J2EE体系架构设计介绍1J2EE模型及J2EE设计模式Java认证考试 PDF转换可能丢失图片或格式，建议阅读原文 https://www.100test.com/kao_ti2020/644/2021_2022_J2EE_E4_BD_93_E7_B3_BB_c104_644482.htm 目前大多数企业采用J2EE技术的结构设计与解决方案。对于我们学习和研究J2EE体系结构来说，了解与掌握J2EE体系结构的设计方法及一些常用模式是必须的.模型-视图-控制(model-view-control，简称MVC)结构是目前最常见的J2EE应用所基于的体系结构，MVC主要适用于交互式的Web应用，尤其是存在大量页面及多次客户访问及数据显示.相比较而言，一个工作流体系结构更多应用于过程控制和较少交互的情况下.除了体系结构外，J2EE的设计模式对我们解决应用系统的设计也有很大的帮助。 一、J2EE的模型-视图-控制(MVC)体系结构 模型-视图-控制结构是交互式应用程序广泛使用的一种体系结构。它有效地在存储和展示数据的对象中区分功能模块以降低它们之间的连接度，这种体系结构将传统的输入、处理和输入模型转化为图形显示的用户交互模型，或者换一种说法，是多层次的Web商业应用.MVC体系结构具有三个层面：模型(Model)、视图(View)和控制(Controller)，每个层面有其各自的功能作用，MVC体系结构如下： 模型层负责表达和访问商业数据，执行商业逻辑和操作。也就是说，这一层就是现实生活中功能的软件模拟.在模型层变化的时候，它将通知视图层并提供后者访问自身状态的能力，同时控制层也可以访问其功能函数以完成相关的任务。 视图层负责显示模型层的内容。它从模型层取得数据并指定这些数据如何被显示出来。在模型层变化的时候，它将自动更新。另外视图层也会将用户的输入传

送给控制器。 编辑特别推荐: 指点一下：到底该不该去考JAVA认证? Java认证权威问答精华集 控制层负责定义应用程序的行为。它可以分派用户的请求并选择恰当的视图以用于显示，同时它也可以解释用户的输入并将它们映射为模型层可执行的操作.在一个图形界面中，常见的用户输入包括点击按钮和菜单选择。在Web应用中，它包括对Web层的HTTP GET和POST的请求.控制层可以基于用户的交互和模型层的操作结果来选择下一个可以显示的视图，一个应用程序通常会基于一组相关功能设定一个控制层的模块，甚至一些应用程序会根据不同的用户类型具有不同的控制层设定，这主要是由于不同用户的视图交互和选择也是不同的。 在模型层、视图层和控制层之间划分责任可以减少代码的重复度，并使应用程序维护起来更简单。同时由于数据和商务逻辑的分开，在新的数据源加入和数据显示变化的时候，数据处理也会变得更简单。 二、J2EE设计模式 一个设计模式描述了对于特定设计问题被验证的解决方案，它综合了所有开发者对这个问题所在领域的知识和见解.同时也是对于常见问题的可重用方案，它们一般适用于单个问题，但是组织在一起就可以提供整个企业系统的解决方案。下面我们列举八种常用于J2EE平台的设计模式，并对每种模式作简单的介绍，便于大家学习、理解与灵活应用。 1、前控制器 前控制器(front controller)主要提供一种可以集中式管理请求的控制器，一个前控制器可以接受所有的客户请求，将每个请求递交给相应的请求句柄，并适当地响应用户。 前控制器也是表示层的设计模式，它的出现主要是由于表示层通常需要控制和协调来自不同用户的多个请求，而这种控制机制又根据不同的需要，可能会集

中式控制或分散式控制。换句话说，就是应用系统需要对于表示层的请求提供一个集中式控制模块，以提供各种系统服务，包括内容提取、视图管理和浏览，如果系统中没有这种集中式控制模块或控制机制，每个不同的系统服务都需要进行单独的视图处理，这样代码的重复性就会提高，致使系统开发代价提高.同时，如果没有一个固定模块管理视图之间的浏览机制，致使其浏览功能下放于每个不同的视图中，最终必将使得系统的可维护性受到破坏.本文中我们主要讨论的是集中式控制模块，而不是分散式控制，因为前者更适合于大型的应用系统。 基于上面所说的问题，研究人员提出了前控制器的设计模式。在这种模式中，控制器提供一个处理不同请求的控制点，这里的处理工作包括安全事务、视图选择、错误处理和响应内容的生成.通过将这些处理工作集中在一点进行，大大地减低了Java代码量，同时这种方法也可以减少在视图模块的程序逻辑，保证了在不同请求之间可以重用大量的逻辑代码。通常，控制器都是和一个分派组件联合工作的，分派组件主要是用于视图管理和浏览，也就是为用户选择下一个应该显示的视图，并同时提供对于相关显示资源的控制。分派组件可以包含在控制器之内，或是在另外一个单独的组件中.虽然前控制器模式推荐对于全部的请求使用统一处理，但是它也没有限制在一个系统中只能具有一个控制器，在系统中的每个层次都可以具有多个控制器，并且映射至不同的系统服务。 下面我们来讨论各个组件。 2、控制器 控制器(controller)是负责处理各种客户请求的控制点，并可以将一定的职能(如用户认证等)下放给帮助类。 (1)分派组件(Dispatcher)。一个分派组件主要是用于视图的管理和浏览

，为用户选择下一个可以显示的视图，并管理相关的显示资源. 分派组件可以在一个控制器内运行，或者作为一个单独的组件与控制器协同工作.开发人员可以在分派组件中实现静态的视图分派技术，或是复杂的动态分派。 (2)帮助类(Helper)。帮助类负责帮助一个视图或控制器来完成其处理工作，因此，帮助类具有多项职责，包括收集数据、存储中间数据模型等.另外，帮助类也可以在保证数据完整性和准确性的情况下，为不同显示需求修改数据模型.也就是说，根据用户的请求，帮助类可以向视图提供未经处理的原始数据，或是已经格式化后的Web内容，一个视图同时可以和多个帮助类协同工作，而后者通常是由JavaBeans和标签(tag)实现的。 3、视图视图(view)负责向用户显示信息，而帮助类则负责支持视图的工作，即打包和建立相应的数据模型，下面我们介绍几种可以实现控制器的方法。 1)基于Servlet前控制器 这种方法建议使用servlet来实现一个控制器，尽管在语法上相差无几，但是它比使用JSP来实现要优越一些.因为控制器所进行的请求处理，多数都是与程序运行和控制流动相关的，这些处理工作虽然与显示模式相关，但是实际上是逻辑独立的，所以它们更适合在servlet中实现，而不是JSP技术中.使用这种方法也存在一些弱点，比如说servlet无法使用JSP运行环境的资源，如请求参数等，但是这个弱点也不是不能解决的，我们可以在servlet中建立相关的句柄来访问同样的资源，当然其代码会变得繁琐一点。 100Test 下载频道开通，各类考试题目直接下载。详细请访问 www.100test.com