

Oracle WebLogic Server: 面向服务的体系结构的坚实基础

Oracle 白皮书
2008 年 6 月更新

Oracle WebLogic Server: 面向服务的体系结构的坚实基础

引言	3
ORACLE 和 JAVA 平台, 企业版 5.....	4
用于 SOA 的更好的业务逻辑.....	5
EJB 3.0 消除了繁琐工作.....	6
批注的作用	6
简化的部署描述符.....	7
Oracle WebLogic Server 使用 Pitchfork 实施 EJB 3.0	7
简化 SOA 的持久性.....	8
同时支持 JPA 和 JDO	9
高性能实施	10
ORACLE TOPLINK	10
改善的 WEB 服务处理.....	11
改善的 WEB 界面.....	12
行业级 SOA 部署和管理.....	12
结论	14

Oracle WebLogic Server: 面向服务的体系结构的坚实基础

引言

面向服务的体系结构 (SOA) 已引发一场 IT 革命。将相关的软件功能包部署为松散耦合的粗粒度服务显著提高了应用程序的灵活性, 允许企业持续调整众多服务以保持 IT 功能与业务目标一致。

Oracle 在帮助企业使用 Java 实现 SOA 优势方面是领先者。凭借 Oracle WebLogic Server, Oracle 为基于 SOA 的 Java 平台, 企业版 5 (Java EE 5) 提供了坚实的基础。Oracle WebLogic Server 随取随用, 非常简便并提供了行业级的可靠性、可用性、可扩展性和性能。客户可以利用强大的配置、部署和管理工具迅速升级并管理现有服务。也可以利用与其他 Oracle 融合中间件产品的集成以及开发人员在开放源代码技术方面 (如 Spring Framework) 的经验。

本白皮书讨论了 Java EE 5 如何显著提高 SOA 应用程序的开发速度, 以及企业开发人员如何使用 Oracle WebLogic Server 以发挥它的强大功能。

Oracle WebLogic Server 的现有客户已证明了 Oracle 对关键任务 SOA 活动的支持价值。但是，最新版的 Oracle WebLogic Server 在此公认平台上会发挥更大的作用：客户可更快地构建服务、更轻松地将它们组合起来，并更有效地管理它们。对于想为开发人员提供行业级功能的组织而言，此解决方案作为最现代、最坚实和最强大的基于 Java 的 SOA 平台，是明智之选。

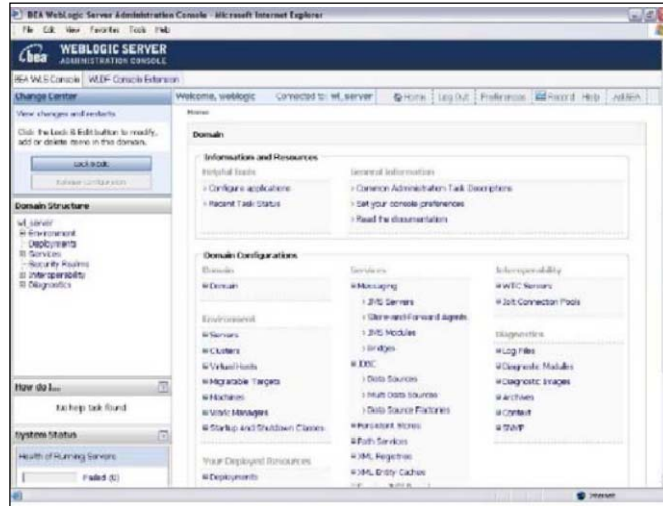


图 2: Oracle WebLogic Server

用于 SOA 的更好的业务逻辑

在 SOA 中，不同服务可在不同抽象级别上运转。对于企业而言，通常与业务域中的任务密切相关的顶层服务包括贷款处理域中的“检查信用评分”、移动通信域中的“备抵帐户”和应收账款域中的“提交发票”。显然，实施这些服务需要为对应域提供某种软件模型。

EJB 的最初目标是为构建支持复杂业务流程的富域模型提供基础架构。实体 bean 操作数据并实施域实体（例如，贷款申请、客户和采购订单）行为。会话 bean 组合协调实体交互以执行每个域的流程，例如“评估贷款”、“登记客户”和“向供应商付款”。

