REST及RESTful的实现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_REST_E5_8F_8ARES_c104_644567.htm 什么是REST？ REST (REpresentation State Transfer) 描述了一个架构样式的网络系统，比如 web 应用程序。它首次出现在 2000 年 Roy Fielding 的博士论文中，他是 HTTP 规范的主要编写者之一。REST 指的是一组架构约束条件和原则。满足这些约束条件和原则的应用程序或设计就是 RESTful。 Web 应用程序最重要的 REST 原则是，客户端和服务器之间的交互在请求之间是无状态的。从客户端到服务器的每个请求都必须包含理解请求所必需的信息。如果服务器在请求之间的任何时间点重启，客户端不会得到通知。此外，无状态请求可以由任何可用服务器回答，这十分适合云计算之类的环境。客户端可以缓存数据以改进性能。 在服务器端，应用程序状态和功能可以分为各种资源。资源是一个有趣的概念实体，它向客户端公开。资源的例子有：应用程序对象、数据库记录、算法等等。每个资源都使用 URI (Universal Resource Identifier) 得到一个惟一的地址。所有资源都共享统一的界面，以便在客户端和服务器之间传输状态。使用的是标准的 HTTP 方法，比如 GET、PUT、POST 和 DELETE。Hypermedia 是应用程序状态的引擎，资源表示通过超链接互联。 另一个重要的 REST 原则是分层系统，这表示组件无法了解它与之交互的中间层以外的组件。通过将系统知识限制在单个层，可以限制整个系统的复杂性，促进了底层的独立性。 当 REST 架构的约束条件作为一个整体应用时，将生成一个可以扩展到大量客户端的应用程序。它还降

低了客户端和服务器之间的交互延迟。统一界面简化了整个系统架构，改进了子系统之间交互的可见性。REST 简化了客户端和服务器的实现。 了解了什么是什么是REST，我们再看看RESTful的实现。最近，使用 RPC 样式架构构建的基于 SOAP 的 Web 服务成为实现 SOA 最常用的方法。RPC 样式的 Web 服务客户端将一个装满数据的信封（包括方法和参数信息）通过 HTTP 发送到服务器。服务器打开信封并使用传入参数执行指定的方法。方法的结果打包到一个信封并作为响应发回客户端。客户端收到响应并打开信封。每个对象都有自己独特的方法以及仅公开一个 URI 的 RPC 样式 Web 服务，URI 表示单个端点。它忽略 HTTP 的大部分特性且仅支持 POST 方法。 由于轻量级以及通过 HTTP 直接传输数据的特性，Web 服务的 RESTful 方法已经成为最常见的替代方法。可以使用各种语言（比如 Java 程序、Perl、Ruby、Python、PHP 和 Javascript[包括 Ajax]）实现客户端。RESTful Web 服务通常可以通过自动客户端或代表用户的应用程序访问。但是，这种服务的简便性让用户能够与之直接交互，使用它们的 Web 浏览器构建一个 GET URL 并读取返回的内容。 在 REST 样式的 Web 服务中，每个资源都有一个地址。资源本身都是方法调用的目标，方法列表对所有资源都是一样的。这些方法都是标准方法，包括 HTTP GET、POST、PUT、DELETE，还可能包括 HEADER 和 OPTIONS。 在 RPC 样式的架构中，关注点在于方法，而在 REST 样式的架构中，关注点在于资源 将使用标准方法检索并操作信息片段（使用表示的形式）。资源表示形式在表示形式中使用超链接互联。 Leonard Richardson 和 Sam Ruby 在他们的著作 RESTful Web

Services 中引入了术语 REST-RPC 混合架构。REST-RPC 混合 Web 服务不使用信封包装方法、参数和数据，而是直接通过 HTTP 传输数据，这与 REST 样式的 Web 服务是类似的。但是它不使用标准的 HTTP 方法操作资源。它在 HTTP 请求的 URI 部分存储方法信息。好几个知名的 Web 服务，比如 Yahoo 的 Flickr API 和 del.icio.us API 都使用这混合架构。 有两个 Java 框架可以帮助构建 RESTful Web 服务。erome Louvel 和 Dave Pawson 开发的 Restlet（见 参考资料）是轻量级的。它实现针对各种 RESTful 系统的资源、表示、连接器和媒体类型之类的概念，包括 Web 服务。在 Restlet 框架中，客户端和服务器都是组件。组件通过连接器互相通信。该框架最重要的类是抽象类 Uniform 及其具体的子类 Restlet，该类的子类是专用类，比如 Application、Filter、Finder、Router 和 Route。这些子类能够一起处理验证、过滤、安全、数据转换以及将传入请求路由到相应资源等操作。Resource 类生成客户端的表示形式。 JSR-311是 Sun Microsystems 的规范，可以为开发 RESTful Web 服务定义一组 Java API。Jersey是对 JSR-311 的参考实现。 JSR-311 提供一组注释，相关类和接口都可以用来将 Java 对象作为 Web 资源展示。该规范假定 HTTP 是底层网络协议。它使用注释提供 URI 和相应资源类之间的清晰映射，以及 HTTP 方法与 Java 对象方法之间的映射。API 支持广泛的 HTTP 实体内容类型，包括 HTML、XML、JSON、GIF、JPG 等。它还将提供所需的插件功能，以允许使用标准方法通过应用程序添加其他类型。 RESTful的实现：构建 RESTful Web 服务的多层架构 RESTful Web 服务和动态 Web 应用程序在许多方面都是类似的。有时它们提供相同或非常

