Leader.us

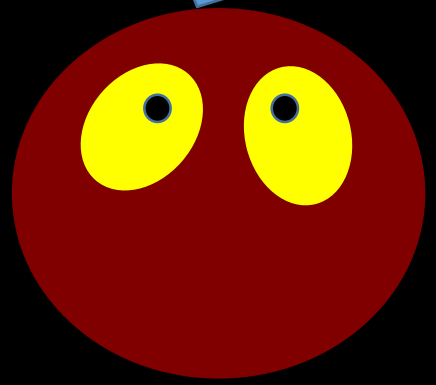
2016年版教程

SPRING+架构篇

SpringBoot篇

Expert Java Leader.us

第一章：快速上手



SpringBoot庞大家族

Expert Java Leader.us

http://mvnrepository.com/artifact/org.springframework.boot

Group: org.springframework.boot

1. **Spring Boot Test Starter** — 1,085 usages
org.springframework.boot » spring-boot-starter-test — Apache
Starter for testing Spring Boot applications with libraries including JUnit, Hamcrest and Mockito

2. **Spring Boot Web Starter** — 617 usages
org.springframework.boot » spring-boot-starter-web — Apache
Starter for building web, including RESTful, applications using Spring MVC. Uses Tomcat as the default embedded container

3. **Spring Boot Starter**
org.springframework.boot » spring-boot-starter
Core starter, including auto-configuration support, logging and YAML

4. **Spring Boot AutoConfigure**
org.springframework.boot » spring-boot-autoconfigure
Spring Boot AutoConfigure

5. **Spring Boot Configuration Processor**
org.springframework.boot » spring-boot-configuration-processor
Spring Boot Configuration Processor

6. **Spring Boot Actuator Starter**
org.springframework.boot » spring-boot-starter-actuator
Starter for using Spring Boot's Actuator which provides production ready features t

7. **Spring Boot**
org.springframework.boot » spring-boot

30. **Spring Boot AMQP Starter** — 15 usages
org.springframework.boot » spring-boot-starter-amqp — Apache
Starter for using Spring AMQP and Rabbit MQ

31. **Spring Boot Data REST Starter** — 15 usages
org.springframework.boot » spring-boot-starter-data-rest — Apache
Starter for exposing Spring Data repositories over REST using Spring Data REST

32. **Spring Boot FreeMarker Starter** — 15 usages
org.springframework.boot » spring-boot-starter-freemarker — Apache
Starter for building MVC web applications using FreeMarker views

33. **Spring Boot Batch Starter** — 14 usages
org.springframework.boot » spring-boot-starter-batch — Apache
Starter for using Spring Batch

34. **Spring Boot Jersey Starter** — 13 usages
org.springframework.boot » spring-boot-starter-jersey — Apache
Starter for building RESTful web applications using JAX-RS and Jersey. An alternative to spring-boot-starter-web

And More .....

SpringBoot模板工程1

Expert Java Leader.us

spring-boot-study

pom.xml

```
<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>leaderus.springboot</groupId>
    <artifactId>SpringBootStudy</artifactId>
    <version>0.1-SNAPSHOT</version>
    <properties>
        <spring.boot.version>1.4.2.RELEASE</spring.boot.version>
        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
    </properties>
    <parent>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-parent</artifact
        <version>1.4.2.RELEASE</version>
    </parent>
    <dependencies>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-web</artifac
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-test</artifactId>
            <scope>test</scope>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-jdbc</artifactId>
```

```
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-jdbc</artifactId>
```

The managed version is 1.4.2.RELEASE The artifact is managed in
org.springframework.boot:spring-boot-dependencies:1.4.2.RELEASE