许多情况下，早期版本的 EJB 已被证实过于复杂而无法支持此方法。API 使用了太多必要机制来确保域流程的强健执行。建立有效的域模型通常需要许多步骤，要经过建立原型、测试和改进这样一个周期。在此周期中，开发人员不得不担忧底层“管道”，这使 EJB 对于许多项目来说过于繁琐。

EJB 3.0 将开发人员和管道完全隔离。在很大程度上，EJB 可被看作常规 Java 对象，从而显著简化了开发。利用该简化方法，开发人员可最终使用 EJB 来实施支持顶层企业服务的各种域模型。

与此相反，EJB 3.0 将开发人员与管道完全隔离。在很大程度上，EJB 可被看做常规 Java 对象，从而显著简化了开发。利用该简化方法，开发人员可最终使用 EJB 来实施支持顶层企业服务的各种域模型。

EJB 3.0 消除了繁琐工作

EJB 3.0 的变化主要影响开发人员与容器工具交互的方式。除了持久性，这些变化不会对工具产生显著影响。

在 EJB 的先前版本中，开发人员不得不执行繁琐的工作以和容器交互。最繁琐的工作是要实施所有用于创建主接口、本地接口以及远程接口的必要接口以及对应于 EJB 类型的接口。对于主接口和远程接口，开发人员需要处理所有的必然异常。实体 Bean 也需要 finder 方法。最后一步是为可适用的 EJB 接口实施所有的生命周期方法。

另一项繁琐的工作是编写 Java 命名和目录接口 (JNDI) 查找以获取资源参考。富域模型的元素之间天然存在许多关系。每种关系以及对基础架构资源的任何参考都需要查找代码。

EJB 3.0 消除了这两项繁琐的工作。开发人员将 EJB 编写为普通的旧式 Java 对象 (POJO)。它们通过向 POJO 代码添加简单的声明式批注来与容器工具进行交互。容器负责处理必要的繁琐工作，因而允许开发人员将重点放在构建域模型上。

批注的作用

如前所述，批注是提升 EJB 3.0 开发人员体验的关键创新。Oracle 帮助开发了批注并努力

拓展了它们在 Java EE 中的使用。如果开发人员可以明确指定要做什么，为什么不自动处理它呢？例如，假设在贷款处理领域工作的开发人员要为执行贷款处理的无状态会话 bean 编写一个客户端。与先前版本的 EJB 必须实施所有接口和 JNDI 查找代码不同，开发人员只需编写

```
import Loanprocessor.Loanprocessor

@stateless public class LoanProcessorClient {

@Inject LoanProcessor

...

}
```

Java EE 5 包括对应于其他类型 EJB 的批注。另外，开发人员可以创建与其他任何 POJO 一样的新实例，而非执行生命周期方法。对于实体 bean，甚至存在指定自动生成的标识符作为主键，然后使用该主键执行查找这种常见情况的批注。除了控制 EJB，Java EE 5 还包括各种批注，用于简化对安全性、持久性以及 Web 服务工具的访问。

@Stateless 批注代替了手动定义远程接口，而 @Inject 批注代替了手动 JNDI 查找。

Java EE 5 包括对应于其他类型 EJB 的批注。另外，开发人员可以创建与其他任何 POJO 一样的新实例，而非执行生命周期方法。对于实体 bean，甚至存在指定自动生成的标识符作为主键，然后使用该主键执行查找这种常见情况的批注。除了控制 EJB，Java EE 5 还包括各种批注，用于简化对安全性、持久性以及 Web 服务工具的访问。

简化的部署描述符

除了与 EJB 先前版本相关的繁琐编程工作，开发人员还不得不与复杂的部署描述符斗争。在处理繁琐工作并编写实际业务逻辑之后，要部署和运行 EJB 还需要使用 XML 编写部署描述符。对于会话 bean，描述符包含的多数是冗余信息，例如相关接口的类和名称以及 EJB 的类型。然后是针对基础架构服务（例如事务管理和安全性）的指令。对于具有容器管理持久性的实体 bean，部署描述符可能包含大量条目，这些条目指定其抽象数据模式和各种针对该模式的查询。

多数生产实施（例如 Oracle WebLogic Server）包括用于生成部署描述符并将其放在正确位置的工具。但是这些替代了开发人员本应关注的业务问题，成了另一件烦心事。在 EJB 3.0 中，部署描述符是可选的。开发人员无需描述符即可通过批注和一组为执行 EJB 提供足够信息的默认值来编写和执行任何类型的 EJB。要指定描述符的开发人员只需指定那些改写默认值的条目。在 EJB 3.0 中，使用一组协作 EJB 实施一个复杂服务需要少得多的文件，每个文件的条目也要少得多。

