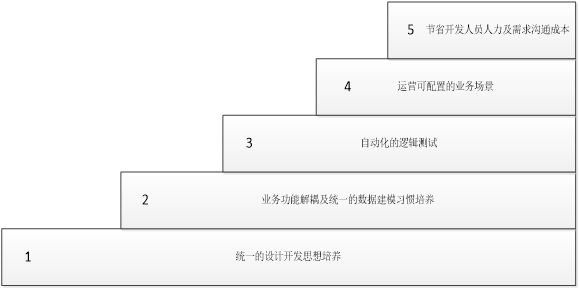
1. 总体思想

NFrame-java是NoahFrame通用框架的java版本，总体构建的思想是通过领域驱动设计的方法论将要解决的业务概念和业务规则转换为软件系统中的类型以及类型的属性和行为，通过合理运用面向对象的封装、继承和多态等设计要素，降低或隐藏整个系统的业务复杂性，提供更好的扩展能力，从而应对多变的现实业务问题。

1. 系统架构
   1. 架构目的



如上图所示：

目标1：主要的工作内容是在基础架构之上实现现有功能的对接，并对开发人员输出架构的开发思想。

目标2：主要的工作内容是对现有功能进行数据建模，抽象相关业务成为领域模型，提供可复用的模块接口。

目标3：主要的工作内容是建设针对模型、插件等系统部件的自动化测试环境，简化测试工作提高发布效率。

目标4：主要的工作内容是建设系统场景配置功能，运营可根据市场/需求变化在现有功能基础上快速建立场景业务。

目标5：主要的工作内容是建设流水线化的开发工作流程并积累技术/功能储备。

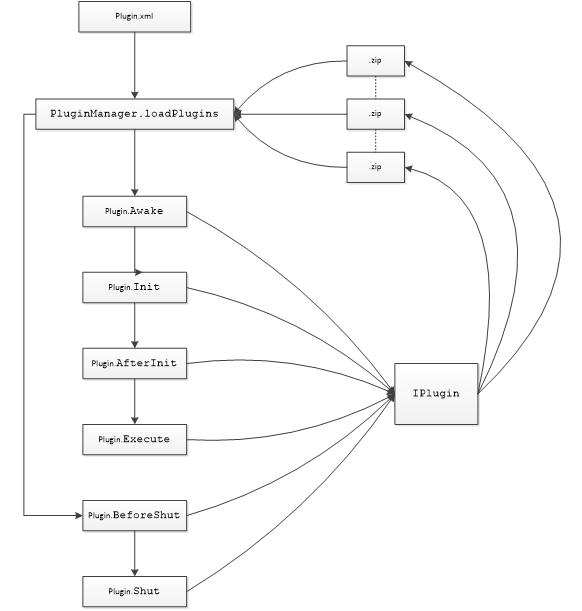
* 1. 领域驱动的架构设计

主要的原理是继承并使用ClassLoader的原理来动态加载jar包到JVM中进行运行。其中模块代码将以编译文件的形式打包到插件zip中，在PluginManager的管理下进行加载或清除到JVM中，并通过ClassLoader对象通过接口执行模块方法。



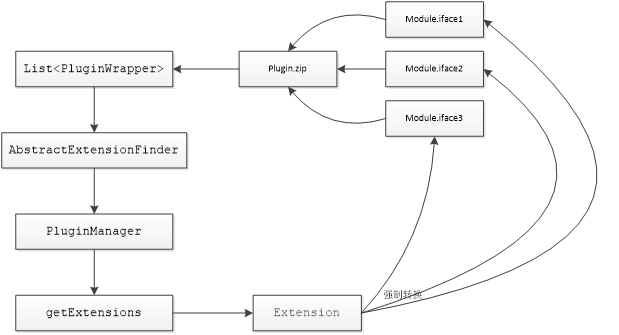
1. 利用java的ClassLoader特性实现了整个插件的基础设施环境建设。
2. 接口负责向用户提供信息接口。
3. 插件作为整个领域的载体提供加载运行的环境支持
4. 插件模块作为领域层实现的是业务规则。

插件的加载和管理功能框架图如下所示:



如图所示插件加载和管理主要通过PluginManager完成通过Plugin.xml的输入文件加载zip插件包然后通过awake、init、afterinit、execute、beforeshut、shut等方法通过IPlugin基类接口执行每个插件的相应方法，达到加载和管理插件的目的。

模块加载和执行的功能框架图如下所示：

模块加载和执行功能框架图

如图所示在插件Plugin中存在模块Module并引用了公共的接口iface，插件加载完成后系统通过获取插件对象，将插件对象传入abstractExtensionFinder类构建模块扩展点类ExtensionWrapper的map集合。其他程序可通过插件管理类取到扩展节点信息，并且通过公共接口定义可执行模块的接口实现。

