1 前言

SoTower 作为 JavaEE 应用开发平台，开发出的项目也存在一些 JavaEE 的弱势：系统各功能模块的文件都是交叉存放的，使得模块不易拆分，系统难以管理；开发人员每次发布、调试一个 class 文件都需要对整个应用进行重启，从而无法对上线的系统进行快速升级和方便维护等；应用的所有模块打包在一个 war 模块中，对于某些大型的应用项目其 war 包就会比较庞大，安装部署时极易出错；平台和应用作为一个整体部署在应用服务器上，使得平台与应用紧密耦合，导致了平台维护与升级的困难。

所有这些问题的根源就在于除了部分 Java 代码以 jar 的方式存在之外，其他大量的文件混杂的存放在 WebContent 下。虽然传统开发方式也提倡面向模块设计，但是 JavaEE 应用中的模块只是逻辑上的，不是物理隔离的，没有实现真正的模块化。要真正解决上述问题，就需要实现应用的模块化，把应用系统划分成一个个独立的业务模块，使模块之间物理隔离。

目前，业界最先进的模块化技术就是 OSGi，OSGi 的本质是将 Java 面向对象的开发转向面向组件和服务的开发。OSGi 框架提供了一套完善的机制用于管理和控制组件（Bundle）、服务（Service）的生命周期，以及组件和服务在其生命周期内的交互。

SoTower-DM 是一个把 SoTower 平台整合到 OSGi 环境的框架。通过 OSGi 技术，实现 SoTower 平台各个模块本身的模块化以及基于

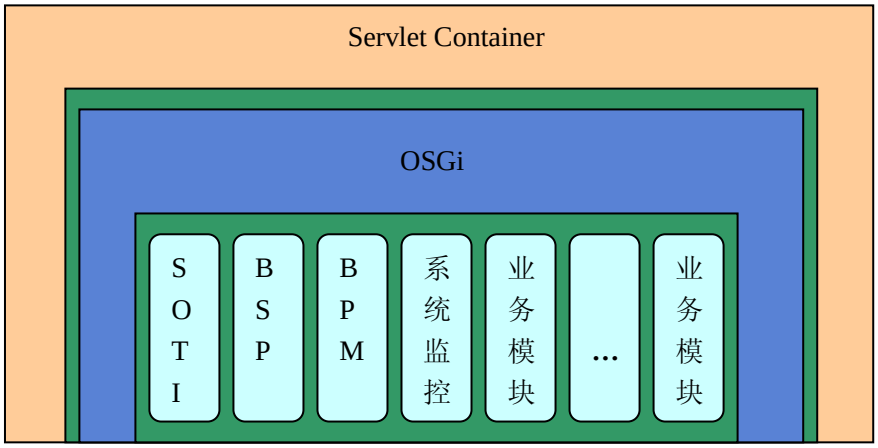
SoTower 平台开发的应用系统的模块化，使应用系统既保留 SoTower 平台的原有的优点，又具备先进的模块化、动态化的特性。

