Broadv
www.broadvie

架构探险

从零开始写Java Web框架

和猿到架构狮的那点事儿，带您步入神奇的架构探险之旅

黄勇 著

中国工信出版集团 電子工業出版社

架 构 探 险

——从零开始写 Java Web 框架

黄 勇 著

電子工業出版社

Publishing House of Electronics Industry

北京 • BEIJING

内 容 简 介

本书首先从一个简单的 Web 应用开始，让读者学会如何使用 IDEA、Maven、Git 等开发工具搭建 Java Web 应用；接着通过一个简单的应用场景，为该 Web 应用添加若干业务功能，从需求分析与系统设计开始，带领读者动手完成该 Web 应用，完善相关细节，并对已有代码进行优化；然后基于传统 Servlet 框架搭建一款轻量级 Java Web 框架，一切都是从零开始，逐个实现类加载器、Bean 容器、IoC 框架、MVC 框架，所涉及的代码也是整个框架的核心基础。为了使框架具备 AOP 特性，从代理技术讲到 AOP 技术，从 ThreadLocal 技术讲到事务控制技术。最后对框架进行优化与扩展，通过对现有框架的优化，使其可以提供更加完备的功能，并以扩展 Web 服务插件与安全控制插件为例，教会读者如何设计一款可扩展的 Web 应用框架。

本书适合具备 Java 基础知识，熟悉 Web 相关理论，并想成为架构师的程序员阅读。

图书在版编目（CIP）数据

架构探险：从零开始写 Java Web 框架/黄勇著. —北京：电子工业出版社，2015.8
ISBN 978-7-121-26829-8

Ⅰ. ①架… Ⅱ. ①黄… Ⅲ. ①JAVA 语言－程序设计 Ⅳ. ①TP312

中国版本图书馆 CIP 数据核字（2015）第 176473 号

责任编辑：陈晓猛
印　　刷：北京天宇星印刷厂
装　　订：北京天宇星印刷厂
出版发行：电子工业出版社
　　北京市海淀区万寿路 173 信箱　邮编 100036
开　　本：787×980　1/16　印张：22.75　字数：509.6 千字
版　　次：2015 年 8 月第 1 版
印　　次：2015 年 10 月第 2 次印刷
印　　数：3001~5000 册　　定价：79.00 元

凡所购买电子工业出版社图书有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系，联系及邮购电话：（010）88254888。

质量投诉请发邮件至 zlts@phei.com.cn，盗版侵权举报请发邮件至 dbqq@phei.com.cn。

服务热线：（010）88258888。

前　言

记得在 2013 年 9 月，那时我工作不太忙，利用自己的闲暇时间，写了一款名为 Smart 的轻量级 Java Web 框架，并发布到“开源中国”（oschina.net）社区网站上。当时引发了同行们的大量关注，随后纷纷出现了其他大量关于“轻量级 Java Web 框架”的开源项目。

当然也有人觉得此类项目没有多大意义，毕竟 Spring 等框架已经足够成熟了，功能完全能够满足我们日常的开发需求，为什么还要“重复发明轮子”呢？

对此问题，我是这样认为的：

1．对于 Spring 框架而言，虽然它已经足够强大了，但也更加臃肿了，因为它提供的所有功能我们并非都需要。

2．市面上有很多优秀的开源框架，我们不妨取其精华，自己动手开发一款适合自身开发需求的框架。

3．国内开源环境与国外差距较大，我们需要从自身做起，才能带动身边更多的人一起投身到国内开源事业中去。

我并非是想让大家都去造轮子，而是想把这个造轮子的过程描述出来，让大家在这个过程有所收获。也许我的力量非常薄弱，但只要能涌现出一批热血的开源人，相信国内开源市场的前景一定会越来越好！

我认为自己的选择并没有错，于是我一边写代码，一边写博客，在短短的几个月里，写了上万行代码、上百篇博客（我的博客地址：http://my.oschina.net/huangyong/blog）。我想把自己在开发过程中遇到的问题与经验都记录下来，并且分享出去，让更多的人受益。

