

# Spring

1. CharacterEncodingFilter 过滤器虽然名为过滤器，实际上并不会过滤请求，只是将非指定字符编码的请求转化为指定编码后放行。

```
<filter>
  <filter-name>CharacterEncodingFilter</filter-name>
  <filter-class>org.springframework.web.filter.CharacterEncodingFilter</filter-
class>
  <init-param>
    <param-name>encoding</param-name>
    <param-value>utf-8</param-value>
  </init-param>
</filter>
```

- 要注意的是，这个过滤器只针对请求中的数据乱码，而无法解决jsp等响应数据的乱码。
2. 所谓的过滤器和拦截器，其实现事实上就是将请求类和响应类作为参数传递到相关的处理函数中，然后在这个函数中对这两个类进行操作。例如，在安全认证的实现中，就是在处理函数中查找请求类中是否带有用户信息，如果没有或用户信息出错，则将错误信息写到响应类中返回给用户。
  3. 分析项目源码，首先需要从项目的请求处理流程的第一步开始，按照请求处理流程一步一步地分析。例如，分析web项目源码首先需要从URL解析相关的源码开始分析。
  4. 在多模块项目中，对于一些可以公用的类或函数，可以将其独立出来，成为一个单独的Spring boot项目，然后在pom.xml中以依赖的方式导入其他项目中。
  5. Spring项目中经用用到的java类有几种：
    - control类。
    - 拦截器类：用于登陆认证等。
    - 各种外部框架需要的类，例如mybatis的pojo类和Mapper接口类等。
    - 以及各种进行逻辑处理的类。
  6. 使用拦截器需要两步：

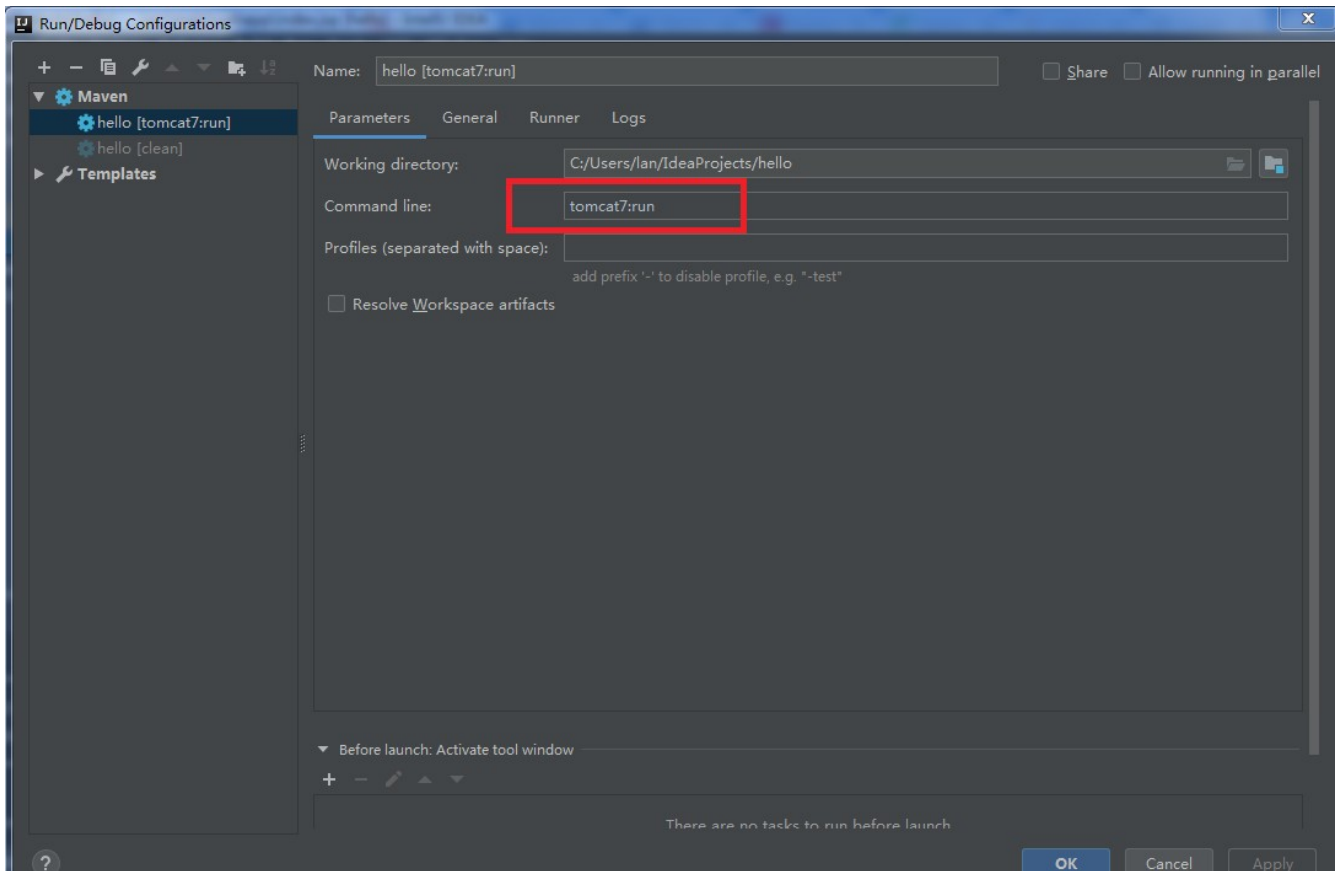
- 继承并扩展相关拦截器接口。
- 在 Springmvc.xml 中设置<mvc:interceptors>。

```
<mvc:interceptors>
    <mvc:interceptor>
        <!-- 拦截订单类请求 -->
        <mvc:mapping path="/order/**"/>
        <bean class="com.xuebusi.portal.interceptor.LoginInterceptor"/>
    </mvc:interceptor>
</mvc:interceptors>
```