Oracle WebLogic Server 使用 Pitchfork 实施 EJB 3.0

如前所述，EJB 3.0 无法显著更改 EJB 容器所提供的工具，只能使这些工具更易于使用。Oracle WebLogic Server 将 EJB 3.0 作为其公认容器的扩展来实施，通过将批注转换为具体指令并解析以声明的方式指定的依赖关系来为运行更简化的 EJB 提供所需的只能。

要创建此扩展，Oracle 与开放源代码 Spring Framework 的开发人员协作以创建特殊版的 Spring — Pitchfork — 一个常用框架，该框架通过依赖性注入来帮助开发人员简化 Java 应用程序的开发。

将 Pitchfork 添加到 Oracle WebLogic Server 和 EJB 容器创建了同类最佳解决方案：一个公认的、行业级的容器和效率提升框架。

将 Pitchfork 添加到 Oracle WebLogicServer 和 EJB 容器中以创建同类最佳解决方案：一个公认的、行业级的容器和效率提升框架。

该方法的其他两个功能是向后兼容性和前瞻意识。由于 Pitchfork 充当 EJB 3.0 代码和传统容器工具的中介，因此 Oracle WebLogic Server 可完全向后兼容 EJB 2.1。EJB 简单地绕过 Pitchfork 直接使用容器工具。客户可并行运行 EJB 3.0 和 EJB 2.1。除了向后兼容性，Pitchfork 还简化了前瞻性意识。许多开发人员已采用先进的编程方法，例如使用 Spring 的面向方面的编程。Pitchfork 允许对 EJB 3.0 使用相同的方法。

简化 SOA 的持久性

EJB 3.0 帮助开发人员避开了复杂工作，从而允许他们专注于构建有代表性的域模型。持久性是下一个挑战。执行业务流程需要操作记录数据。前面例子中所提到的贷款申请、客户和采购订单实体都具有在 EJB 容器中执行的 Java 表示。但是，执行实际工作需要加载和保存存储在后端数据库中的相应表示，从而确保了每个细粒度数据单元有唯一一个“真正”副本，且操作同一数据单元的不同服务互不干扰。

理论上，诸如 SOA 等多层方法使应用层独立于数据层。实际上，大部分应用层代码执行数据管理任务。根本问题在于应用层如何增加价值。大多数服务在特定域内适当地提供了独特价值，它们在更广泛的流程中执行一组分配的任务。此特性要求它们以特定的方式来操作业务实体数据。它们希望以最便捷的形式安排所需数据，这不足为奇。

因此，每个服务必须把细粒度后端数据单元映射到其粗粒度实体模型。手动编写和调试映射代码非常费时且极易出错。自动映射方法也不是万能之计，因为它在灵活性和复杂性之间进行了折衷。不够灵活意味着开发人员必须手动编写大量代码来满足需求。由于过于复杂，开发人员感觉使用自动化工具就和编写大量代码一样。在大量 Oracle 实地经验基础上，Oracle WebLogic Server 合并了可与 JPA 和 Java 数据对象 (JDO) 无缝集成的 Oracle Kodo。通过 Oracle WebLogic Server，开发人员可为特定域模型选择最佳机制。

在大量 Oracle 实地经验基础上，Oracle WebLogic Server 合并了可与 JPA 和 Java 数据对象 (JDO) 无缝集成的 Oracle Kodo。通过 Oracle WebLogic Server，开发人员可为特定域模型选择最佳机制。

同时支持 JPA 和 JDO

如前所述，EJB 3.0 使用批注来减少接口实施编码和部署描述符中的条目。对于具有容器管理持久性的实体 bean 而言，后者的优势显而易见。抽象的编码加上冗长的部署描述符使 EJB 2.1 使用起来更加繁琐。此外，尽管 EJB 2.1 做出一些努力使其持久性工具独立于数据库技术，但它倾向于将使用关系数据库的典型用例变得很复杂。

JPA 解决了所有这些缺陷。开发人员只需提供代码批注便可指导容器如何访问适当的关系数据库数据。一组访问器之前的 @Id 批注定义了主键。甚至还有用于指定组合键的批注。集合定义之前的 @OneToMany 批注定义了一对多的关系。对于多对多关系，@JoinTable 批注提供了设置连接表的方法。更复杂的批注（如 @Transaction 和 @NamedQueries）包括更多的属性以精确控制与关系数据库的特定交互。