直到有一天，我接到了博文视点陈晓猛编辑的邀请，陈编辑想让我出一本书，选题让我自

己来定。当时我既欢喜又发愁，欢喜是由于终于有机会出版自己的书了，发愁是由于自己的经验与水平十分有限，难以写出好的作品，所以只能大胆尝试了。

在写这本书之前，我发现在国内介绍如何使用一些开源框架的书已经非常多了，但教会大家怎样从零开始写框架的书却非常少，因此我打算写一本这样的书——从零开始写 Java Web 框架。在写书前我备感压力，因为我只写过一点技术博客，还从未写过书。对于写书而言，我是非常缺乏经验的，总是担心自己写得不够好，要么写得太晦涩，要么写得太肤浅。反反复复逐字推敲，直到交稿的时候，我心里还是忐忑不安。

我在写这本书时下定了决心，面对的读者非常具体，他们必须具备一定的 Java 功底，尤其是 Java Web 方面的开发经验，他们都是希望步入架构师行列的程序员。所以，这本书的内容不能过于理论，而是需要将理论融入到实践中。全书以实现一款轻量级 Java Web 框架为主线，建议读者能亲手实践，这样才能学到更多有价值的东西。

本书内容

全书共 5 章，每章有先后顺序，建议读者按照章节顺序进行阅读。

第 1 章：从一个简单的 Web 应用开始，教会读者如何使用 IDEA、Maven、Git 等开发工具来搭建一个 Java Web 项目。

第 2 章：为 Web 应用添加业务功能，从需求分析与系统设计开始，带领读者动手开发一款简单的 Web 应用，并进行了相关细节完善与代码优化。

第 3 章：搭建轻量级 Java Web 框架，一切都是从零开始，逐个实现类加载器、Bean 容器、IOC 框架、MVC 框架，本章所涉及的代码也是整个框架的核心基础。

第 4 章：使框架具备 AOP 特性，从代理技术讲到 AOP 技术，从 ThreadLocal 技术讲到事务控制技术，通过阅读本章，读者可学会如何实现 AOP 框架，以及事务管理框架。

第 5 章：框架优化与功能扩展，通过对现有框架的优化，使框架提供更加完备的功能，并以扩展 Web 服务插件与安全控制插件为例，教会读者如何对框架进行扩展。

致谢

首先要感谢我的家人，特别是我的妻子，当占用大量休息时间写作的时候，她能够给予我极大的理解与支持，独自承担了所有的家务，以及照顾宝宝的生活，让我能够在没有任何打扰的环境下，全身心地投入到写作当中。如果没有她的帮助，这本书是无法完成的。

感谢开源中国社区网站，感谢创始人红薯，感谢曾经帮助我的朋友们，他们是：大漠真人、

july、小菜的粉丝、ipandage、蛙牛、Liuzh_533、罗盛力、Bieber、哈库纳、悠悠然然、黄亿华、Dead_knight、疑似一僧、杨唯浩、V 神、石头哥哥、乒乓狂魔、邹建芳、山哥、光石头（排名不分先后），没有你们的支持与鼓励，我是没有决心把开源做下去的，更不可能写出这本书。

