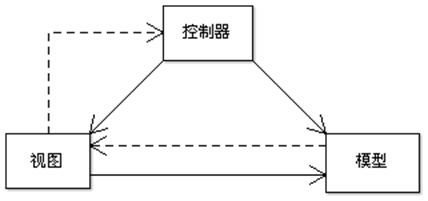
MVC架构模式

一、什么MVC

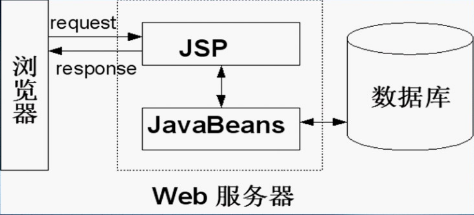
MVC全名是Model View Controller，是模型(model)－视图(view)－控制器(controller)的缩写，一种软件设计典范，用一种业务逻辑、数据、界面显示分离的方法组织代码，将业务逻辑聚集到一个部件里面，在改进和个性化定制界面及用户交互的同时，不需要重新编写业务逻辑。MVC被独特的发展起来用于映射传统的输入、处理和输出功能在一个逻辑的图形化用户界面的结构中。MVC模式最早由Trygve Reenskaug在1978年提出[1]，是施乐帕罗奥多研究中心（Xerox PARC）在20世纪80年代为程序语言Smalltalk发明的一种软件设计模式。MVC模式的目的是实现一种动态的程序设计，使后续对程序的修改和扩展简化，并且使程序某一部分的重复利用成为可能。除此之外，此模式通过对复杂度的简化，使程序结构更加直观。软件系统通过对自身基本部分分离的同时也赋予了各个基本部分应有的功能。



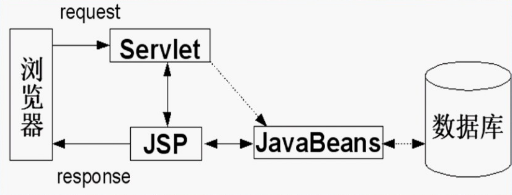
二、WEB MVC

随着Web的兴起，人们开始把MVC等知识应用到Web环境下，Java是Web MVC最早的实践者，开发出Model1和Model2，尤其Model2中使用JavaBean，JSP，Servlet分别对应MVC中的三个组成部分，形成一个典型的MVC模式。

**Model1:Jsp+JavaBeans**



**Model2:Jsp+JavaBeans+Servlet**



三、MVC编程模式

MVC 是一种使用 MVC（Model View Controller 模型-视图-控制器）设计创建 Web 应用程序的模式：

Model（模型）表示应用程序核心（比如数据库记录列表）。

View（视图）显示数据（数据库记录）。

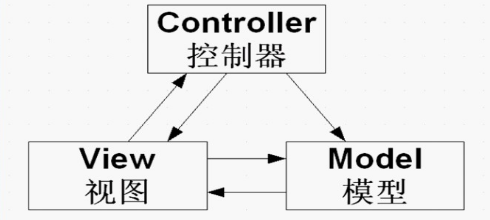
Controller（控制器）处理输入（写入数据库记录）。

MVC 模式同时提供了对 HTML、CSS 和 JavaScript 的完全控制。

**Model（模型）**是应用程序中用于处理应用程序数据逻辑的部分，通常模型对象负责在数据库中存取数据。

**View（视图）**是应用程序中处理数据显示的部分，通常视图是依据模型数据创建的。

**Controller（控制器）**是应用程序中处理用户交互的部分，通常控制器负责从视图读取数据，控制用户输入，并向模型发送数据。



三、MVC的优缺点

**MVC的优点**

1.【低耦合性】视图层和业务层分离，这样就允许更改视图层代码而不用重新编译模型和控制器代码，同样，一个应用的业务流程或者业务规则的改变只需要改动MVC的模型层即可。因为模型与控制器和视图相分离，所以很容易改变应用程序的数据层和业务规则。

2.【高重用性和可适用性】随着技术的不断进步，现在需要用越来越多的方式来访问应用程序。MVC模式允许你使用各种不同样式的视图来访问同一个服务器端的代码。它包括任何WEB（HTTP）浏览器或者无线浏览器（wap），比如，用户可以通过电脑也可通过手机来订购某样产品，虽然订购的方式不一样，但处理订购产品的方式是一样的。由于模型返回的数据没有进行格式化，所以同样的构件能被不同的界面使用。例如，很多数据可能用HTML来表示，但是也有可能用WAP来表示，而这些表示所需要的命令是改变视图层的实现方式，而控制层和模型层无需做任何改变。

3.【较低的生命周期成本】MVC使开发和维护用户接口的技术含量降低。

4.【快速的部署】使用MVC模式使开发时间得到相当大缩减，它使程序员（Java开发人员）集中精力于业务逻辑，界面程序员（HTML和JSP开发人员）集中精力于表现形式上。

5.【可维护性】分离视图层和业务逻辑层也使得WEB应用更易于维护和修改。

6.【有利于软件工程化管理】由于不同的层各司其职，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。

**MVC的缺点**

1.【**没有明确的定义】**完全理解MVC并不是很容易。使用MVC需要精心的计划，由于它的内部原理比较复杂，所以需要花费一些时间去思考。同时由于模型和视图要严格的分离，这样也给调试应用程序带来了一定的困难。每个构件在使用之前都需要经过彻底的测试。

2.【**不适合小型，中等规模的应用程序**】花费大量时间将MVC应用到规模并不是很大的应用程序通常会得不偿失。

3.【**增加系统结构和实现的复杂性】**对于简单的界面，严格遵循MVC，使模型、视图与控制器分离，会增加结构的复杂性，并可能产生过多的更新操作，降低运行效率。

4.【**视图与控制器间的过于紧密的连接】**视图与控制器是相互分离，但却是联系紧密的部件，视图没有控制器的存在，其应用是很有限，这样就妨碍了他们的独立重用。

5.【**视图对模型数据的低效率访问**】依据模型操作接口的不同，视图可能需要多次调用才能获得足够的显示数据。对未变化数据的不必要的频繁访问，也将损害操作性能。

6.【**一般高级的界面工具或构造器不支持模式**】改造这些工具以适应MVC需要和建立分离的部件的代价是很高的，会造成MVC使用的困难

四、基于MVC的Web框架