Jump to location

SpringBoot模板工程2

Expert Java Leader.us

```
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-dependencies</artifactId>
```

```
<activemq.version>5.13.4</activemq.version>
<antlr2.version>2.7.7</antlr2.version>
<appengine.version>1.9.44</appengine.version>
<artemis.version>1.3.0</artemis.version>
<aspectj.version>1.8.9</aspectj.version>
<assertj.version>2.5.0</assertj.version>
<atomikos.version>3.9.3</atomikos.version>
<bitronix.version>2.1.4</bitronix.version>
<caffeine.version>2.3.4</caffeine.version>
<cassandra-driver.version>2.1.9</cassandra-driver.version>
<classmate.version>1.3.3</classmate.version>
<commons-beanutils.version>1.9.3</commons-beanutils.version>
<commons-collections.version>3.2.2</commons-collections.version>
<commons-codec.version>1.10</commons-codec.version>
<commons-dbcp.version>1.4</commons-dbcp.version>
<commons-dbcp2.version>2.1.1</commons-dbcp2.version>
<commons-digester.version>2.1</commons-digester.version>
<commons-pool.version>1.6</commons-pool.version>
<commons-pool2.version>2.4.2</commons-pool2.version>
<couchbase-client.version>2.2.8</couchbase-client.version>
<couchbase-cache-client.version>2.0.0</couchbase-cache-client.ve
<crashub.version>1.3.2</crashub.version>
<derby.version>10.12.1.1</derby.version>
<dom4j.version>1.6.1</dom4j.version>
<dropwizard-metrics.version>3.1.2</dropwizard-metrics.version>
<ehcache.version>2.10.3</ehcache.version>
<ehcache3.version>3.1.3</ehcache3.version>
<embedded-mongo.version>1.50.5</embedded-mongo.version>
<flyway.version>3.2.1</flyway.version>
<freemarker.version>2.3.25-incubating</freemarker.version>
<elasticsearch.version>2.4.1</elasticsearch.version>
<gemfire.version>8.2.0</gemfire.version>
<glassfish-el.version>3.0.0</glassfish-el.version>
```

```
<gson.version>2.7</gson.version>
<h2.version>1.4.193</h2.version>
<hamcrest.version>1.3</hamcrest.version>
<hazelcast.version>3.6.6</hazelcast.version>
<hibernate.version>5.0.11.Final</hibernate.version>
<hibernate-validator.version>5.2.4.Final</hibernate-validator.version>
<hikaricp.version>2.4.7</hikaricp.version>
<hikaricp-java6.version>2.3.13</hikaricp-java6.version>
<hornetq.version>2.4.7.Final</hornetq.version>
<hsqldb.version>2.3.3</hsqldb.version>
<htmlunit.version>2.21</htmlunit.version>
<httpasyncclient.version>4.1.2</httpasyncclient.version>
<httpclient.version>4.5.2</httpclient.version>
<httpcore.version>4.4.5</httpcore.version>
<infinispan.version>8.2.4.Final</infinispan.version>
<jackson.version>2.8.4</jackson.version>
<janino.version>2.7.8</janino.version>
<javassist.version>3.20.0-GA</javassist.version> <!-- Same as Hibernate -->
<javax-cache.version>1.0.0</javax-cache.version>
<javax-mail.version>1.5.6</javax-mail.version>
<javax-transaction.version>1.2</javax-transaction.version>
<jaxen.version>1.1.6</jaxen.version>
<jaybird.version>2.2.11</jaybird.version>
<jboss-logging.version>3.3.0.Final</jboss-logging.version>
<jboss-transaction-spi.version>7.3.4.Final</jboss-transaction-spi.version>
<jdom2.version>2.0.6</jdom2.version>
<jedis.version>2.8.2</jedis.version>
<jersey.version>2.23.2</jersey.version>
<jest.version>2.0.3</jest.version>
<jetty.version>9.3.14.v20161028</jetty.version>
<jetty-jsp.version>2.2.0.v201112011158</jetty-jsp.version>
<jetty-el.version>8.0.33</jetty-el.version>
<jms-api.version>1.1-rev-1</jms-api.version>
<jmustache.version>1.12</jmustache.version>
<jna.version>4.2.2</jna.version>
```

And More ……

SpringBoot模板工程3

```
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-dependencies</artifactId>
```

Expert Java Leader.us

And More ……

```
<hibernate.version>5.0.11.Final</hibernate.version>
<hibernate-validator.version>5.2.4.Final</hibernate-validator.version>
<dependency>

        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-data-jpa</artifactId>
        <version>1.4.2.RELEASE</version>
</dependency>
<dependency>

        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-jdbc</artifactId>
        <version>1.4.2.RELEASE</version>
        </dependency>
```

No Mybatis !!!!!

Why!!!!!