7. 在使用 Idea 做项目时，Idea 实质上是使用 Maven 来做项目管理，例如 pom.xml、编译项目等都是 Maven 的内容。
8. jsp 文件默认的 ISO-8859-1 字符集中无中文字符，所以在 jsp 文件中出现中文时会出现乱码，这时需要指定文件字符编码：

```
<%@ page contentType="text/html; charset=utf-8"%>
<html>
<body>
<h2>测试</h2>
</body>
</html>
```

9. idea 社区版本本身并不集成 spring 和 tomcat, spring 需要自己在 pom.xml 中添加依赖, 而 tomcat 以插件的形式来使用, 所以调试时需要手动添加 “mvn tomcat7:run” 命令。



10. 在 xml 文件中, 可能会需要使用某些命名空间, 如:






```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:p="http://www.springframework.org/schema/p"
  xmlns:context="http://www.springframework.org/schema/context"
  xmlns:mvc="http://www.springframework.org/schema/mvc"
  xsi:schemaLocation="http://www.springframework.org/schema/beans
    http://www.springframework.org/schema/beans/spring-beans-3.1.xsd
    http://www.springframework.org/schema/context
    http://www.springframework.org/schema/context/spring-context-3.1.xsd
    http://www.springframework.org/schema/mvc
```

```
http://www.springframework.org/schema/mvc/spring-mvc-4.0.xsd">
```

```
<context:component-scan base-package="com.pikaq"/>  
<bean id="xxx" class="xxx.xxx.xxx.Xxx">  
<property name="xxx" value="xxxx"/>  
</bean>  
</beans>
```

其中：

- **xmlns**：是 XML Namespace 的缩写，译为“XML 命名空间”。后面的链接为命名空间的地址。对于上面的“xmlns=”开头的那个命名空间，它是默认的命名空间，默认命名空间不需要写前缀。
  - **xmlns:xsi**：该栏表示使用 xsi 来作为 xml 实例命名空间的前缀，之后的“xsi:schemaLocation”就是该前缀的使用。
  - **xmlns:xxx**：xxx 表示命名空间前缀，在使用命名空间的元素时，需要使用类似<xxx:sss>的形式，其中的 sss 表示命名空间的元素。例如<context:component-scan>中的 context 就是命名空间前缀，而 component-scan 是命名空间的元素。
  - **xsi:schemaLocation**：值由一个 URI 引用对组成，两个 URI 之间以空白符分隔。第一个 URI 是名称空间的名字，第二个 URI 是相关命名空间文档的位置。
- xml 命名空间的根地址为 <http://www.springframework.org/schema/>，需要使用哪个 spring 模块的 xml 元素，就在里面找，然后导入，例如 jdbc 的命名空间地址为 <http://www.springframework.org/schema/jdbc>
- 11.Spring 默认的视图处理类是 DispatcherServlet，如果不需要修改 jsp 文件的存放位置等内容，就不必在 web.xml 中重新定义。
  - 12.Spring 默认的直接访问不经过控制器，所以要避免直接访问。所以通常将 jsp 文件存放到 WEB-INF 目录下，但这样无法直接访问，但需要配置控制器。
  - 13.html 的 js、css 和 images 等文件不能放到 WEB-INF 里面，因为该目录的文件需要经过控制器才能解释。所以这些静态文件应该放在和 WEB-INF 同级的目录里。

 css	2018/12/16 0:32	文件夹	
 images	2018/12/16 0:32	文件夹	
 js	2018/12/16 0:32	文件夹	
 WEB-INF	2018/12/16 0:32	文件夹	
 category.json	2017/1/5 10:47	JSON 文件	30 KB

14. 在一个普通项目，<context:component-scan>和<mvc:annotation-driven />是必须的，因为一个作用是开启自动扫描，一个是开启注解。

15. resources 目录下的所有子目录都会成为 classes 的子目录。所以可以使用类似“classpath:xxx”之类的语句。

16. 当 DispatcherServlet 的 URL 匹配模式为 “\*.html” 等扩展名匹配的时候，RequestMapping 中设置的扩展名是可有可无，但实际访问的时候需要加上扩展名，否则请求会被拦截而返回 404。

web.xml

```
.....
<servlet>
  <servlet-name>aiairedian</servlet-name>
  <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
  <init-param>
    <param-name>contextConfigLocation</param-name>
    <param-value>classpath:spring/springmvc.xml</param-value>
  </init-param>
  <load-on-startup>1</load-on-startup>
</servlet>
<servlet-mapping>
  <servlet-name>aiairedian</servlet-name>
  <url-pattern>*.html</url-pattern>
</servlet-mapping>
</web-app>
```