2 SoTower DM 组成与功能概述

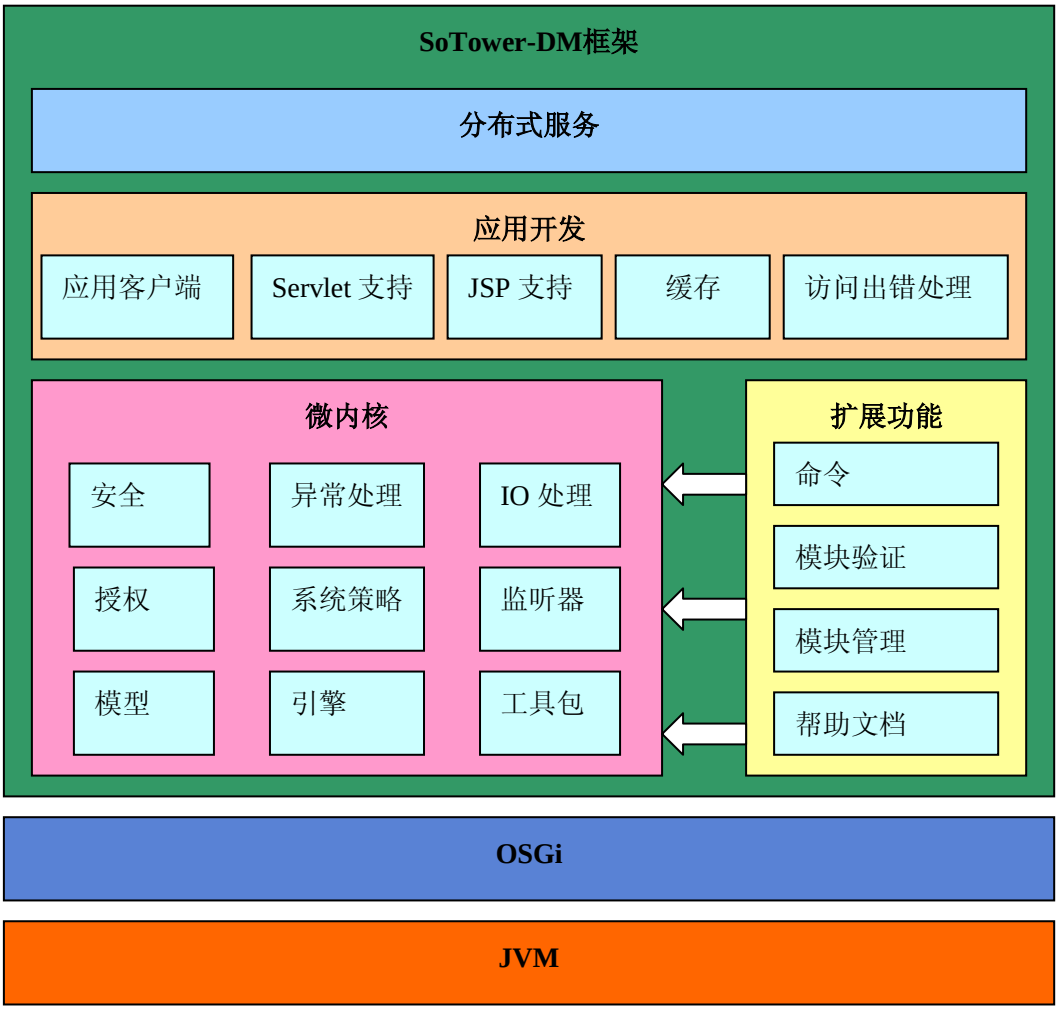
OSGi（Open Service Gateway Initiative ，开放服务网关协议）的初始目标是：使服务提供商通过住宅网关，为各种家庭智能设备提供服务。例如：通过 Web 页面控制咖啡机等。后来该平台逐渐成为一个为室内、交通工具、移动电话和其他环境下的所有类型的网络设备的应用程序和服务进行传递和远程管理的开放式服务平台。

OSGi 作为新的事实工业标准正在各领域蓬勃发展起来。最初的 OSGi 标准主要应用于 J2ME 和 J2SE。嵌入式系统的一个成功案例是 BMW 汽车的应用控制系统，该系统采用 OSGi 作为其底层架构，用来控制汽车上的音箱、灯光等等设备；在 Java 应用软件领域，Eclipse 的底层架构 Equinox 就是 OSGi 的一个实现。后来 OSGi 在 J2EE 环境下的应用也越来越广泛，目前在 Servlet 应用中使用 OSGi 有两种方式：Servlet Container in OSGi 和 OSGi in Servlet Container。第一种方式需要使得采用 JSP、Servlet 和 Struts 等 J2EE 技术的 Web 应用系统可以运行于 OSGi 环境中，这是一种纯 OSGi 的方式，在这种方式下，开发应用与开发普通的 OSGi 模块类似；第二种方式需要提供一个中间的桥梁将 OSGi 框架和 Servlet Container 联系起来，使得基于 OSGi 的应用可以嵌入到现有的 Web 服务器（如 Tomcat，Jetty 等）和应用服务器（如 Websphere，Weblogic 等）中。

