](javascript:void(0);)

package yunche.test;

import java.io.\*;

/\*\*

\* @ClassName: Serialize

\* @Description: 序列化与反序列化对象

\* @author: yunche

\* @date: 2018/08/24

\*/

public class Serialize

{

public static void main(String[] args)

{

Hello h = new Hello();

//准备一个文件用于存储该对象的信息

File f = new File("hello.obj");

try(FileOutputStream fos = new FileOutputStream(f);

ObjectOutputStream oos = new ObjectOutputStream(fos);

FileInputStream fis = new FileInputStream(f);

ObjectInputStream ois = new ObjectInputStream(fis)

)

{

//序列化对象，写入到磁盘中

oos.writeObject(h);

//反序列化对象

Hello newHello = (Hello)ois.readObject();

//测试方法

newHello.sayWorld();

}

catch (FileNotFoundException e)

{

e.printStackTrace();

}

catch (IOException e)

{

e.printStackTrace();

}

catch (ClassNotFoundException e)

{

e.printStackTrace();

}

}

}