感谢与我一起共事的同事们，让我体会到了团队的力量，在你们的身上我学到了许多宝贵的经验，也感谢你们为本书提供的宝贵建议。

感谢博文视点的编辑，这本书能够如期出版，离不开你们的敬业精神与一丝不苟的工作态度，我为你们点赞！

本书资源

本书下载资源请登录博文视点官网（www.broadview.com.cn/26829）中的“下载资源”进行下载。

读者俱乐部

欢迎加入本书读者俱乐部 QQ 群：179323031。

黄勇

2015 年 6 月于上海

目　　录

第 1 章　从一个简单的 Web 应用开始……1

正所谓“工欲善其事，必先利其器”，在正式开始设计并开发我们的轻量级 Java Web 框架之前，有必要首先掌握以下技能：

- 使用 IDEA 搭建并开发 Java 项目；
- 使用 Maven 自动化构建 Java 项目；
- 使用 Git 管理项目源代码。

1.1　使用 IDEA 创建 Maven 项目……3
1.1.1　创建 IDEA 项目……3
1.1.2　调整 Maven 配置……3
1.2　搭建 Web 项目框架……5
1.2.1　转为 Java Web 项目……5
1.2.2　添加 Java Web 的 Maven 依赖……6
1.3　编写一个简单的 Web 应用……10
1.3.1　编写 Servlet 类……10
1.3.2　编写 JSP 页面……11
1.4　让 Web 应用跑起来……12
1.4.1　在 IDEA 中配置 Tomcat……12

1.4.2 使用 Tomcat 的 Maven 插件……13
1.4.3 以 Debug 方式运行程序……13
1.5 将代码放入 Git 仓库中……14
1.5.1 编写.gitignore 文件……14
1.5.2 提交本地 Git 仓库……15
1.5.3 推送远程 Git 仓库……15
1.5.4 总结……16
第 2 章 为 Web 应用添加业务功能……17

我们将在这个应用的基础上增加一些业务功能，您将学会更多有关项目实战的技能，具体包括：

- 如何进行需求分析；
- 如何进行系统设计；
- 如何编写应用程序。

2.1 需求分析与系统设计……19
2.1.1 需求分析……19
2.1.2 系统设计……19
2.2 动手开发 Web 应用……21
2.2.1 创建数据库……22
2.2.2 准备开发环境……22
2.2.3 编写模型层……23
2.2.4 编写控制器层……25
2.2.5 编写服务层……27
2.2.6 编写单元测试……28
2.2.7 编写视图层……31
2.3 细节完善与代码优化……31
2.3.1 完善服务层……32
2.3.2 完善控制器层……59
2.3.3 完善视图层……60
2.4 总结……65

第 3 章　搭建轻量级 Java Web 框架……66

我们需要这样的框架，它足够轻量级、足够灵巧，不妨给它取一个优雅的名字——Smart Framework，本章我们就一起来实现这个框架。
您将通过本章的学习，掌握如下技能：
- 如何快速搭建开发框架；
- 如何加载并读取配置文件；
- 如何实现一个简单的 IOC 容器；
- 如何加载指定的类；
- 如何初始化框架。

3.1　确定目标……68
3.2　搭建开发环境……70
3.2.1　创建框架项目……70
3.2.2　创建示例项目……73
3.3　定义框架配置项……74
3.4　加载配置项……75
3.5　开发一个类加载器……78
3.6　实现 Bean 容器……87
3.7　实现依赖注入功能……90
3.8　加载 Controller……93
3.9　初始化框架……97
3.10　请求转发器……98
3.11　总结……109

第 4 章　使框架具备 AOP 特性……110

在本章中，您将学到大量有用的技术，具体包括：
- 如何理解并使用代理技术；
- 如何使用 Spring 提供的 AOP 技术；
- 如何使用动态代理技术实现 AOP 框架；
- 如何理解并使用 ThreadLocal 技术；
- 如何理解数据库事务管理机制；
- 如何使用 AOP 框架实现事务控制。

4.1　代理技术简介……112
4.1.1　什么是代理……112
4.1.2　JDK 动态代理……114
4.1.3　CGlib 动态代理……116

4.2 AOP 技术简介……118
4.2.1 什么是 AOP……118
4.2.2 写死代码……119
4.2.3 静态代理……120
4.2.4 JDK 动态代理……121
4.2.5 CGLib 动态代理……122
4.2.6 Spring AOP……124
4.2.7 Spring + AspectJ……136
4.3 开发 AOP 框架……142
4.3.1 定义切面注解……142
4.3.2 搭建代理框架……143
4.3.3 加载 AOP 框架……150
4.4 ThreadLocal 简介……158
4.4.1 什么是 ThreadLocal……158
4.4.2 自己实现 ThreadLocal……161
4.4.3 ThreadLocal 使用案例……163
4.5 事务管理简介……172
4.5.1 什么是事务……172
4.5.2 事务所面临的问题……173
4.5.3 Spring 的事务传播行为……175
4.6 实现事务控制特性……178
4.6.1 定义事务注解……178
4.6.2 提供事务相关操作……181
4.6.3 编写事务代理切面类……182
4.6.4 在框架中添加事务代理机制……184
4.7 总结……185
第 5 章 框架优化与功能扩展……186