类似的数据和函数，尽管客户端的种类不同。例如，在线电子商务分类网站为用户提供一个浏览器界面，用于搜索、查看和订购产品。如果还提供 Web 服务供公司、零售商甚至个人能够自动订购产品，它将非常有用。与大部分动态 Web 应用程序一样，Web 服务可以从多层架构的关注点分离中受益。业务逻辑和数据可以由自动客户端和 GUI 客户端共享。惟一的不同点在于客户端的本质和中间层的表示层。此外，从数据访问中分离业务逻辑可实现数据库独立性，并为各种类型的数据存储提供插件能力。 展示了自动化客户端，包括 Java 和各种语言编写的脚本，这些语言包括 Python、Perl、Ruby、PHP 或命令行工具，比如 curl。在浏览器中运行且作为 RESTful Web 服务消费者运行的 Ajax、Flash、JavaFX、GWT、博客和 wiki 都属于此列，因为它们都代表用户以自动化样式运行。自动化 Web 服务客户端在 Web 层向 Resource Request Handler 发送 HTTP 响应。客户端的无状态请求在头部包含方法信息，即 POST、GET、PUT 和 DELETE，这又将映射到 Resource Request Handler 中资源的相应操作。每个请求都包含所有必需的信息，包括 Resource Request Handler 用来处理请求的凭据。 从 Web 服务客户端收到请求之后，Resource Request Handler 从业务逻辑层请求服务。Resource Request Handler 确定所有概念性的实体，系统将这些实体作为资源公开，并为每个资源分配一个惟一的 URI。但是，概念性的实体在该层是不存在的。它们存在于业务逻辑层。可以使用 Jersey 或其他框架（比如 Restlet）实现 Resource Request Handler，它应该是轻量级的，将大量职责工作委托给业务层。 Ajax 和 RESTful Web 服务本质上是互为补充的。它们都可

以利用大量 Web 技术和标准，比如 HTML、JavaScript、浏览器对象、XML/JSON 和 HTTP。当然也不需要购买、安装或配置任何主要组件来支持 Ajax 前端和 RESTful Web 服务之间的交互。RESTful Web 服务为 Ajax 提供了非常简单的 API 来处理服务器上资源之间的交互。 Web 浏览器客户端作为 GUI 的前端，使用表示层中的 Browser Request Handler 生成的 HTML 提供显示功能。Browser Requester Handler 可以使用 MVC 模型（JSF、Struts 或 Spring 都是 Java 的例子）。它从浏览器接受请求，从业务逻辑层请求服务，生成表示并对浏览器做出响应。表示供用户在浏览器中显示使用。表示不仅包含内容，还包含显示的属性，比如 HTML 和 CSS。 业务规则可以集中到业务逻辑层，该层充当表示层和数据访问层之间的数据交换的中间层。数据以域对象或值对象的形式提供给表示层。从业务逻辑层中解耦 Browser Request Handler 和 Resource Request Handler 有助于促进代码重用，并能实现灵活和可扩展的架构。此外，由于将来可以使用新的 REST 和 MVC 框架，实现它们变得更加容易，无需重写业务逻辑层。数据访问层提供与数据存储层的交互，可以使用 DAO 设计模式或者对象-关系映射解决方案（如 Hibernate、OJB 或 iBATIS）实现。作为替代方案，业务层和数据访问层中的组件可以实现为 EJB 组件，并取得 EJB 容器的支持，该容器可以为组件生命周期提供便利，管理持久性、事务和资源配置。但是，这需要一个遵从 Java EE 的应用服务器（比如 JBoss），并且可能无法处理 Tomcat。该层的作用在于针对不同的数据存储技术，从业务逻辑中分离数据访问代码。数据访问层还可以作为连接其他系统的集成点，可以成为其他 Web 服务的客户

端。 数据存储层包括数据库系统、LDAP 服务器、文件系统和企业信息系统（包括遗留系统、事务处理系统和企业资源规划系统）。使用该架构，您可以开始看到 RESTful Web 服务的力量，它可以灵活地成为任何企业数据存储的统一 API，从而向以用户为中心的 Web 应用程序公开垂直数据，并自动化批量报告脚本。 什么是REST：结束语 REST 描述了一个架构样式的互联系统（如 Web 应用程序）。REST 约束条件作为一个整体应用时，将生成一个简单、可扩展、有效、安全、可靠的架构。由于它简便、轻量级以及通过 HTTP 直接传输数据的特性，RESTful Web 服务成为基于 SOAP 服务的一个最有前途的替代方案。用于 web 服务和动态 Web 应用程序的多层架构可以实现可重用性、简单性、可扩展性和组件可响应性的清晰分离。Ajax 和 RESTful Web 服务本质上是互为补充的。开发人员可以轻松使用 Ajax 和 RESTful Web 服务一起创建丰富的界面。 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