# 什么是mvc?

mvc是一种软件设计典范:它按数据、界面、业务逻辑三块将代码分离。

model：模型，作用是封装数据，再具体点就是我们建立的pojo，例如Student类。

view:视图，作用是数据展示的场所，用户交互的媒介。

controle：控制层，业务逻辑处理。

好处：

* 耦合度低：例如为了页面友好，可能需要美化view层，model与controle就无需变动，业务逻辑流程发生了变动，model、view层的代码无需变动。
* 代码重用性高

比如：同样的构件能被不同的界面使用。

* 开发效率高：开发时间得到相当大的缩减，程序员集中精力于业务逻辑，美工程序员（HTML和JSP开发人员）集中精力于表现形式上。
* **有利软件工程化管理：由于**每一层不同的应用具有某些相同的特征，有利于通过工程化、工具化管理程序代码。

# 为什么使用mvc?

提高开发效率，或者人类天生惰性使然:

在mvc出现之前两种处理方式：

第一种方式纯jsp开发，即在jsp页面写java代码，jsp页面可能非常大，可读性差，某业务逻辑功能很难定位，修改一处无法预知是否影响到其他功能，更无从谈代码公用。

第二种方式jsp+servlet:把之前在jsp页面进行的业务处理代码移植到servlet，但是代价是在web.xml中配置非常多的servlet。

Mvc出现之后：

Struts1、struts2、springmvc都是优秀的代表：

1. 数据处理更简单:view🡪controle时框架帮我们组织成了model实例，我们无需操心转型等（struts2利用拦截器、spring利用？）。

Controle🡪view时：我们可以使用简单的el表达式、框架标签（struts2、springmvc都提供有自己的标签）来渲染。

而我们之前必须使用request.getParameter(xxx)来获取…

1. 数据校验

Strus2利用拦截器实现，springmvc利用？

1. 国际化
2. …