SoTower-DM 就是一个把基于 OSGi 的应用系统嵌入到应用服务器中的框架，它的主要功能包括：第一是把 SoTower 平台整合到 OSGi 框架中，第二是把 OSGi 框架嵌入到 Servlet Container 中。应用系统由一个轻量级的 Web 应用和一个个的模块组成，轻量级的 Web 应用负责提供 Servlet 容器，模块运行在 OSGi 环境中。我们采用的 Equinox 作为 OSGi 框架的实现，SoTower 平台模块和应用系统的业务模块运行在 Equinox 中。最终由轻量级的 Web 应用、Equinox 实现、SoTower-DM 框架以及所有模块构成一个完整可运行的系统，部署在应用服务器上。



SoTower-DM 框架由以下功能结构组成：



- 微内核，启动 SoTower-DM 框架所需的最小模块集合；
- 扩展功能，对核心框架的一些扩展功能，通过服务的方式提供，可以启用和停止，不影响核心框架的运行；
- 应用开发，为在 OSGi 环境之上的运行企业级应用提供支持，包括 JSP、Servlet、页面资源、缓存、访问出错处理等；
- 分布式服务，把应用中各模块的功能发布成分布式服务，通过分布式服务实现跨系统的交互，从而实现整个应用以模块为粒度进行分布式部署。

2.1 应用系统模块化

2.1.1 平台与应用解耦

基于 SoTower-DM 的应用系统由一个个独立的模块组成，模块之间物理隔离，并且每个模块都包含了与当前功能相关的所有类型的文件。应用系统中的模块可以分为 2 种，一种是 SoTower 平台自身的模块，另一种是应用系统的业务模块。

SoTower 平台的模块由 SoTower 产品提供和维护，用户只需要选择所需要的平台版本号即可获得对应的平台模块，具体包括以下内容：

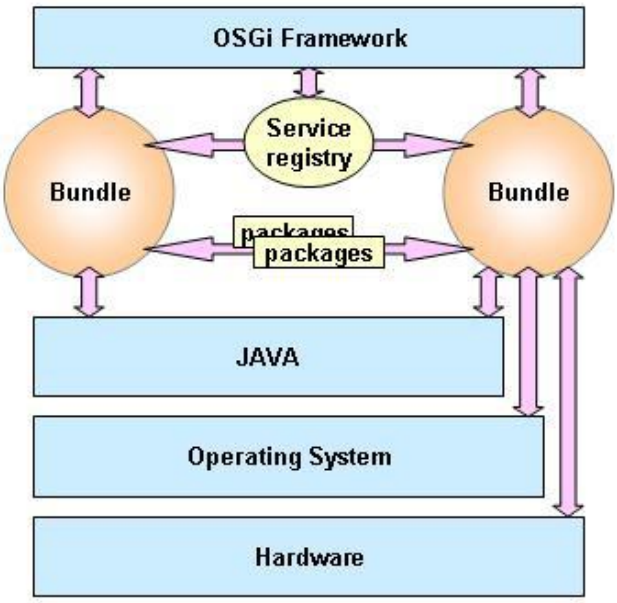
模块	说明
sotower-soti	SoTower 平台 SoTI 部分的所有 jar 文件集合
sotower-config	提供 SoTower 平台的原有默认配置
sotower-skins	提供 SoTower 平台的原有 js、css 以及图片
sotower-webtlds	支持 TLD 标签
sotower-presentation	统一注册 SoTower 平台的构件
sotower-datasource	提供数据源
sotower-persistence	提供访问数据库的持久层
sotower-bsp	SoTower 平台 BSS 部分
sotower-wf	SoTower 平台工作流部分
sotower-monitor-injector	SoTower 平台系统监控代码注入部分
sotower-monitor-common	SoTower 平台系统监控公共部分
sotower-monitor-server	SoTower 平台系统监控服务端
sotower-monitor-client	SoTower 平台系统监控客户端
sotower-apache	所有 Apache 辅助 jar 文件集合
sotower-hibernate	提供 Hibernate 框架

