## 框架内容

MVC是一个框架模式，它强制性的使应用程序的输入、处理和[输出](http://baike.baidu.com/view/700402.htm)分开。使用MVC应用程序被分成三个核心部件：模型、[视图](http://baike.baidu.com/view/71981.htm)、控制器。它们各自处理自己的任务。最典型的MVC就是JSP + [servlet](http://baike.baidu.com/view/25169.htm) + [javabean](http://baike.baidu.com/view/28155.htm" \t "_blank)的模式。

**视图**

视图是用户看到并与之交互的界面。对老式的Web应用程序来说，视图就是由[HTML](http://baike.baidu.com/view/692.htm" \t "_blank)元素组成的界面，在新式的Web应用程序中，[HTM](http://baike.baidu.com/view/177797.htm" \t "_blank)L依旧在视图中扮演着重要的角色，但一些新的技术已层出不穷，它们包括[Adobe Flash](http://baike.baidu.com/view/1128244.htm" \t "_blank)和像[XHTML](http://baike.baidu.com/view/15906.htm" \t "_blank)，[XML](http://baike.baidu.com/view/63.htm" \t "_blank)/[XSL](http://baike.baidu.com/view/45861.htm),[WML](http://baike.baidu.com/view/160091.htm)等一些标识语言和[Web services](http://baike.baidu.com/view/1086510.htm" \t "_blank).

MVC好处是它能为应用程序处理很多不同的[视图](http://baike.baidu.com/view/71981.htm" \t "_blank)。在视图中其实没有真正的处理发生，不管这些数据是联机存储的还是一个雇员列表，作为视图来讲，它只是作为一种输出数据并允许用户操纵的方式。

**模型**

模型表示企业数据和业务规则。在MVC的三个部件中，模型拥有最多的处理任务。例如它可能用像[EJB](http://baike.baidu.com/view/3542.htm" \t "_blank)s和ColdFusion Components这样的构件对象来处理数据库，被模型返回的数据是中立的，就是说模型与数据格式无关，这样一个模型能为多个视图提供数据，由于应用于模型的代码只需写一次就可以被多个视图重用，所以减少了代码的重复性。

**控制器**

控制器接受用户的输入并调用模型和视图去完成用户的需求，所以当单击Web页面中的超链接和发送[HTML表单](http://baike.baidu.com/view/3488030.htm" \t "_blank)时，控制器本身不输出任何东西和做任何处理。它只是接收请求并决定调用哪个模型构件去处理请求，然后再确定用哪个视图来显示返回的数据

## 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。**

MVC 分层有助于管理复杂的应用程序，因为您可以在一个时间内专门关注一个方面。例如，您可以在不依赖业务逻辑的情况下专注于视图设计。同时也让应用程序的测试更加容易。

MVC 分层同时也简化了分组开发。不同的开发人员可同时开发视图、控制器逻辑和业务逻辑。

**框架和设计模式的区别**

有很多程序员往往把框架模式和设计模式混淆，认为MVC是一种设计模式。实际上它们完全是不同的概念。

框架、设计模式这两个概念总容易被混淆，其实它们之间还是有区别的。**框架通常是代码重用，而设计模式是设计重用，架构则介于两者之间**，部分代码重用，部分设计重用，有时分析也可重用。在软件生产中有三种级别的重用：**内部重用**，即在同一应用中能公共使用的抽象块;**代码重用**，即将通用模块组合成库或工具集，以便在多个应用和领域都能使用；应用**框架的重用**，即为专用领域提供通用的或现成的基础结构，以获得最高级别的重用性。

框架与设计模式虽然相似，但却有着根本的不同。设计模式是对在某种环境中反复出现的问题以及解决该问题的方案的描述，它比框架更抽象；框架可以用代码表示，也能直接执行或复用，而对模式而言只有实例才能用代码表示;**设计模式是比框架更小的元素，一个框架中往往含有一个或多个设计模式，框架总是针对某一特定应用领域，但同一模式却可适用于各种应用。**可以说，框架是软件，而设计模式是软件的知识。

框架模式有哪些？

MVC、MTV、[MVP](http://baike.baidu.com/subview/7294/10754979.htm" \t "_blank)、CBD、[ORM](http://baike.baidu.com/view/197951.htm" \t "_blank)等等；

框架有哪些？

[C++](http://baike.baidu.com/view/824.htm)语言的QT、MFC、gtk，Java语言的[SSH](http://baike.baidu.com/view/16184.htm" \t "_blank) 、**[SSI](http://baike.baidu.com/view/188371.htm" \t "_blank)**，[php](http://baike.baidu.com/view/99.htm" \t "_blank)语言的 smarty(MVC模式)，[python](http://baike.baidu.com/view/21087.htm" \t "_blank)语言的django(MTV模式)等等