JPA 不仅简化了 EJB 2.1 的持久性功能。它还包括急需的增强。最重要的是，开发人员不再需要为实体 bean 指定生命周期方法。容器自动提供一个用于控制实例生命周期的 EntityManager 对象。JPA 还使 Java 类能够在映射到数据库时指定处理继承的策略。与 2.1 版的 EJB 查询语言相比，JPA 查询语言包括若干新特性，例如批量操作、外部连接以及子查询。同时，易用性和功能增强使开发人员能够为最常见的 SOA 数据访问需求快速实现持久性。

虽然 JPA 有了大幅改进，但许多开发人员仍然喜欢使用 JDO。因为 JPA 专注于关系数据库的持久性，它采用了关系样式。使用高级面向对象模型时，JDO 面向对象的语法更具吸引力。在许多情况下，选择只是一种个人偏好。凭借 Oracle WebLogic Server，开发人员可通过 JPA 使用容器管理持久性，或通过 JDO 使用 bean 管理持久性。虽然 JPA 和 JDO 共享很多底层功能，但是各方都有对方所没有的独特功能。Oracle WebLogic Server 提供了针对两个 API 的扩展，使它们具有相同的功能。开发人员选择其中之一方时，不必再牺牲另一方的功能。

JPA 所带来的易用性以及功能增强使开发人员能够为最常见的 SOA 数据访问需求快速实现持久性。

高性能实施

Oracle 对于通过 Java EE 5 构建高价值服务的承诺不仅仅是为开发人员提供持久性 API 选择。还包括无论开发人员选择哪种方式，均能提供高性能。

Oracle 对于通过 Java EE 5 构建高价值服务的承诺不仅仅为开发人员提供了持久性 API 选择。还包括无论开发人员选择哪种方式，均能提供高性能。如前所述，同一个持久性引擎可同时执行 JPA 和 JDO 功能。

Oracle Kodo 包括许多行业级的特性。为顶级业务服务提供持久性的最大挑战或许是支持大规模、长时间运行的事务处理。完成业务流程中的关键步骤可实现对许多不同数据的大量更新。Oracle Kodo 支持无限规模的事务处理。保证复杂业务流程间的协作可能需要事务处理在数分钟、数小时甚至数天内处于运行状态。Oracle Kodo 可在如此长时间运行的事务处理过程中有效地管理与数据源的连接。

ORACLE TOPLINK

Oracle WebLogic Server 还包含另一种高性能持久性技术 — Oracle TopLink。此解决方案是商业级版本和 TopLink Essentials 的扩展集。Oracle 作为新 EJB 3.0/JPA 的合作规范领导者，帮助设计了新 JPA 规范，并为其提供了基础架构指导。此外，Oracle 还为用于 JPA 参考实施的 TopLink Essentials 代码做出了贡献。TopLink Essentials 目前为开放源代码。

Oracle WebLogic Server 还包含另一种高性能持久性技术 — Oracle TopLink。此解决方案是商业级版本和 TopLink Essentials 的扩展集。

与 Oracle WebLogic Server 整合后，Oracle TopLink 包括了 TopLink Essentials 所不具有的高级对象关系映射 (ORM) 功能。这些功能包括支持集群应用程序部署的合作缓存以及更多非侵入式、优化的锁定策略。Oracle TopLink 提供了独立于平台的存储过程和功能支持并实现了历史映射和时间点查询。Java 管理扩展 (JMX) 管理 beans (MBeans) 允许管理和监控 Oracle TopLink 会话及其缓存。当在 Oracle 数据库环境中工作时，Oracle TopLink 提供以下特性：

- 一个虚拟专用数据库
- XML 类型映射和 SQLX 查询生成
- 提示
- 级联查询
- ORM、阵列、结构、对象引用和嵌套表
- 定制业务范围、时间标记和双字节数据类型

最后，Oracle TopLink 支持使用 Java 连接器体系结构资源适配器映射到执行信息系统。它在用于 XML 绑定的 Java 体系结构 (JAXB) 1.0 的基础上提供 XML 映射，并对 JAXB 2.0 功能提供早期支持。因为 Oracle 致力于可热插拔的体系结构，所以开发人员可选择使用适合其项目的持久性技术。