controller

```
package com.aiairedian;
```

```
import org.springframework.stereotype.Controller;
```

```

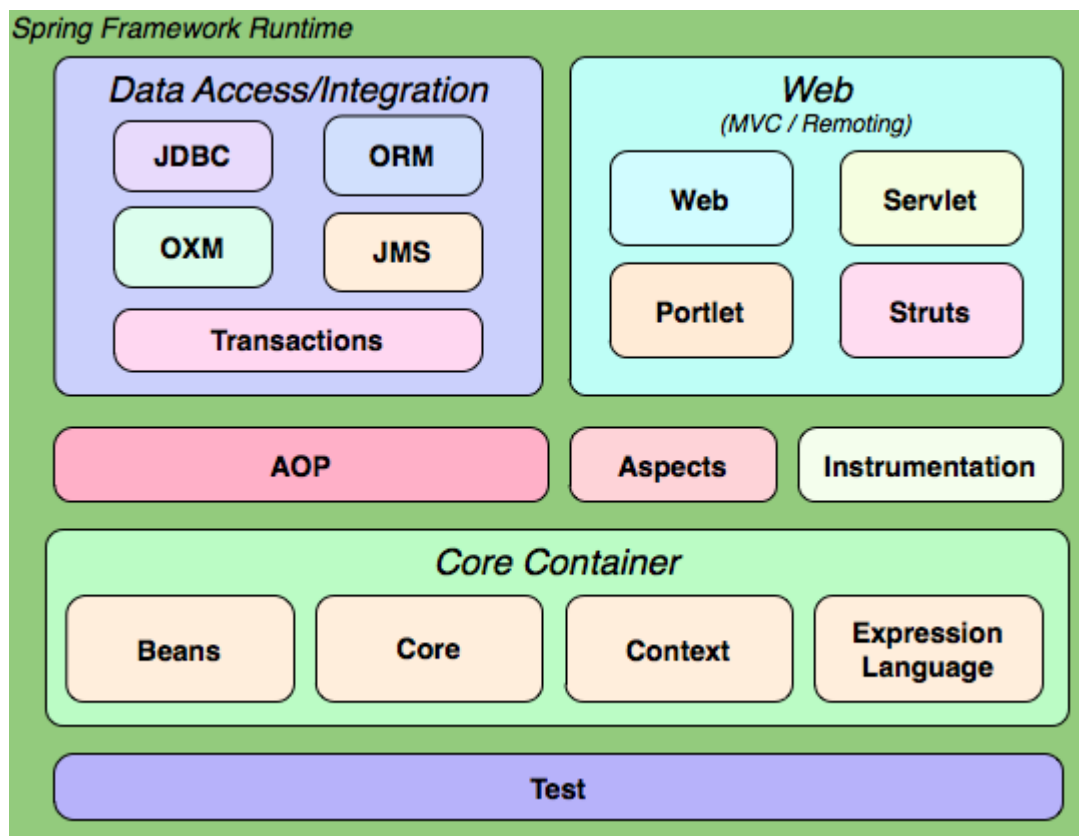
import org.springframework.web.bind.annotation.RequestMapping;

@Controller
public class login {
    @RequestMapping("/login.html")
    public String login() {
        return "login";
    }

    @RequestMapping("/index")
    public String index(){
        return "index";
    }
}

