java认证:Eclipse新成员Swordfish详解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_ti2020/571/2021_2022_java_E8_AE_A4_E8_AF_81_c104_571338.htm 企业服务总线（ESB）一直以来都是每个运营商SOA策略的基石。然而一直以来，由于其过于庞大，架构集中化，而且与已有的应用程序难以结合的关系，ESB一直未能将SOA的价值充分的发挥出来。 正因如此，Swordfish选择了一条全新的道路。建立在Apach ServiceMix和Apache CXF等开源组件之上，Swordfish提供了一个可扩展框架。这个框架为应用程序开发者以及系统集成商们提供了自己建立ESB的可能性而且是量身订造。而且Swordfish不仅仅是框架：秉持着Eclipse “ 可扩展框架与exemplary插件相结合 ” 的传统，Swordfish项目也把目标放在了企业级插件上。这将打造一个对 “ E ” （企业）十分重视的，成熟的开源ESB。为了展示Swordfish相对于其他开源ESB的优势，我们以下会着重说明那些总结了在真实的企业环境下积累了数年SOA经验所带来的功能。 第一个功能可以在运行时（runtime）中将一个服务注册（Service Registry）与一个动态绑定服务（dynamically bind services）整合到一起。也就是说，与其他ESB所不同，Swordfish执行这个任务将无需一个静态的关联，以往这个静态关联是用来将使用服务的组件（客户）与提供服务的组件（供应方）联系起来的。Swordfish的做法是，一个客户找到一个供应方靠的是逻辑标识符，这个逻辑标识符又依照一个策略将此服务界面标识为重要。所谓策略就是有关用户非功能能力及需求的一个描述。有了这两条信息，加上包括了已有服务供应方的数据库以及他们各

自的策略，服务注册会选择一个合适的供应方并计算出一个有效策略，这个策略将掌管以后在客户和供应方之间的一切通信。这个方法的优势是明显的：客户与供应方被松散的组合在一起，无需改动商业应用程序代码就可以轻松改动双方的非功能参数。另一个Swordfish的显著特征是它的架构模式“分布的ESB”（Distributed ESB），或者叫做“联邦的ESB”（Federated ESB）。这个意思就是说，两个服务参与者之间的通信无需中心组件，只需建立通信后便可实现。相对于“传统的”中心辐射型集成架构，这种模式的优势在于它不会因中心组件过时而形成性能瓶颈，从而整个系统具有更强的伸缩性，用户在SOA化的过程中实现更多功能的同时无需花费大笔的精力财力在硬件设施上。另一方面，联邦ESB的潜在缺陷在于它令系统的管理更加复杂化。不过这个缺陷可以通过一个远程配置机制来弥补。在Swordfish中，管理员可以方便并高效的配置大数量分布的Swordfish个体，而且监控功能也很完善。监控功能在业务过程管理（BPM）中尤其的重要，因为细致监控事件是一个完整的业务活动监控（BAM）的前提，而BAM一般都基于复杂事件处理（CEP）。OSGi和JBI全面详解 Swordfish框架所使用的是现在SOA通用的标准。在底层有OSGi，具体来说就是Equinox，即Eclipse基金会的OSGi。OSGi提供了组件模型，一个模块部署系统，以及一个clean class加载系统。在上层有JBI标准来实现组件之间的消息抽取（messaging abstraction）以及消息路由（message routing）。一开始的JBI 1.0使用自己的组件模型和部署系统，不过后来转而使用了OSGi。这也符合JBI 2.0工作组的情况，还有Apache ServiceMix的第4版。Apache ServiceMix这个Apache的JBI容器

（JBI container）正是Swordfish中的核心消息路由引擎。而ServiceMix不仅仅提供路由功能，它本身就包含了大量做好了的组件，这些组件可以直接被用于连接业务逻辑（business logic）或用于传输通道以及协议。 综合来说，Swordfish是一个建立在ServiceMix基础上，并将功能延伸至企业SOA范围的框架。Swordfish核心使用 ServiceMix的公共扩展点（public extension points）在规范化消息路由器（NMR）中与消息流建立联系。 与NMR内部建立联系的中心概念就是拦截器（Interceptor）。拦截器是一段代码，这段代码在消息经过NMR进行消息交换时将其拦截下来并对这个消息做一番动作。Java界面可以在Swordfish API中起到拦截器的作用，借助一些插件便可让这个拦截器实现一些特定任务，如消息认证（validation），消息转换（transformation）等。拦截器对消息交换作用的次序可以通过Swordfish框架核心中一个叫做Planner的组件来计算并定义计划。具体计划如何实施可以由用户来决定，所以API中还有一个叫做PlannerStrategy的界面，此界面可以通过插件来使用。举例来说，评估一个消息交换中的策略便可以通过这个 PlannerStrategy来实现。除了传统拦截器以外，在特定情况下还可以使用特定的拦截器，比如有一种拦截器可以在服务注册中观察一个服务并为相应的交换重建路由。这个拦截器的基本性能（JBI相关）是核心运行中的一部分，不过具体的观察过程如何实施则靠一个将ServiceResolver界面实施到框架API的插件来完成。所有的公共API都是这样工作的。 面向应用程序开发者的注册和版本库 以上我们简单介绍了使用服务注册来解决运行时服务端点的优点。不过，服务注册还有更好的地方：它提供了一个面

向服务架构中全部服务的全面总览，而且对于服务再利用有很大的用处，而服务再利用则是SOA规范的主要优势之一。如果服务界面在定义的时候就仔细注意到再利用的方面，那么无论多么复杂的服务，在理论上也无非就是把一堆封闭的盒子排列好，再连接起来的问题而已。 以后，服务注册还会增添服务版本库（Service Repository）的功能，此功能可以将所有SOA相关的部件（比如服务界面介绍，策略等）作为SOA工作流中的一部分，安全并有序的保存起来。由此可以实现版本控制，定义认可工作流，分析和报告可行性等。这些功能令企业建立并推广管理流程（governance process）成为可能，而管理流程则是成功SOA的必要组成部分。 