sotower-jdbc-driver	所有数据库 JDBC 驱动集合
sotower-jee	所有 JEE 相关 jar 文件集合
sotower-logger	包含 commons-logging 与 log4j 文件
sotower-mina	提供 Mima 通讯方式
sotower-others	其他辅助 jar 文件
sotower-spring	提供 Spring 框架
sotower-spring-mvc	提供 Spring-MVC 框架
sotower-tomcat	Tomcat 服务器的辅助 jar 文件集合
sotower-xml	提供处理 XML 的相关 jar

业务模块是用户根据应用系统的实际需求开发的模块。把应用系统按业务功能划分为一个个的模块，为每一个业务模块创建一个模块项目，模块之间物理隔离。每个模块都是功能完整的，并包含了所有的各类型文件，包括 Java 源代码、Spring 配置文件、页面文件、脚本文件以及当前模块要依赖的第三方 JAR 包等等。应用系统就不再是一个大的 Web 应用项目，而是由一个个的业务模块项目组成。

2.1.2 松耦合的模块间交互

模块与模块之间可以通过 package 和 service 这 2 种方式进行交互。



Package 方式首先需要使模块可以对外暴露自己的类，暴露的级别以包为单位。模块可以设置对外暴露哪些包，包中包含的类对外都是可见的。模块在对外暴露包的时候还要支持设置过滤条件，使得只有特定的模块项目可以访问暴露的包，或者除了特定的模块项目之外其他的模块项目都可以访问暴露的包。

模块之间的依赖可以分为 2 种级别，一种是模块级别，一种是包级别的。模块级别就是一个模块完全依赖另一个模块，模块可以访问被依赖模块对外暴露的所有的包中的类。包级别就是模块可以从所有其他模块对外暴露的包中选择要依赖哪个包，只有选择的包是可见的，同一个模块中其他未被选择的包仍是不可见，包级别的依赖粒度更小更精确。

Service 方式是指服务提供者模块可以把满足了某个接口的类发布为一个服务，并注册在 SoTower-DM 框架中，服务消费者模块就可以向 SoTower-DM 框架请求满足某个接口的服务，为了更精准的查找服务，SoTower-DM 将支持服务消费者模块在引用服务的时候设置过

滤条件，设置服务级别，当有多个服务满足条件时，可以优先选择高级别的服务，或者当有高级别的服务时自动替换低级别的服务。

2.1.3 可控的模块启动顺序

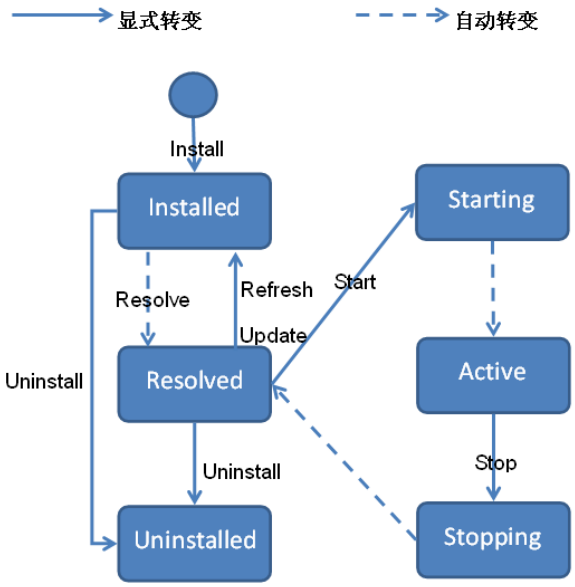
当系统由很多个模块组成时，因为模块之间存在依赖关系或者服务调用，各个模块的启动顺序就成了一个问题，所以 SoTower-DM 支持对模块的启动顺序进行设置，使得可以设置每个模块的启动级别和启动顺序。高级别的比低级别的模块先启动，同一个级别的模块按启动顺序依次启动。