```

17. 想要了解 Spring 各种依赖，必须知道 Spring 的结构，下面是 Spring3 的结构图：

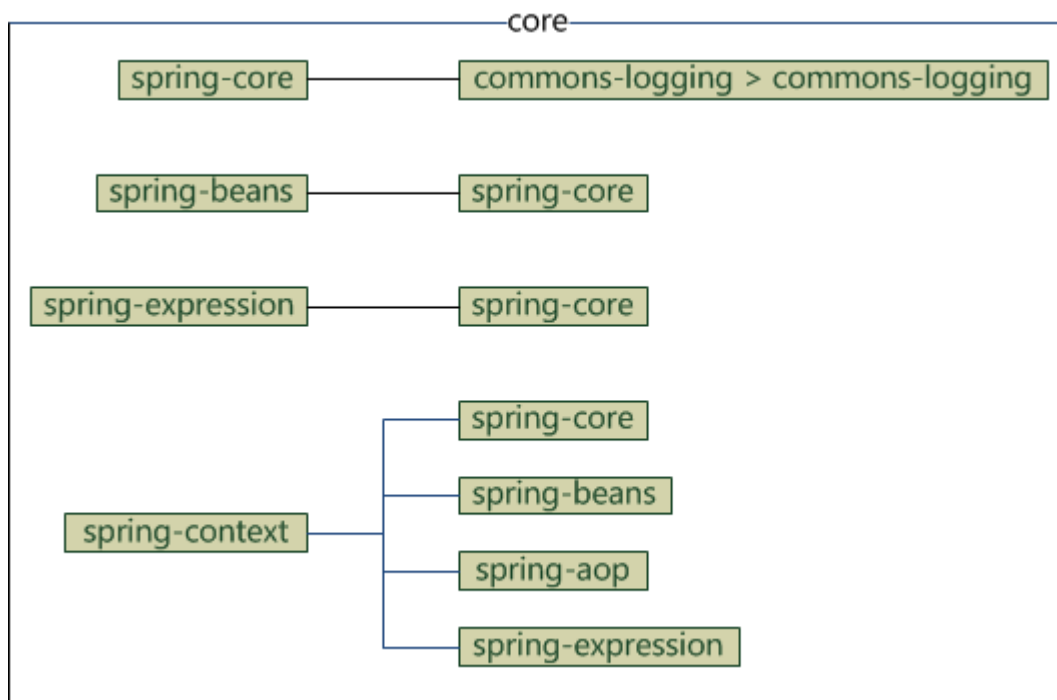


图中将 spring 分为 5 个部分：core、aop、data access、web、test，图中每个圆角矩形都对应一个 jar。

(1) core 部分包含 4 个模块：

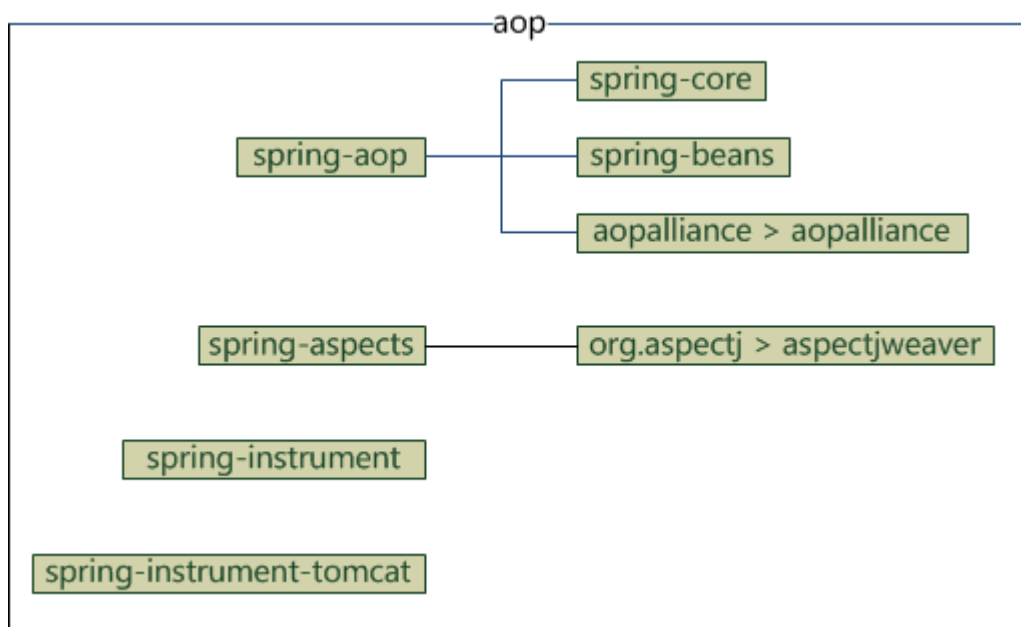
- spring-core：依赖注入 IoC 与 DI 的最基本实现
- spring-beans：Bean 工厂与 bean 的装配
- spring-context：spring 的 context 上下文即 IoC 容器
- spring-expression：spring 表达式语言

下面这种图是这些依赖相互之间的依赖关系，例如 spring-beans 就依赖于 spring-core：



(2) aop 部分包含 4 个模块

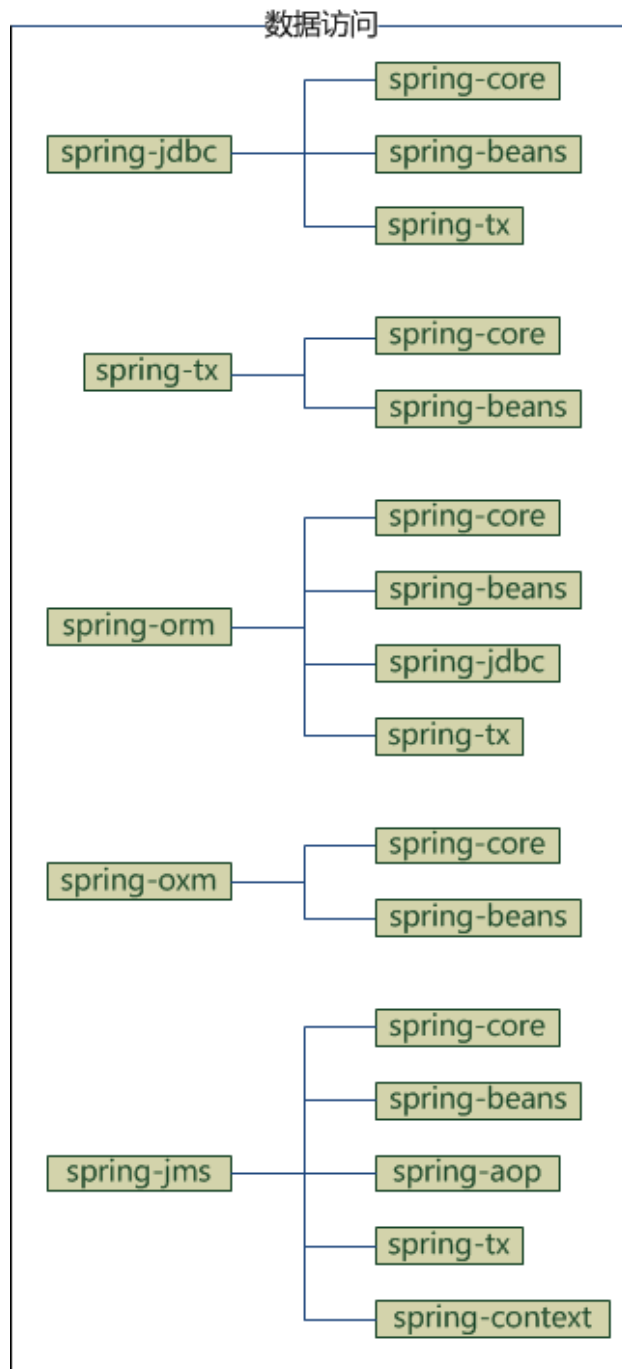
- spring-aop：面向切面编程。
- spring-aspects：集成 AspectJ。
- spring-instrument：提供一些类级的工具支持和 ClassLoader 级的实现，用于服务器。
- spring-instrument-tomcat：针对 tomcat 的 instrument 实现。