本章将对现有框架进行优化，并提供一些扩展功能。通过本章的学习，您可以了解到：

- 如何优化 Action 参数；
- 如何实现文件上传功能；
- 如何与 Servlet API 完全解耦；
- 如何实现安全控制框架；

• 如何实现 Web 服务框架。
5.1 优化 Action 参数……188
5.1.1 明确 Action 参数优化目标……188
5.1.2 动手优化 Action 参数使用方式……188
5.2 提供文件上传特性……191
5.2.1 确定文件上传使用场景……191
5.2.2 实现文件上传功能……194
5.3 与 Servlet API 解耦……214
5.3.1 为何需要与 Servlet API 解耦……214
5.3.2 与 Servlet API 解耦的实现过程……215
5.4 安全控制框架——Shiro……219
5.4.1 什么是 Shiro……219
5.4.2 Hello Shiro……220
5.4.3 在 Web 开发中使用 Shiro……224
5.5 提供安全控制特性……230
5.5.1 为什么需要安全控制……230
5.5.2 如何使用安全控制框架……231
5.5.3 如何实现安全控制框架……242
5.6 Web 服务框架——CXF……261
5.6.1 什么是 CXF……261
5.6.2 使用 CXF 开发 SOAP 服务……262
5.6.3 基于 SOAP 的安全控制……278
5.6.4 使用 CXF 开发 REST 服务……291
5.7 提供 Web 服务特性……308
5.8 总结……329
附录 A Maven 快速入门……330
附录 B 将构件发布到 Maven 中央仓库……342

第 1 章

从一个简单的Web 应用开始

如果您已经掌握了 Java 的基础知识，理解了面向对象的基本概念，对 Java Web 开发有过一定的了解，比如会用 Servlet 与 JSP 开发一些简单的应用程序，那么本章的内容就是为这类您准备的。

现在我们打算使用业界最牛的 Java 开发工具 IntelliJ IDEA（简称 IDEA）开发一个简单的 Web 应用，该应用不包含任何业务功能，只是为了熟悉 Web 应用的开发过程与 IDEA 的使用方法。同时，我们也会使用业界最强大的项目构建工具 Maven 与代码版本控制工具 Git，利用这些工具让开发过程更加高效。

正所谓“工欲善其事，必先利其器”，在正式开始设计并开发我们的轻量级 Java Web 框架之前，有必要先掌握以下技能：

- 使用 IDEA 搭建并开发 Java 项目；
- 使用 Maven 自动化构建 Java 项目；
- 使用 Git 管理项目源代码。

假设您已经在 Windows 上安装了 IDEA、Maven、Git 这三个工具，现在要做的就是双击 IDEA 图标来启动它，如图 1-1 所示。

图 1-1　IDEA 启动界面

实际上 IDEA 有两个版本，一个是社区版（开源的个人版），另外一个是完全版（收费的企业版）。个人版提供的功能对于我们而言是基本够用的，可能对于 Web 开发来说有些不太方便，但足以满足我们基本的开发需求。不管使用 IDEA 的哪个版本，本书都是适用的。

下面我们就使用 IDEA 来创建一个基于 Maven 的 Java Web 项目。

如果此时不知道 Maven 是什么，或者听说过但不会使用它，那么强烈建议您通过“附录 A”来了解 Maven 是什么以及它的基本用法。

1.1　使用 IDEA 创建 Maven 项目

1.1.1　创建 IDEA 项目

我们无须单独下载 Maven，更不用将其集成到 IDEA 中，因为 IDEA 默认已经将其整合了。在 IDEA 中创建 Maven 项目非常简单，只需要按照以下步骤进行即可：