应用系统模块化后，每个模块可以独立启停，所以 SoTower-DM 还支持业务模块的延迟启动，使得业务模块可以在第一次有用户请求的时候再启动，这样应用系统在启动时就可以只启动一些必要的模块，从而加快系统的启动速度。

2.2 应用系统动态化

2.2.1 模块的动态启动

支持对模块进行单独启停，每个模块具有生命周期，包括以下状态：INSTALLED（已安装）、RESOLVED（已解析）、STARTING（启动中）、ACTIVE（激活）、STOPPING（停止中）、UNINSTALLED（已卸载），用户可以安装、启动、更新、刷新、停止和卸载模块。

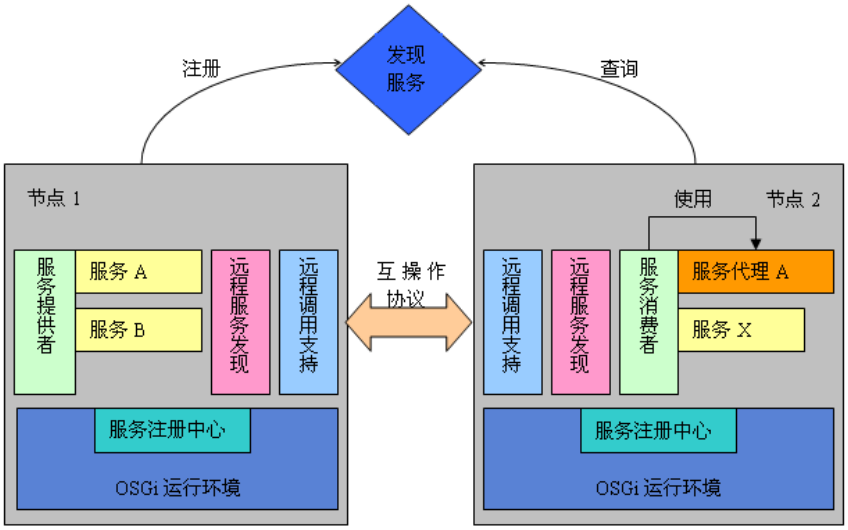


2.2.2 服务的动态更替

在 SoTower-DM 框架中模块之间可以通过服务发布与引用来实现交互，当服务提供者模块不可用时，系统能自动发现服务提供者模块的失效，并主动查找其他满足条件的服务。当有新的满足条件的服务出现时，会对当前的服务进行对比，看是否需要进行服务替换。

2.3 基于模块的分布式部署

基于 OSGi 技术的 SoTower 平台中，应用系统由一个个的业务模块组成，不同的业务模块有各自的业务逻辑，有的时候某个业务逻辑可能比较复杂，运行时比较占用资源，可能需要把这个复杂的业务逻辑独立出来，单独创建一个模块，并把它发布成服务供其他模块调用。提供服务的业务模块可以单独部署在一台服务器上运行，也可以同时部署在多个服务器上。通过分布式服务实现跨系统的相互协作，用于系统与系统之间的调用。



3 SoTower DM 的技术特色

3.1 与生俱来的 SOA 架构

面向服务的组件模型（Service-Oriented Component Model）的设计思想是 OSGi 的核心设计思想，OSGi 推崇系统采用模块的方式来划分，模块由多个构件（Component）来实现，构件通过对外提供服务接口和引用其他模块的服务接口来实现构件间的交互。从这个核心的设计思想上可以看出，基于 OSGi 实现的系统自然就是符合 SOA 体系架构的。SoTower-DM 又是基于 OSGi 来实现的，所以基于 SoTower-DM 开发的应用系统也是符合 SOA 思想的。它可以让一个模块导出服务，而其它的模块可以在不必了解源模块的任何信息的情况下消费这些导出的服务。正是由于 OSGi 的这种隐藏真实的服务实现类的能力，因此它为面向服务的应用提供了良好的类与接口的组合。