(3) data access 部分包含 5 个模块

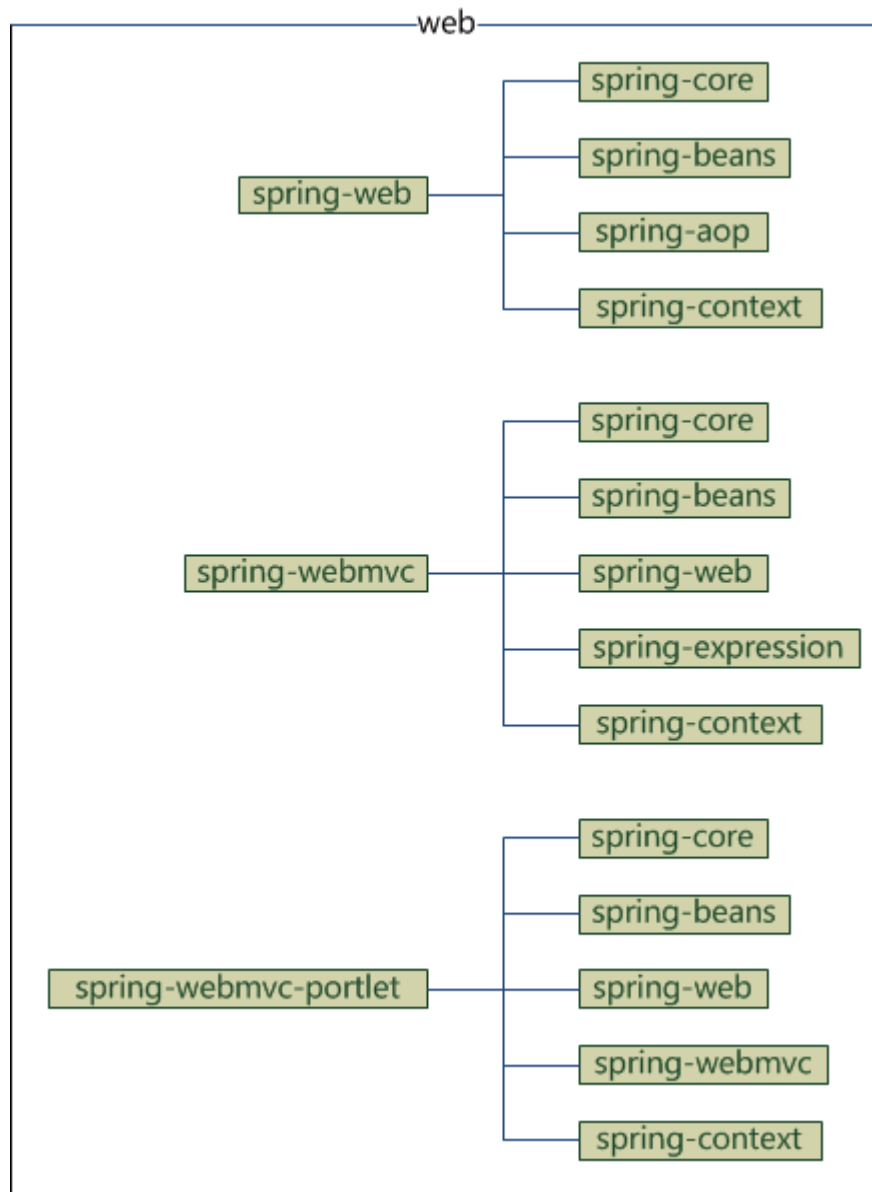
- spring-jdbc: jdbc 的支持
- spring-tx: 事务控制
- spring-orm: 对象关系映射, 集成 orm 框架
- spring-oxm: 对象 xml 映射
- spring-jms: java 消息服务





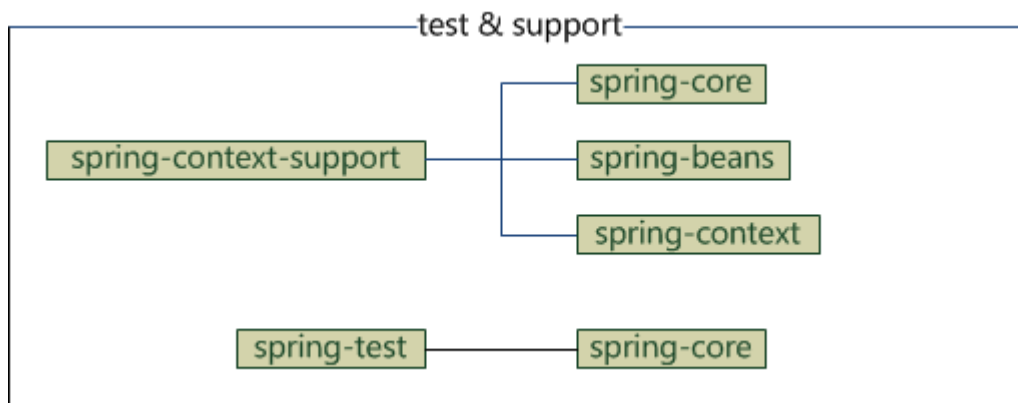
(4) web 部分包含 4 个模块

- spring-web: 基础 web 功能, 如文件上传。
- spring-webmvc: mvc 实现。
- spring-webmvc-portlet: 基于 portlet 的 mvc 实现。
- spring-struts: 与 struts 的集成, 不推荐, spring4 不再提供。