设计模式有哪些？

[工厂模式](http://baike.baidu.com/view/1306799.htm)、适配器模式、策略模式等等

简而言之：框架是大智慧，用来对软件设计进行分工；设计模式是小技巧，对具体问题提出解决方案，以提高代码复用率，降低耦合度。

## 常见框架

### Struts

[Struts](http://baike.baidu.com/view/25603.htm)是[Apache](http://baike.baidu.com/view/28283.htm" \t "_blank)软件基金下Jakarta项目的一部分。Struts框架的主要架构设计和开发者是Craig R.McClanahan。Struts 是Java Web MVC框架中不争的王者。经过长达九年的发展，Struts已经逐渐成长为一个稳定、成熟的框架，并且占有了MVC框架中最大的市场份额。但是Struts某些技术特性上已经落后于新兴的MVC框架。面对[Spring MVC](http://baike.baidu.com/view/1834982.htm" \t "_blank)、Webwork2这些设计更精密，扩展性更强的框架，Struts受到了前所未有的挑战。但站在产品开发的角度而言，Struts仍然是最稳妥的选择。

Struts有一组相互协作的类（组件）、[Servlet](http://baike.baidu.com/view/25169.htm" \t "_blank)以及jsp tag lib组成。基于struts构架的web应用程序基本上符合JSP Model2的设计标准，可以说是MVC设计模式的一种变化类型。根据上面对framework的描述，很容易理解为什么说Struts是一个web framework，而不仅仅是一些标记库的组合。但 Struts 也包含了丰富的标记库和独立于该框架工作的实用程序类。Struts有其自己的控制器（[Controller](http://baike.baidu.com/view/1805013.htm" \t "_blank)），同时整合了其他的一些技术去实现模型层（[Model](http://baike.baidu.com/view/2159664.htm)）和视图层（View）。**在模型层，Struts可以很容易的与数据访问技术相结合**，包括[EJB](http://baike.baidu.com/view/3542.htm)，[JDBC](http://baike.baidu.com/view/25611.htm" \t "_blank)和Object Relation Bridge。**在视图层，Struts能够与JSP, Velocity Templates,XSL等等这些表示层组件相结合。**

### Spring

[Spring](http://baike.baidu.com/view/23023.htm)实际上是Expert One-on-One J2EE Design and Development 一书中所阐述的设计思想的具体实现。在One-on-One 一书中，[Rod Johnson](http://baike.baidu.com/view/2192255.htm)倡导[J2EE](http://baike.baidu.com/view/1507.htm)实用主义的[设计思想](http://baike.baidu.com/view/14903484.htm)，并随书提供了一个初步的开发框架实现（interface21 开发包）。而Spring 正是这一思想的更全面和具体的体现。Rod Johnson 在interface21 开发包的基础之上，进行了进一步的改造和扩充，使其发展为一个更加开放、清晰、全面、高效的开发框架。

**Spring是一个开源框架，由Rod Johnson创建并且在他的著作《J2EE设计开发编程指南》里进行了描述。它是为了解决企业应用开发的复杂性而创建的。Spring使使用基本的JavaBeans来完成以前只可能由EJB完成的事情变得可能了。然而，Spring的用途不仅限于服务器端的开发。从简单性、可测试性和松耦合的角度而言，任何Java应用都可以从Spring中受益。**

简单来说，Spring是一个轻量的控制反转和面向切面的容

框架。当然，这个描述有点过于简单。但它的确概括出了Spring是做什么的。

## JavaBeans

**JavaBeans**是[Java](https://zh.wikipedia.org/wiki/Java)中一种特殊的[类](https://zh.wikipedia.org/wiki/%E7%B1%BB_(%E8%AE%A1%E7%AE%97%E6%9C%BA%E7%A7%91%E5%AD%A6))，可以将多个[对象](https://zh.wikipedia.org/wiki/%E5%AF%B9%E8%B1%A1_(%E8%AE%A1%E7%AE%97%E6%9C%BA%E7%A7%91%E5%AD%A6))封装到一个对象（bean）中。特点是可[序列化](https://zh.wikipedia.org/wiki/%E5%BA%8F%E5%88%97%E5%8C%96)，提供[无参构造器](https://zh.wikipedia.org/w/index.php?title=%E6%97%A0%E5%8F%82%E6%9E%84%E9%80%A0%E5%99%A8&action=edit&redlink=1)，提供[getter方法和setter方法](https://zh.wikipedia.org/w/index.php?title=%E8%B5%8B%E5%80%BC%E6%96%B9%E6%B3%95&action=edit&redlink=1)访问对象的属性。名称中的“Bean”是用于Java的可重用软件组件的惯用叫法。

### 优点

* Bean可以控制它的属性、事件和方法是否暴露给其他程序。
* Bean可以接收来自其他对象的事件，也可以产生事件给其他对象。
* 有软件可用来配置Bean。
* Bean的属性可以被序列化，以供日后重用。

### JavaBeans规范

要成为JavaBean[类](https://zh.wikipedia.org/wiki/%E7%B1%BB_(%E8%AE%A1%E7%AE%97%E6%9C%BA%E7%A7%91%E5%AD%A6))，则必须遵循关于命名、构造器、方法的特定规范。有了这些规范，才能有可以使用、复用、替代和连接JavaBeans的工具。

规范如下：

* 有一个公有的无参构造器。
* 属性可以通过*get*、*set*、*is*（可以替代get，用在布尔型属性上）方法或遵循特定命名规范的其他方法访问。
* 可序列化。