Spring 入门

Spring 中集成： → 完全面向接口的设计

Ioc：控制反转

AOP：面向方面编程

Spring 管理 Hibernate 以及事务

学习 Spring 之前必须关于对象的产生问题

重要概念：接口的概念 → 工厂设计

JAVA 回顾：

对于产生一个对象 JAVA 中有那几种常见的做法？（**针对接口操作**）

- 方法一：直接用 new 的方式开辟对象：类 对象 = new 类()；

  接口类型 对象名称 = new 接口子类()；→ 在客户端调用时必须明确的知道有那几个子类，直接造成了一个重要缺点：两个程序之间紧密耦合

```
package cn.mldn.lxh.demo01;


public interface Fruit {

   // 吃水果

   public void eat() ;

}
```

```
package cn.mldn.lxh.demo01;


public class Apple implements Fruit {


   public void eat() {

      System.out.println("吃苹果。。。") ;
```

```
    }


}
```

```
package cn.mldn.lxh.demo01;


public class Orange implements Fruit {


    public void eat() {
        System.out.println("吃橘子。。。") ;
    }


}
```

```
package cn.mldn.lxh.demo01;


public class TestDemo01 {


    /**
     * @param args
     */
```

```
    public static void main(String[] args) {
        // TODO Auto-generated method stub
        // 产生接口的对象
        Fruit f = new Orange() ;
        f.eat() ;
    }

}
```

• 方法二：使用工厂设计模式，引入工厂类，可以解决客户端与具体子类之间的耦合，确实解决了耦合问题，但是新的问题造成了？

```
package cn.mldn.lxh.demo01;


public class TestDemo01 {


    /**
     * @param args
     */
    public static void main(String[] args) {
        // TODO Auto-generated method stub
        // 产生接口的对象
        Fruit f = FruitFactory.getFruitInstance(2) ;
        f.eat() ;
```

```
    }


}
```

在整个程序中，只要有一个接口就需要一个工厂，如果接口过多，则会造成工厂过多？

- 方法三：使用反射机制：Class.forName（JAVA 视频）
  联合 IO 操作、联合工厂设计、联合 Properties 类（代码量非常大，而且不易管理）

- 方法四：使用对象克隆：Object 类中的 clone 方法（需要一个具体类的对象）

Spring 提供的是一个容器，是一个开发框架

可以从www.springframework.org下载最新的开发包

Spring 开发包分为两种：

核心 Spring 包：包含了 Spring 核心的开发文件

依赖 Spring 包：包含了大量的其他开源组件（文件比较大）

将全部的开发包拷贝到 WebRoot/web-inf/lib 中

注意：Spring 同 Hibernate 一样，需要一个统一的配置文件管理（*.xml）

建立好了一个 applicationContext.xml 文件，此文件为 Spring 的核心配置文件，使用此文件可以统一进行 bean 的管理

```
<beans>

   <bean id="fru" class="cn.mldn.lxh.demo01.Apple"></bean>

</beans>
```

需要找到设置的 id，回想一下：Apple 是不是 Fruit 的子类

所以一切的对象都可以用 Fruit 接口接收

问题：如何在 Spring 中通过一定的方法得到此 Fruit 对象的实例？

使用 ApplicationContext 接口，ClassPathXmlApplicationContext 子类完成查找操作

结论：

使用 Spring 之后，工厂被取消了，而由 Spring 去替代工厂了，证明了 Spring 中主要有两个操作：一个是替代工厂，第二个，所有的工厂在资源文件（*.xml）中进行管理。

客户端代码之中，所有的代码的设计重要性变为接口 → Spring 中主要就是接口的设计

典型的范例：

JSP 中 DAO 设计模式 → VO（POJO）、工厂、数据源、DAO 接口、具体实现类

Spring 中的 Ioc（控制反转、依赖注入）

后面：

Spring 的 Ioc 设计（BEAN 的管理）
Spring 的 AOP 开发
Spring 的 JdbcTemplate 的使用
Spring 联合 Hibernate 开发
SSH 技术：Struts+Spring+Hibernate