(5) test 部分只有 spring-test 一个模块，提供 junit 与 mock 测试功能。

(6) spring-context-support 模块：spring 额外支持包，比如邮件服务、视图解析等。



18.在添加依赖时，假设需要 spring-beans 依赖，而 spring-beans 又依赖于 spring-core，我们在 pom.xml 文件中添加了 spring-beans 之后无需再添加 spring-core，因为 Maven 会自动添加 spring-core 依赖。也就是说，Maven 会自动为我们解决嵌套依赖的问题。

19.连接池只是 Spring 的一个组件，其实只是一个 bean，所以 mybatis 不需要与连接池进行整合。

20.JDBC 中 mysql 的 url 格式类似如下：

```
jdbc.url=jdbc:mysql://localhost:3306/aiairedian?characterEncoding=utf-8&serverTimezone=GMT%2B8
```

其中的 aiairedian 是 mysql 中的数据库名，问号后面的是参数，serverTimezone 是设置时区，这里为东 8 区，不设置会出错。

21.在使用连接池连接 mysql 时，需要引入 com.mysql.jdbc.Driver 所在的包 mysql-connector-java，以提供 mysql 支持，否则会报错无法找到 com.mysql.jdbc.Driver。

➤ 也就是说，在使用连接池时，需要引入至少两个依赖：连接池依赖和相关的数据库依赖。

22.Druid 连接池支持网络监控，监控设置如下：

web.xml

```

.....
<!-- 连接池 启用 Web 监控统计功能 start-->
<filter>
    <filter-name>DruidWebStatFilter</filter-name>
    <filter-class>com.alibaba.druid.support.http.WebStatFilter</filter-class>

```

```

    <init-param>
      <param-name>exclusions</param-name>
      <param-value>*.js,*.gif,*.jpg,*.png,*.css,*.ico,/druid/*</param-value>
    </init-param>
  </filter>
  <filter-mapping>
    <filter-name>DruidWebStatFilter</filter-name>
    <url-pattern>/*</url-pattern>
  </filter-mapping>

  <servlet>
    <servlet-name>DruidStatView</servlet-name>
    <servlet-class>com.alibaba.druid.support.http.StatViewServlet</servlet-class>
  </servlet>
  <servlet-mapping>
    <servlet-name>DruidStatView</servlet-name>
    <url-pattern>/druid/*</url-pattern>
  </servlet-mapping>
</web-app>

```

## spring 上下文 xml

```

<context:property-placeholder location="classpath:resource/*.properties" />
<bean id="dataSource" class="com.alibaba.druid.pool.DruidDataSource" init-
method="init" destroy-method="close">
  .....
  <property name="filters" value="stat" />
  .....
</bean>

```

监控端的网址为 “http://localhost:8080/druid/index.html”。

- 注意 Druid 的 url 匹配模式和 Spring 的不一样，Druid 的匹配模式为 “/\*”，而 Spring 的匹配模式为 “/”。

23. 在时候某些插件依赖于某些 jar 包，可以使用<plugin>的<dependencies>子标签来为插件添加依赖：

```

<plugin>
  <groupId>org.mybatis.generator</groupId>

```

```
<artifactId>mybatis-generator-maven-plugin</artifactId>
<version>1.3.7</version>
<dependencies>
    <dependency>
        <groupId>mysql</groupId>
        <artifactId>mysql-connector-java</artifactId>
        <version>8.0.13</version>
    </dependency>
</dependencies>
</plugin>
```

24.Spring 整合 Mybatis 地 SQL 映射时有原始 dao 和 mapper 代理两种方式:

(1) 原始 dao: 需要写 dao 接口和 dao 实现类。

(2) mapper 代理: 需要编写代理接口, 但不需要接口实现。

➤ 两种方式都需要编写映射文件。

25.MyBatis 三剑客指的是: MyBatis-Generate、Mybatis Plus、MyBatis-PageHelper。

- Mybatis Generator : 使用这个 maven 插件来快速生成 Dao 类、mapper 配置文件和 Model 类。
- Mybatis Plus: 简称 MP, 是一个 Mybatis 的增强工具, 在 Mybatis 的基础上只做增强不做改变, 为简化开发、提高效率而生。这个插件有一些通用的 CRUD 接口, 可以简化 MyBatis 开发, 同时也提供在 xml 中编辑 SQL 语句时自动补全功能。
- MyBatis-PageHelper: 一个通用的分页插件。

26.使用 Mybatis Generator 有个步骤:

(1) 在 pom.xml 中添加依赖:

```
<plugin>
    <groupId>org.mybatis.generator</groupId>
    <artifactId>mybatis-generator-maven-plugin</artifactId>
    <version>1.3.7</version>
    <dependencies>
        <dependency>
```