改善的 WEB 服务处理

EJB 3.0 对常规 Java 对象的使用使开发人员能够快速简便地构建富域模型，并将这些模型连接到后端数据库。当然，随后这些模型必须在企业 SOA 所支持的更大业务流程中用作服务。实现所有服务的无缝协作需要一个用于 Web 服务协议处理的坚实基础。

协议处理主要存在两个问题。首先，互操作性需要实施整个协议堆栈以及经验证的与不同供应商堆栈的兼容性。Oracle WebLogic Server 支持面向安全性和性能的最新 Web 服务 (WS) 协议。在安全方面，更新支持 WS-Security 1.1 与 WS-SecurityPolicy 1.1 和 1.2。为 WS-Trust 和 WS-SecureConversation 提供了新支持，使服务能够建立共享的安全环境并维持长期的信任关系。

在性能方面，为 XML-binary Optimized Packaging (XOP) 和消息传输优化机制 (MTOM) 规范提供了新支持。XOP 允许服务像现在一样传输二进制数据，而无需 base64 编码，并且将数据放在相同的 MIME 程序包中作为 XML 文件的其余部分。此方法降低了所需带宽及编码成本。MTOM 描述了如何使用 XOP 来优化简单对象访问协议消息的传输。Oracle 参与了多个互操作性事件来确保这些协议的实施以及 Oracle WebLogic Server 堆栈的其他协议与其他供应商协议间的良好协作。

第二个协议处理问题更加微妙。以前版本 Java EE 中的基于 XML 的远程过程调用的 Java API (JAX-RPC) 只支持 RPC 样式，遗憾的是，这种样式是所有 Web 服务交互样式中最不灵活的一种。Java EE 5 引入一个新的处理 API，用于 XML Web 服务的 Java API (JAX-WS)，它支持更加灵活的面向文档样式。Oracle WebLogic Server 还为第三种样式提供基本支持，这种样式是表现状态传输，可以简化某些交互。用于多种 Web 服务样式的处理基础架构允许开发人员在 SOA 内定制交互以满足不同企业的需求。

虽然这不是一个严格的协议处理问题，但对于只希望构建作为服务客户而没有面向服务器功能负担的轻型应用程序的客户来说，仅使用客户端的 Web 服务堆栈是一个实际问题。Oracle WebLogic Server 提供一个特殊的 Java 库，该库仅实现 Web 服务互操作性组织 Basic Profile 定义的协议中的客户端部分。

改善的 WEB 界面

几乎 SOA 支持的所有业务流程均需要某种形式的用户交互，而且大多数流程涉及大量的用户交互。顶层 SOA 服务趋向于提供一个业务任务的自然表示。利用此表示来提供更复杂的用户界面是一个巨大的机会。Java EE 5 为其 Web 界面 API 提供了一个集成增强包。这些增强支持更复杂的用户交互，使界面编程更容易，还扩大了人员与服务交互的广度和深度。

Java EE 5 Web 界面 API 的核心是 JavaServer Faces (JSF)。JSF 的主要目标是使 Web 界面开发与 GUI 开发更相似。框架基础模型使开发人员能够把界面功能与逻辑组件相连，从而在本质上创建抽象部件。用户和部件间的交互使用事件驱动模型。开发的最后一步是把抽象的交互模型与特定技术（如 Web）绑定。

此交互样式使在框架内控制界面状态更加容易。一个直接好处就是开发人员不需要在其 JavaServer 页面 (JSP) 中执行大量的状态相关的 Java 和表达式语言 (EL) 代码。JSF 还可以完美处理数值转换和验证。由于 JSF 与 GUI 使用相同的模型，因此它完全可以与界面开发工具协作。早期传统的客户端/服务器开发人员就认识到使用优秀工具构建界面远比人工编码效率高。

现在，JSF 是 Java EE 的一部分，它能与 JSP 无缝协作。它们共享一个统一的 EL，且 JSP 是 JSF 的默认渲染机制。更重要的是，还有大量的机会来扩展标准平台的用户界面功能。JSF 与 JSP 标记库的合并使创建可重用的用户界面组件更简便。而且，多数框架功能是即插即用的。开发人员可使用增强版本逐个替换它们，甚至可以使用备选方案（如 Spring）来更换整个框架。