（1）单击“Create New Project”按钮，弹出 New Project 对话框。

（2）选择 Maven 选项，单击“Next”按钮。

（3）输入 GroupId（org.smart4j）、ArtifactId（chapter1）、Version（1.0.0），单击“Next”按钮。

（4）输入 Project name（chapter1）、Project location（C:\chapter1），单击“Finish”按钮。

需要说明的是，其中 GroupId 建议为网站域名的倒排方式，以便确保唯一性，类似于 Java 的包名。

说明：本书中出现的 GroupId（org.smart4j）所对应的域名 smart4j.org 是我购买的个人域名，您可使用自己喜欢的任意 GroupId，当然也可以与我保持一致。

按照以上操作，只需片刻之间，IDEA 就创建了一个基于 Maven 的目录结构，如图 1-2 所示。

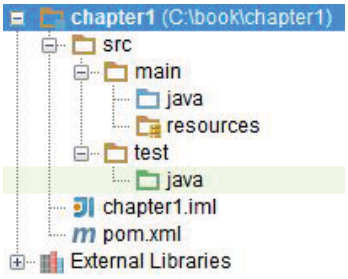

图 1-2　Maven 项目目录结构

1.1.2　调整 Maven 配置

当打开 Maven 项目配置文件（pom.xml）后，看到的应该是这样的配置（经笔者稍作整理）：

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns=http://maven.apache.org/POM/4.0.0
```

```
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
        http://maven.apache.org/xsd/maven-4.0.0.xsd">

    <modelVersion>4.0.0</modelVersion>

    <groupId>org.smart4j</groupId>
    <artifactId>chapter1</artifactId>
    <version>1.0.0</version>

</project>
```

技巧：当调整 pom.xml 后，需单击右上角气泡中的 Import Changes，使 Maven 配置立即生效，此操作表示“手动生效”。也可以单击右上角气泡中的 Enable Auto-Import，一旦对 pom.xml 文件进行修改就会自动导入，此操作表示“自动生效”。不过建议还是使用前者，即“手动生效”方式，这样会更加可控一些，至少能够确定自己做过那些事情。

以上只是 pom.xml 的基本配置，下面需要为它添加一些常用配置。

首先，需要统一源代码的编码方式，否则使用 Maven 编译源代码的时候就会出现相关警告。一般情况下，我们都使用 UTF-8 进行编码，需要添加配置如下：

```
...
<properties>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
</properties>
...
```

除了需要统一源代码编码方式以外，还需要统一源代码与编译输出的 JDK 版本。由于本书中使用 JDK 1.6 进行的开发，因此需要添加如下配置：

```
<build>
    <plugins>
        <!-- Compile -->
        <plugin>
            <groupId>org.apache.maven.plugins</groupId>
```

```
            <artifactId>maven-compiler-plugin</artifactId>
            <version>3.3</version>
            <configuration>
                <source>1.6</source>
                <target>1.6</target>
            </configuration>
        </plugin>
    </plugins>
</build>
```

所有的 plugin 都需要添加到 build/plugins 标签下，Maven 插件或下文中所出现的 Maven 依赖都来自于 Maven 中央仓库，可以通过以下链接访问：

```
http://search.maven.org/
```

如果说上面添加的两个配置是必需的，那么下面这个配置应该就是可选的。如果在使用 Maven 打包时想跳过单元测试，那么可以添加如下插件：