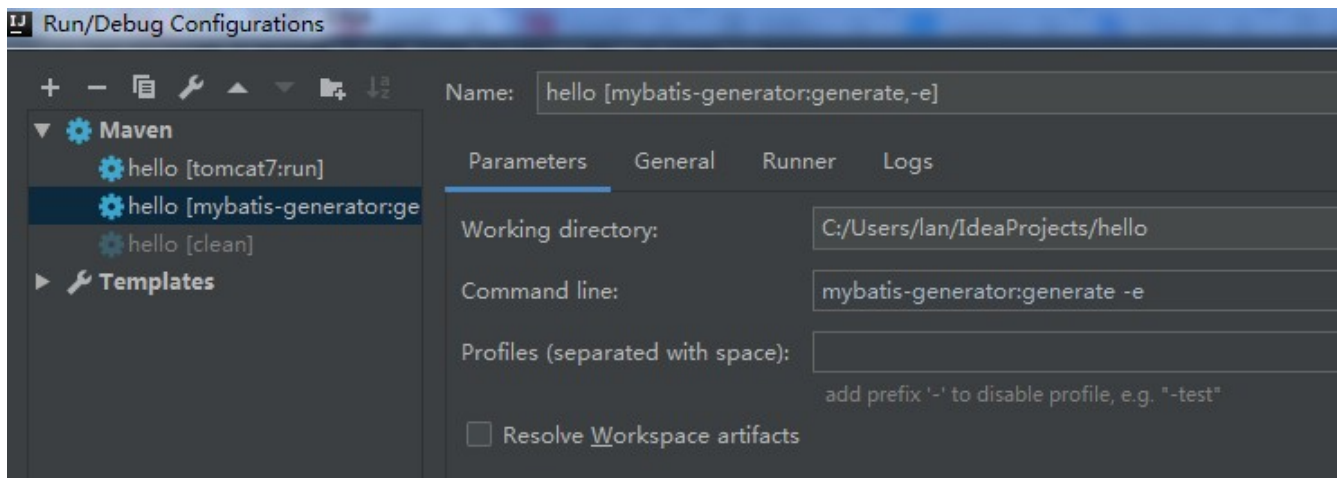
```

        <groupId>mysql</groupId>
        <artifactId>mysql-connector-java</artifactId>
        <version>8.0.13</version>
    </dependency>
</dependencies>
</plugin>

```

(2) 在 resources 目录下编写 generatorConfig.xml 文件，注意，该文件必须在 resources 目录下，不能在 resources 目录的子目录下。

(3) 在 Idea 中为 Mybatis Generator 添加配置项，Command line 中为 “mybatis-generator:generate -e”：



27. mybatis 正常使用要有 mybatis 和 mybatis-spring 两个依赖：

```

<dependency>
    <groupId>org.mybatis</groupId>
    <artifactId>mybatis-spring</artifactId>
    <version>1.3.2</version>
</dependency>
<dependency>
    <groupId>org.mybatis</groupId>
    <artifactId>mybatis</artifactId>
    <version>3.4.6</version>
</dependency>

```

28. SqlSessionFactory 不会自动生成, 需要通过@Autowired 自动填充等功能由用户生成。

```
@Controller
public class login {
    @Autowired
    SqlSessionFactory loginSqlSessionFactory;

    @RequestMapping("/index")
    public String index(){
        SqlSession session = loginSqlSessionFactory.openSession();
        try {
            AiaiPostsMapper selectpost = session.getMapper(AiaiPostsMapper.class);
            AiaiPostsWithBLOBs post = selectpost.selectByPrimaryKey(252L);
            System.out.println(post.getPostTitle());
            System.out.println(post.getPostContent());
        }finally {
            session.close();
        }
        return "index";
    }
}
```

29.

## Linux 系统

1. 软件管理工具 yum 有以下常用命令:

- yum install: 安装软件。
- yum remove: 卸载软件。

```
yum remove php
```

- yum list: 列出当前可用源有没有指定的可用软件。

```
yum list php*
```

- yum update: 升级软件包, 可以升级内核。
- yum upgrade: 升级软件包, 该命令无法升级内核。
- yum list: 显示所有已安装的软件信息。
- yum clean: 清除缓存。

2. apt-get 和 yum 一样都是软件包管理工具。有以下常用命令:

- apt-cache search: 搜索指定的软件包。
- apt-cache show: 显示指定的软件包信息。
- apt-get install: 安装指定的软件包。
- apt-get remove: 删除指定的软件包。
- apt-get update: 更新源。注意这个命令和 yum 的 update 不同, 这个并不更新软件, 只是更新软件的状态, 例如是否有最新版本等。
- apt-get upgrade: 更新已安装的包。
- apt-get dist-upgrade: 升级系统。
- apt-get clean: 清理无用的包。
- apt-get autoclean: 清理无用的包。

3. chkconfig 用于设置软件的启动级别:

- on: 设置为开机自启动。

```
#chkconfig apache on
```

- off: 关闭开机自启动:
- --list: 查看开机启动项。后不接软件名是查看所有自启动项, 接软件名时查看指定软件的启动项。

4. chmod 用于修改文件的读写可执行权限, 除了使用 “chmod 777 xxx” 这种形式外, 还可以使用下列形式:



```
chmod a+x,g+w exer1
```

- a: 表示对所有用户
- u: 表示系主用户
- g: 表示同组用户
- o: 表示其他用户

5. Linux 操作系统包括三种不同类型的进程，每种进程都有自己的特点和属性。

- 交互进程：由一个 shell 启动的进程。交互进程既可以在前台运行，也可以在后台运行。
- 批处理进程：这种进程和终端没有联系，是一个进程序列。
- 监控进程：也称守护进程，Linux 系统启动时启动的进程，并在后台运行。

6. linux 中的源类似 android 中的软件中心，ubuntu 中的软件源文件为/etc/apt/sources.list，其源列表如下：

```
deb http://mirrors.aliyun.com/ubuntu/ bionic-security main restricted
# deb-src http://security.ubuntu.com/ubuntu bionic-security main restricted
deb http://mirrors.aliyun.com/ubuntu/ bionic-security universe
# deb-src http://security.ubuntu.com/ubuntu bionic-security universe
deb http://mirrors.aliyun.com/ubuntu/ bionic-security multiverse
# deb-src http://security.ubuntu.com/ubuntu bionic-security multiverse
```