用户界面对于使用 SOA 的强大功能是至关重要的。使它们更易于构建并为快速未来增强创造机会为在 SOA 上获得更好的投资回报奠定基础。

行业级 SOA 部署和管理

通过进一步简化业务服务及其界面的编写，Java EE 5 促进了更丰富的 SOA 生存法则。实际上，在企业内部建立这样的生存法则需要保持单个服务实例和广泛 SOA 环境的良好状况。单个 API 规范不能满足这个要求。企业需要行业级的实施方案，它具有任何关键任务 IT 组件所需的部署、管理和强健性。

用户界面对于使用 SOA 的强大功能是至关重要的。使它们更易于构建并为快速未来增强创造机会为在 SOA 上获得更好的投资回报奠定了基础。

Oracle WebLogic Server 已经证实可在企业环境中运行。最新版本将此成功经验扩展到 Java EE 5，因此客户可从熟悉的控制台管理所有新功能，同时使其尽可能的易于升级现有服务。升级意味着只需把现有服务重新部署到新的平台，而无需迁移。

Oracle WebLogic Server 已经证实可在企业环境中运行。最新版本将此成功经验扩展到 Java EE 5，因此客户可从熟悉的控制台管理所有新功能，同时使其尽可能的易于升级现有服务。

企业还面临将客户端升级到新版服务的挑战。几个特性简化了此成熟 SOA 的自然结果。相同服务的多个版本可运行在同一服务器或集群上。管理员可根据客户端群来分配版本访问，例如，仅为局域网或内部网段的客户端提供新版本。最后，管理员可以声明的方式指定客户端迁移策略。

Oracle WebLogic Server 还解决了其他几个客户问题。对于不支持多播的企业网，最新版提供了单播集群。运行集群时，以往将 Java 消息服务和 Java 事务 API 服务从一台机器迁移到另一台机器需要多项手动操作。最新版本提供了自动服务迁移功能。它还实现了控制台操作的记录和脚本功能。

随着 Oracle WebLogic Server 成为 IT 基础架构的标准部分，许多企业希望将其作为简单网络管理协议 (SNMP) 控制台的一部分进行监视。最新版本支持 SNMP 3，包括一个 JMX Runtime MBean 的 SNMP 视图。此外，用于管理服务器的 SNMP 代理现在为整个 Oracle WebLogic Server 域提供视图。这些改进兑现了 Oracle 的承诺：让 SOA 使整个 IT 组织，而不只是架构师和开发人员受益。

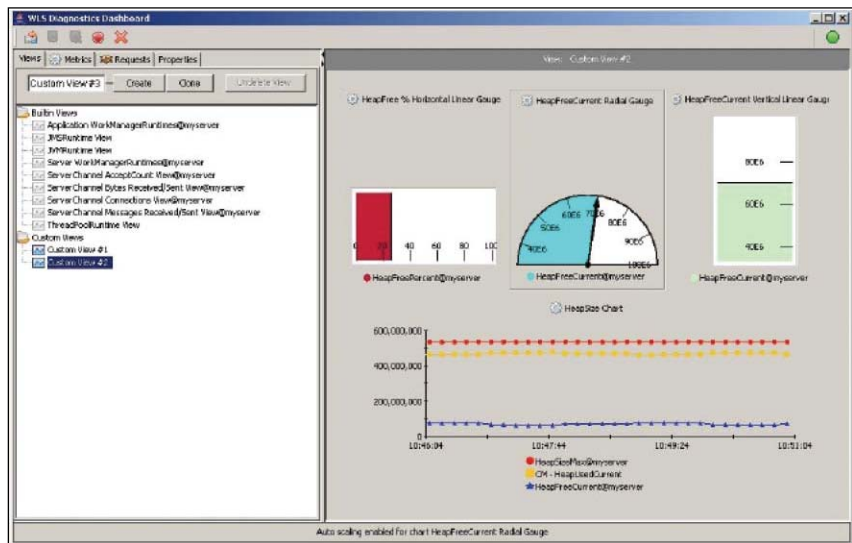


图 3: Oracle WebLogic Server 诊断框架

结论

Java EE 5 显著提高了 SOA 应用程序的开发速度。EJB 3.0 使开发人员更多关注域模型而更少关注中间件管道，从而简化了业务逻辑。JPA 简化了最常见的与将域对象映射到后端关系数据库相关的持久性管理任务。Web 界面 API 实现了与 SOA 更丰富、更灵活的用户交互，且 JAX-WS 在 SOA 中的服务间实现了更丰富、更灵活的协作。