@SpringBootApplication注解分析1

Expert Java Leader.us

```
@Target(ElementType.TYPE)
@Retention(RetentionPolicy.RUNTIME)
@Documented
@Inherited
@SpringBootConfiguration
@EnableAutoConfiguration
@ComponentScan(excludeFilters = @Filter(type = FilterType.CUSTOM, classes = TypeExcludeFilter.class))
public @interface SpringBootApplication {
```

开启@componentScan功能

By default,@SpringBootApplication enables component scanning from the package where it‘s declared

@SpringBootApplication(scanBasePackages="edu.ldcollege.")**

org.springframework.context.annotation.FilterType
用法： @Filter(type = FilterType.CUSTOM, classes = TypeExcludeFilter.class)

A component scan filter narrows down which of those classes to instantiate beans for.

- You might only want to consider classes which have a certain annotation, e.g. `@Component`, and you'd use an **annotation** filter for that.
- You might want to consider classes that implement a certain interface, e.g. `Dao`, and you'd use **assignable** for that.
- You might want to pick out some classes by name, e.g. `com.foo.**.service.*`, and you'd use a **regex** for that.
- You might want to use expressions to pick out a complex subset of classes, e.g. `com.foo..service.* && !com.foo..MockService`, and you'd use **aspectj** for that.
- You might extremely rarely want to pick out classes by their metadata, e.g. create a bean if the class has an enclosing class named `Foo`, and you'd write a **custom** `TypeFilter` to do that, which gives you access to that metadata.

```
public interface TypeFilter {

    /**
     * Determine whether this filter matches for the class described by
     * the given metadata.
     * @param metadataReader the metadata reader for the target class
     * @param metadataReaderFactory a factory for obtaining metadata readers
     * for other classes (such as superclasses and interfaces)
     * @return whether this filter matches
     * @throws IOException in case of I/O failure when reading metadata
     */
    boolean match(MetadataReader metadataReader, MetadataReaderFactory metadataReaderFactory)
            throws IOException;

}
```

@SpringBootApplication注解分析2

@SpringBootConfiguration

使得自身成为@Configuration配置，里面可以定义Bean

Indicates that a class provides Spring Boot application @Configuration. Can be used as an alternative to the Spring's standard @Configuration annotation so that configuration can be found automatically (for example in tests). Application should only ever include one @SpringBootConfiguration and most idiomatic Spring Boot applications will inherit it from @SpringBootApplication.

```
@Bean
@ConfigurationProperties(prefix = "spring.datasource")
public DataSource primaryDataSource() {
    System.out.println("------------------- DataSource init --------------------");
    return DataSourceBuilder.create().build();
}

@Bean
PlatformTransactionManager transactionManager(DataSource dataSource) {
    DataSourceTransactionManager transactionManager = new DataSourceTransactionManager();
    transactionManager.setDataSource(dataSource);
    return transactionManager;
}
```

所以，可以在有此注解的类上，定义@Bean了

@SpringBootApplication注解分析3

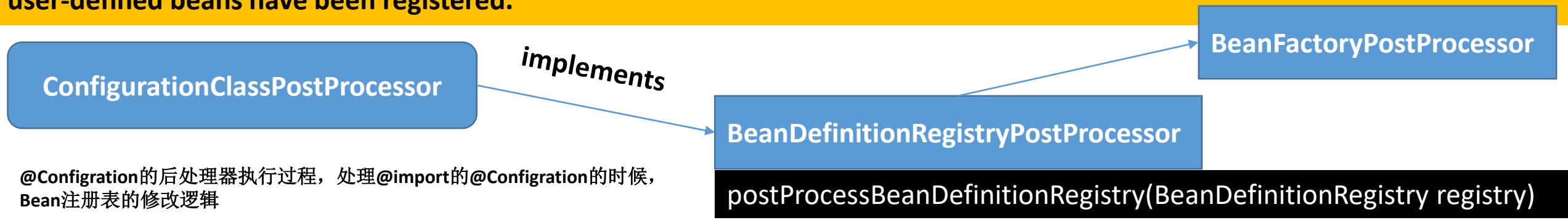
Expert Java Leader.us

@EnableAutoConfiguration

最智能的一个标签：这个注解告诉Spring Boot根据添加的jar依赖猜测你想如何配置Spring

For example, If you have tomcat-embedded.jar on your classpath you are likely to want a TomcatEmbeddedServletContainerFactory (unless you have defined your own EmbeddedServletContainerFactory bean).

Auto-configuration tries to be as intelligent as possible and will back-away as you define more of your own configuration. You can always manually exclude() any configuration that you never want to apply (use excludeName() if you don't have access to them). You can also exclude them via the spring.autoconfigure.exclude property. Auto-configuration is always applied after user-defined beans have been registered.



