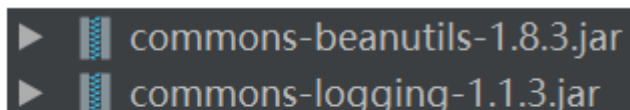


01_JSP模式

- Sun公司在推出jsp后，也为我们提供了两种jsp开发模式
- Model1:jsp+javaBean
 - jsp:显示数据+业务处理
 - javaBean:数据封装
 - 优缺点
 - 使用简单快捷，对开发人员要求不高。
 - 它不适合完成比较复杂的项目，而且所有操作都在jsp页面上，为我们的程序开发与维护带来不便：
- Model2:jsp+javaBean+Servlet
 - jsp:显示数据
 - Servlet:业务处理
 - javaBean:数据封装
 - 优缺点
 - 维护方便，开发人员各司其职，有利于我们进行分工操作，它比较适合开发一些比较复杂项目，因为它的很多组件可以重用。
 - 开发难度增大，对开发人员要求比较高。
- 案例演示
 - Model1模式
 - Model2模式

02_BeanUtils工具类使用及原理

- 导入jar包



```
▶ commons-beanutils-1.8.3.jar
▶ commons-logging-1.1.3.jar
```

-
- 使用
 - populate方法
- 原理代码
 - 注意,页面上的参数可能和javaBean上的参数不一致!所以,需要拿map中的参数和t中的参数做匹配

```
public static void populate(Object obj , Map<String,String[]> map){
    Class<?> clazz = obj.getClass();
    Field[] fields = clazz.getDeclaredFields();
    Set<String> fieldNames = map.keySet();
    for (String fieldName : fieldNames) {
        for (Field field : fields) {
            if (fieldName.equals(field.getName())) {
```

```

        String methodName =
"set"+fieldName.substring(0,1).toUpperCase()+fieldName.substring(1);
        try {
            Method method = clazz.getMethod(methodName, field.getType());
            if (field.getType().getName().equals("java.lang.Integer")) {
                method.invoke(obj,Integer.parseInt(map.get(fieldName)[0]));
            } else {
                method.invoke(obj,map.get(fieldName)[0]);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
}
}
}

```

03_MVC设计模式

- MVC全名是model view controller
- 它是一种软件设计典范，用一种业务逻辑，数据，界面分离的方式来组织代码，将业务逻辑聚集到一个部件中，方便程序的重复使用，提高我们的开发效率。
- jsp model2就是一个mvc设计模式的一种。
- 业务分工如下：
 - M:model 即模型层，维护数据提供数据的访问 javaBean
 - V:view 即视图层 数据的显示 jsp
 - C:controller 控制层，用于业务处理，而业务处理包含：处理请求、业务逻辑、操作数据库
- Web中的三层架构
 - web层：处理请求
 - service层:业务逻辑，业务层
 - dao层：操作数据库，持久层
 - 使用三层架构完成登录案例。