- deb: 代表包的类型，deb 表示二进制包，deb-src 表示源码包。
  - 第 2 部分的网址表示源的地址。
  - 第 3 部分的“bionic-security”为发行的版本。
  - 最后一部分是软件包分类，可能有多个值，例如第一个就有 main 和 restricted 两种分类。不同的系统支持不同的分类，例如 ubuntu 下的 main 代表官方支持的自由软件。
- 通常我们可以通过直接修改 source.list 文件的形式来修改源。

7. PPA，即个人软件包档案。是 Ubuntu Launchpad 网站提供的一项源服务，允许个人用户上传软件源代码，通过 Launchpad 进行编译并发布为二进制软件包，作为 apt 源供其他用户下载和更新。PPA 的一般形式是：

```
ppa:user/ppa-name
```

## 8. PPA 源命令:

- 使用 add-apt-repository 添加 PPA 源:

```
sudo add-apt-repository ppa:user/ppa-name
```

- 使用 sudo add-apt-repository -r 命令删除 PPA 源:

```
sudo add-apt-repository -r ppa:user/ppa-name
```

## 9. 使用 “ps -ef” 来查看指定的服务是否正在运行。

## 10. shadowsocks 最简单的安装方式是使用 pip 来安装，然后使用进程管理工具 supervisor 来管理 shadowsocks。

```
easy_install pip
pip install shadowsocks
pip install -U shadowsocks
easy_install supervisor
```

supervisor 是一个进程管理工具，用于管理后台进程。如果我们修改了 shadowsocks 的配置文件，例如修改了端口，那么我们需要使用 supervisor 来为 shadowsocks 重新加载配置文件，而不是使用 shadowsocks 本身的 ssserver 命令。

## 11. supervisor 常用命令:

- supervisorctl reload: 重新启动配置中的所有程序。
- supervisorctl start program\_name: 启动 program\_name 进程。
- supervisorctl: 查看正在守候的进程。
- supervisorctl stop program\_name: 停止 program\_name 进程。
- supervisorctl restart program\_name: 重启 program\_name 进程。
- supervisorctl stop all: 停止全部进程。

## 12. Supervisor 的配置文件是/etc/supervisor.conf，里面定义的是 supervisor 管理的服

务:

```
[program:ss]
```

```
command=/usr/bin/ssserver -c /etc/shadowsocks/shadowsocks.json
redirect_stderr=true
user=root
stdout_logfile=/tmp/ss.log
```

13.ubuntu 右下角的快捷方式所在的位置为 “/usr/share/applications/” 。

## 常用 Maven 命令

1. mvn archetype:create: 创建项目:

```
#mvn archetype:create -DgroupId=org.sonatype.mavenbook.ch -DartifactId=simple
-DDarchetypeArtifactId=maven-archetype-webapp
```

其中:

- -D: 指定参数。
- groupId: 指定组织 ID。
- artifactId: 指定项目 ID。
- DarchetypeArtifactId=maven-archetype-webapp: 表示创建 Web 项目。

2. mvn compile : 编译源代码。

3. mvn test-compile : 编译测试代码。

4. mvn test : 运行应用程序中的单元测试。

5. mvn site : 生成项目相关信息的网站。

6. mvn clean : 清除目标目录中的生成结果。

7. mvn package : 依据项目生成 jar 文件。

8. mvn dependency:tree: 显示 maven 依赖树。

9. mvn dependency:list: 显示 maven 依赖列表。

- 10.mvn tomcat:run: 启动 tomcat。
- 11.mvn tomcat:deploy: 运行打包部署。
- 12.mvn tomcat:undeploy: 撤销部署。
- 13.mvn tomcat:start: 启动 web 应用。
- 14.mvn tomcat:stop: 停止 web 应用。
- 15.mvn tomcat:redploy: 重新部署。

## Redis 开发与运维

### 第 1 章 初识 redis

1. Redis 是一种基于键值对的 NoSQL 数据库，与很多键值对数据库不同的是，Redis 中的值可以是由 string、hash、list、set、zset、Bitmaps、HyperLogLog、GEO 等多种数据结构和算法组成，因此 Redis 可以满足很多的应用场景。
  - Bitmaps 即位图，而 GEO 即地理信息定位。
2. 因为 Redis 会将所有数据都存放在内存中，所以它的读写性能非常惊人。
3. Redis 还可以将内存的数据利用快照和日志的形式保存到硬盘上，这样在发生类似断电或者机器故障的时候，内存中的数据不会“丢失”。
4. Redis 中的值不仅可以是字符串，而且还可以是具体的数据结构，这样不仅能便于在许多应用场景的开发，同时也能够提高开发效率。
5. Redis 还提供了许多额外的功能：
  - 提供了键过期功能，可以用来实现缓存。
  - 提供了发布订阅功能，可以用来实现消息系统。
  - 提供了简单的事务功能，能在一定程度上保证事务特性。
6. Redis 提供了复制功能，实现了多个相同数据的 Redis 副本，复制功能是分布式 Redis 的基础。

## 第 2 章 API 的理解和使用

1. 使用 `redis-server` 命令来启动 Redis。

```
$ redis-server
```

2. 使用 `redis-cli` 命令在命令行上连接 Redis 服务，该命令有以下常用参数：

- `-h`：指定远程服务器的主机名。
- `-p`：指定远程服务器的主机端口。
- `-a`：如果设置了密码，指定密码。

```
$redis-cli -h 127.0.0.1 -p 6379 -a "mypass"
```

如果是本地主机，可以省略参数：

```
$redis-cli
```

3.