Oracle WebLogic Server 是首个 Java EE 5 的产品级实施之一。它不仅遵守了标准的字面含义，而且还发扬了其精神实质 — 简化而不简单。开发人员得益于简化的API，而无需在 Oracle WebLogic Server 的基础上放弃任何经验验证的基础架构。Oracle TopLink 也被包含在内，它提供了一个 Java EE 持久性参考实施的高性能实例。Oracle WebLogic Server 的最终目的是帮助客户更好地运营业务并以最佳方式提供 Java EE 5 功能。

Oracle WebLogic Server 是最受欢迎、拥有最高效率和性能的 Java EE 平台。现在它率先为企业开发人员提供了产品级 Java EE 5 实施的强大功能。

Oracle WebLogic Server 是最受欢迎、拥有最高效率和性能的 Java EE 平台。现在它率先为企业开发人员提供了产品级 Java EE 5 实施的强大功能。现有客户可直接利用改进的 API，而新客户也获得了采用经验验证领先产品的信心。

甲骨文（中国）软件系统有限公司

北京总部

地址：北京市朝阳区建国门外大街1号，国贸大厦2座2208室
邮编：100004
电话：(86.10) 6535-6688
传真：(86.10) 6505-7505

北京上地6号办公室

地址：北京市海淀区上地信息产业基地，上地西路8号，
上地六号大厦D座702室
邮编：100085
电话：(86.10) 8278-7300
传真：(86.10) 8278-7373

上海分公司

地址：上海市卢湾区湖滨路222号，企业天地商业中心1号楼16层
邮编：200021
电话：(86.21) 2302-3000
传真：(86.21) 6340-6055

广州分公司

地址：广州市天河区北路233号，中信广场53楼5301&5308室
邮编：510613
电话：(86.20) 8513-2000
传真：(86.20) 3877-1026

成都分公司

地址：成都市人民南路二段18号，四川川信大厦20层A&D座
邮编：610016
电话：(86.28) 8619-7200
传真：(86.28) 8619-9573

大连分公司

地址：大连软件园东路23号，大连软件园国际信息中心2号楼
五层502号A区
邮编：116023
电话：(86.411) 8465-6000
传真：(86.411) 8465-6499

济南分公司

地址：济南市泺源大街150号，中信广场11层1113单元
邮编：250011
电话：(86.531) 8518-1122
传真：(86.531) 8518-1133

甲骨文软件研究开发中心（北京）有限公司

地址：北京市海淀区中关村软件园孵化器2号楼A座一层
邮编：100094
电话：(86.10) 8278-6000
传真：(86.10) 8282-6455

甲骨文研究开发中心（深圳）有限公司

地址：深圳市南山区高新南一道飞亚达大厦16层
邮编：518057
电话：(86.755) 8396-5000
传真：(86.755) 8601-3837

沈阳分公司

地址：沈阳市沈河区青年大街219号，华新国际大厦17层D单元
邮编：110016
电话：(86.24) 2396 1175
传真：(86.24) 2396 1033

南京分公司

地址：南京市玄武区洪武北路55号，置地广场19层1911室
邮编：210028
电话：(86.25) 8476-5228
传真：(86.25) 8476-5226

杭州分公司

地址：杭州市西湖区杭大路15号，嘉华国际商务中心702室
邮编：310007
电话：(86.571) 8717-5300
传真：(86.571) 8717-5299

西安分公司

地址：西安市高新区科技二路72号，零壹广场主楼1401室
邮编：710075
电话：(86.29) 8833-9800
传真：(86.29) 8833-9829

福州分公司

地址：福州市五四路158号，环球广场1601室
邮编：350003
电话：(86.591) 8801-0338
传真：(86.591) 8801-0330

重庆分公司

地址：重庆市渝中区邹容路68号，大都会商厦1611室
邮编：400010
电话：(86.23) 6370-8898
传真：(86.23) 6370-8700

深圳分公司

地址：深圳市南山区高新南一道飞亚达大厦16层
邮编：518057
电话：(86.755) 8396-5000
传真：(86.755) 8601-3837

甲骨文亚洲研发中心（上海）

地址：上海市杨浦区淞沪路290号，创智天地10号楼512-516单元
邮编：200433
电话：86-21-6095 2500
传真：86-21-6095 2555