1. 目录结构及使用
   1. 目录结构
      1. DataCfg

数据配置目录详细参考C++版本说明

* + 1. NFComm

通用的系统架构方法和库

* + - 1. NFCore

业务接口、回调及核心类、枚举定义

* + - 1. NFMessageDefine

通讯协议及消息定义

* + - 1. NFPluginLoader

系统插件加载及管理类定义

* + - 1. NFPluginModule

系统插件核心环境及ClassLoader重写、继承。

* + 1. NFSDK

系统核心插件业务，包含了log，kernel、config、net

* + 1. NFServer

服务提供入口测试

* + 1. NFTools
       1. NFFileProcess

文件操作通用方法

* + - 1. NFUtilProcess

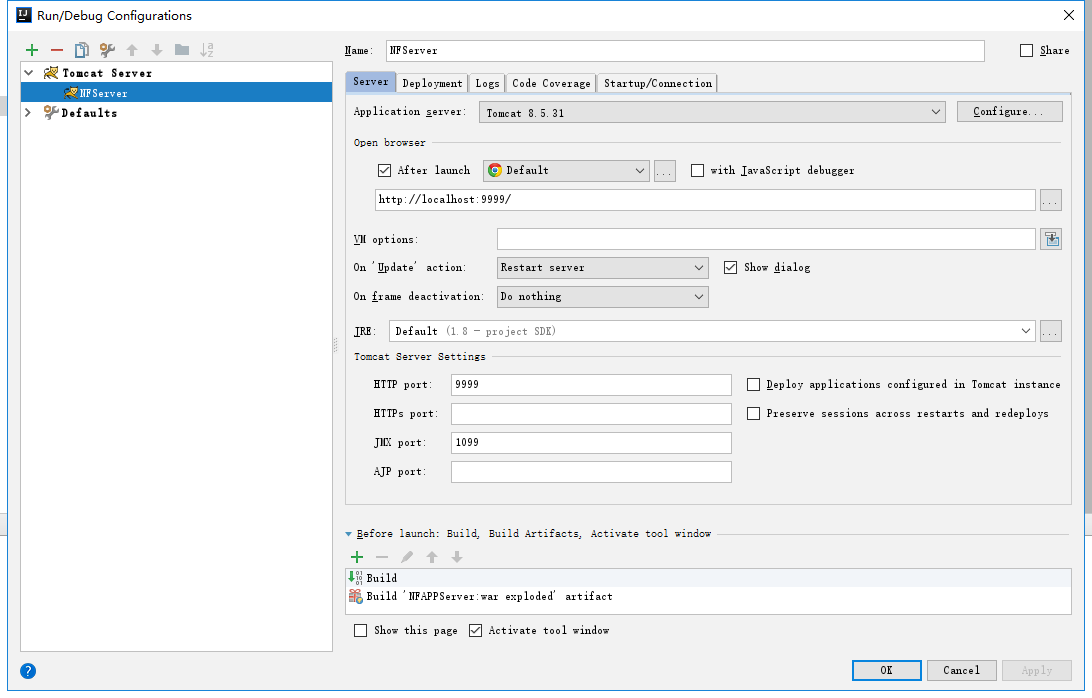
系统通用操作方法

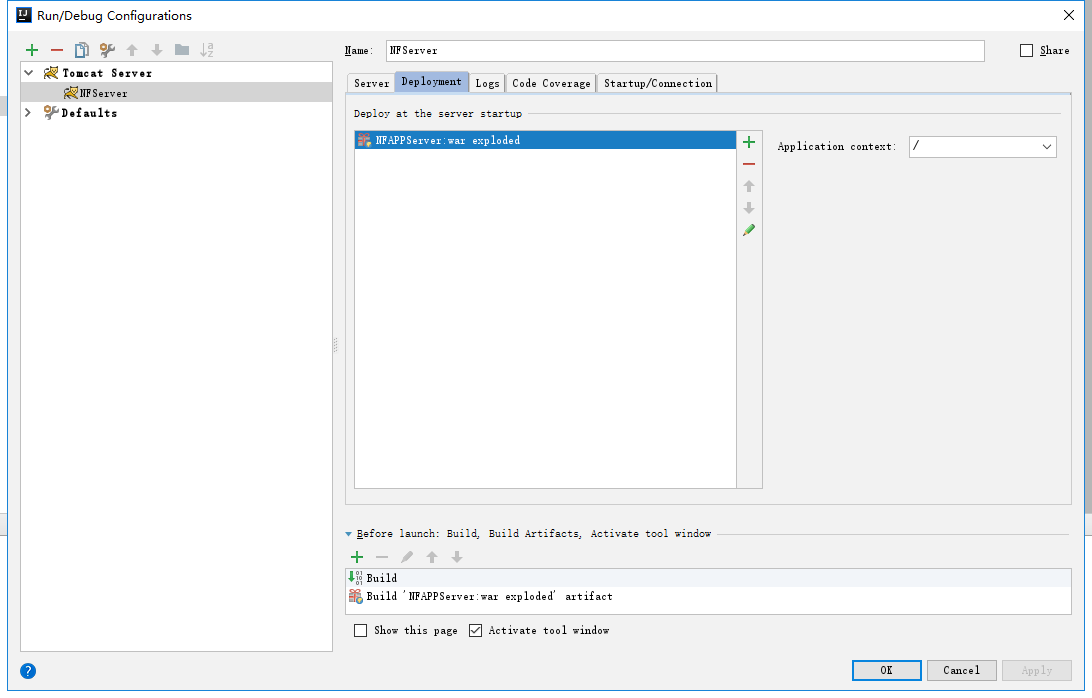
* + 1. Plugins

系统插件运行目录

* 1. 启动调试程序

在idea下配置tomcat环境如下图





然后将使用maven install生成的插件NFConfigPlugin-1.0.0.zip、NFKernelPlugin-1.0.0.zip、NFLogPlugin-1.0.0.zip等复制到文件夹plugins下启动调试即可，需要增加自己的插件请修改NFServer\NFAPPServer\src\main\resources下的Plugin.xml来增加插件

* 1. 启动部署程序

配置tomcat，并修改NFServer\NFAPPServer\src\main\resources下的sys.properties中的plugins\_dir和res\_dir已运行根目录相对目录的方式指向插件目录（plugins目录）和资源路径（DataCfg目录）然后启动tomcat即可。

下一篇内容：NFrame-java如何创建插件和模块