3.2 面向服务与组件的开发

在基于 SoTower-DM 构建系统时，模块内部的构件的通讯采用 Spring Bean 的方式进行通讯，推荐这些 Spring Bean 以类似 OSGi Service 的方式（即服务接口与实现组件的方式）进行编写；而模块之间的构件的通讯则必须采用 OSGi Service 的方式来实现。即源模块先开发一个功能构件，并把这个功能的实现类作为一个 Bean 注册到本模块的 Spring 容器，再把这个 Bean 注册到 OSGi 容器，作为一个 OSGi 服务对外发布，它可以注册在一个或多个接口下。目标模块可以向 OSGi 容器请求注册在某一接口下的服务，一旦它发现该服务，目标模块就会将该服务引入到本模块的 Spring 容器中，这样其他的 Bean 就可以引用这个服务，绑定到服务的接口，并能调用该接口中的方法。

3.3 易扩展的应用系统

SoTower-DM 在设计时提倡采用可扩展式的设计，即可通过系统中预设的扩展点来扩充系统的功能，有两种方式来实现：

3.3.1 引用服务的方式

通过在组件中允许引用服务接口的多个实现来实现组件功能的不断扩展，例如 A 组件的作用为显示菜单，其通过引用菜单服务接口来获取系统中所有的菜单服务，此时系统中有两个实现此服务的组件，分别为文件菜单组件和编辑菜单组件，那么 A 组件相应的就会

显示出文件菜单和编辑菜单，而当从系统中删除编辑菜单的组件时，A 组件显示的菜单就只剩文件菜单了，如此时再部署一个实现菜单服务接口的视图菜单组件模块到系统中，那么显示出来的菜单则会为文件、视图。

3.3.2 扩展模块的方式

原 SoTower 平台的所有文件都已经按照模块拆分到了各个模块中，如果业务应用子系统中需要修改过 SoTower 平台的文件，就可以创建扩展模块，用子系统中修改过的平台文件覆盖平台默认的文件。

3.4 可重用的企业模块仓库

使用 SoTower-DM 一个很大的意义就是它很大程度的提升了系统的可重用性，由于 OSGi 提供了规范的模块化和规范的模块交互机制，并具备了充分的语义，这样的话使得模块化级别的重用甚至是系统级的重用成为了可能。

建立公司级的模块仓库对于公司而言具备很大的意义，建立起模块仓库后，公司在进行每个项目之前都可通过模块仓库来寻找是否有相似功能的模块，这样就可以快速的搭建起项目的脚手架了，这对于公司项目的积累和共享是非常有帮助的，而这也是在使用 SoTower-DM 后能给公司带来的最明显的帮助，同时还可借助互联网上的各种公开的模块仓库，提升系统的开发速度，减少重复劳动。

4 SoTower DM 领先的技术价值

基于 SoTower-DM 的应用系统符合先进的 SOA 思想，并且具备模块化和动态化的特性，将为客户带来了众多突出的价值：

- 系统开发：为了能实现模块化，我们在设计系统时就能完全以模块的思想来设计系统，并以更标准化的要求来设计系统，使系统的接口与实现分离，模块高内聚低耦合。因为系统被划分成一个个模块，这样开发人员在开发系统时，就可以只关注自己负责的模块，而不用被不相关模块的问题所干扰，实现关注分离；
- 系统集成：系统的集成就可以看成是模块的简单叠加，提供系统集成效率；
- 系统部署：在部署系统时也不用等到所有的业务功能都实现了再部署，而是可以开发完成一个部署一个，实现系统分模块上线；
- 系统更新：实现了动态模块后，当我们要更新系统时就不需要每次都停止整个应用了，而是可以在运行期动态的安装、启动、更新、停止和卸载某个模块。