```
<!-- Test -->
<plugin>
   <groupId>org.apache.maven.plugins</groupId>
   <artifactId>maven-surefire-plugin</artifactId>
   <version>2.18.1</version>
   <configuration>
      <skipTests>true</skipTests>
   </configuration>
</plugin>
```

至此，一个 Maven 项目就搭建完毕了，下面我们就要在这个项目中添加有关 Java Web 的代码。

1.2　搭建 Web 项目框架

1.2.1　转为 Java Web 项目

目前，在 pom.xml 中还没有任何的 Maven 依赖（dependency），随后会添加一些 Java Web 所需的依赖，只不过在添加这些依赖之前，有必要先将这个 Maven 项目调整为 Web 项目结构。

提示：可以简单地将 Maven 依赖理解为 jar 包，只不过 Maven 依赖具备传递性，只需配置某个依赖，就能自动获取该依赖的相关依赖。

只需按照以下三步即可实现：

（1）在 main 目录下，添加 webapp 目录。

（2）在 webapp 目录下，添加 WEB-INF 目录。

（3）在 WEB-INF 目录下，添加 web.xml 文件。

此时，IDEA 给出一个提示，如图 1-3 所示。

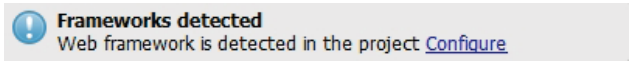

图 1-3 IDEA 框架检测提示

表示 IDEA 已经识别出目前我们使用了 Web 框架（即 Servlet 框架），需要进行一些配置才能使用。单击“Configure”按钮，会看到一个确认框，单击“OK”按钮即可将当前项目变为 Web 项目。

这里打算使用 Servlet 3.0 框架，所以在 web.xml 中添加如下代码：

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns="http://java.sun.com/xml/ns/javaee"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
        http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd"
        version="3.0">
</web-app>
```

实际上，使用 Servlet 3.0 框架是可以省略 web.xml 文件的，因为 Servlet 无须在 web.xml 里配置，只需通过 Java 注解的方式来配置即可，下面会描述具体的用法。

1.2.2 添加 Java Web 的 Maven 依赖

由于 Web 项目是需要打 war 包的，因此必须在 pom.xml 文件里设置 packaging 为 war（默认为 jar），配置如下：

```
<packaging>war</packaging>
```

接下来就需要添加 Java Web 所需的 Servlet、JSP、JSTL 等依赖了，添加如下配置：

```
<dependencies>
    <!-- Servlet -->
    <dependency>
        <groupId>javax.servlet</groupId>
        <artifactId>javax.servlet-api</artifactId>
        <version>3.1.0</version>
        <scope>provided</scope>
    </dependency>
    <!-- JSP -->
    <dependency>
        <groupId>javax.servlet.jsp</groupId>
        <artifactId>jsp-api</artifactId>
        <version>2.2</version>
        <scope>provided</scope>
    </dependency>
    <!-- JSTL -->
    <dependency>
        <groupId>javax.servlet</groupId>
        <artifactId>jstl</artifactId>
        <version>1.2</version>
        <scope>runtime</scope>
    </dependency>
<dependencies>
```

需要说明的是：

（1）Maven 依赖的“三坐标”（groupId、artifactId、version）必须提供。

（2）如果某些依赖只需参与编译，而无须参与打包（例如，Tomcat 自带了 Servlet 与 JSP 所对应的 jar 包），可将其 scope 设置为 provided。

（3）如果某些依赖只是运行时需要，但无须参与编译（例如，JSTL 的 jar 包），可将其 scope 设置为 runtime。

为了便于阅读，以下是 pom.xml 的完整代码：

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
        http://maven.apache.org/xsd/maven-4.0.0.xsd">
```

```
    <modelVersion>4.0.0</modelVersion>

    <groupId>org.smart4j</groupId>
    <artifactId>chapter1</artifactId>
    <version>1.0.0</version>
    <packaging>war</packaging>

    <properties>
        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
    </properties>

    <dependencies>
        <!-- Servlet -->
        <dependency>
            <groupId>javax.servlet</groupId>
            <artifactId>javax.servlet-api</artifactId>
            <version>3.1.0</version>
            <scope>provided</scope>
        </dependency>
        <!-- JSP -->
        <dependency>
            <groupId>javax.servlet.jsp</groupId>
            <artifactId>jsp-api</artifactId>
            <version>2.2</version>
            <scope>provided</scope>
        </dependency>
        <!-- JSTL -->
        <dependency>
            <groupId>javax.servlet</groupId>
            <artifactId>jstl</artifactId>
            <version>1.2</version>
            <scope>runtime</scope>
        </dependency>
    </dependencies>

    <build>
```