@Configration的后处理器执行过程，处理@import的@Configration的时候，Bean注册表的修改逻辑

EnableAutoConfigurationImportSelector.selectImports(AnnotationMetadata) line: 86
ConfigurationClassParser.processDeferredImportSelectors() line: 474
ConfigurationClassParser.parse(Set<BeanDefinitionHolder>) line: 184
ConfigurationClassPostProcessor.processConfigBeanDefinitions(BeanDefinitionRegistry) line: 324
ConfigurationClassPostProcessor.postProcessBeanDefinitionRegistry(BeanDefinitionRegistry) line: 246
PostProcessorRegistrationDelegate.invokeBeanDefinitionRegistryPostProcessors(Collection<BeanDefinitionRegistryPostProcessor>, BeanDefinitionRegistry) line: 270
PostProcessorRegistrationDelegate.invokeBeanFactoryPostProcessors(ConfigurableListableBeanFactory, List<BeanFactoryPostProcessor>) line: 93

@SpringBootApplication注解分析4

Expert Java Leader.us

@EnableAutoConfiguration

@AutoConfigurationPackage
@Import(EnableAutoConfigurationImportSelector.class)
public @interface EnableAutoConfiguration {

String[] selectImports(AnnotationMetadata importingClassMetadata);

EnableAutoConfigurationImportSelector.java

```
public String[] selectImports(AnnotationMetadata metadata) {
    if (!isEnabled(metadata)) {
        return NO_IMPORTS;
    }
    try {
        AnnotationAttributes attributes = getAttributes(metadata);
        List<String> configurations = getCandidateConfigurations(metadata,
                attri
        configuration
        Set<String> e
        configuration
        configuration
```

configurations= ArrayList<E> (id=40)
elementData= Object[138] (id=51)
[0...99]
[0]= "org.springframework.boot.autoconfigure.admin.SpringApplicationAdminJmxAutoConfiguration" (id=54)
[1]= "org.springframework.boot.autoconfigure.aop.AopAutoConfiguration" (id=58)
[2]= "org.springframework.boot.autoconfigure.amqp.RabbitAutoConfiguration" (id=59)
[3]= "org.springframework.boot.autoconfigure.MessageSourceAutoConfiguration" (id=60)

发现了：org.springframework.boot.autoconfigure.web.WebMvcAutoConfiguration

发现@Configration注解类的实际工作是SPRING核心完成的：

org.springframework.core.io.support.SpringFactoriesLoader

@SpringBootApplication注解分析5

Expert Java Leader.us

org.springframework.core.io.support.SpringFactoriesLoader

SpringFactoriesLoader loads and instantiates factories of a given type from "META-INF/spring.factories" files which may be present in multiple JAR files in the classpath. The spring.factories file must be in Properties format, where the key is the fully qualified name of the interface or abstract class, and the value is a comma-separated list of implementation class names. For example:
example.MyService=example.MyServiceImpl1,example.MyServiceImpl2
where example.MyService is the name of the interface, and MyServiceImpl1 and MyServiceImpl2 are two implementations.

```
public static List<String> loadFactoryNames(Class<?> factoryClass, ClassLoader classLoader) {
    String factoryClassName = factoryClass.getName();
    try {
        Enumeration<URL> urls = (classLoader != null ? classLoader.getResources(FACTORIES_RESOURCE_LOCATION) :
                ClassLoader.getSystemResources(FACTORIES_RESOURCE_LO
        List<String> result = new ArrayList<String>();
        while (urls.hasMoreElements()) {
            URL url = urls.nextElement();
            Properties properties = PropertiesLoaderUtils.loadProper
            String factoryClassNames = properties.getProperty(factor
            result.addAll(Arrays.asList(StringUtils.commaDelimitedLi
        }
        return result;
    }
    catch (IOException ex) {
        throw new IllegalArgumentException("Unable to load [" + factoryClass.getName() +
                "] factories from location [" + FACTORIES_RESOURCE_LOCATION + "]", ex);
    }
}
```

String
org.springframework.core.io.support.SpringFactoriesLoader.FACTOR
ATION : "META-INF/spring.factories"

The location to look for factories.

Can be present in multiple JAR files.

@SpringBootApplication注解分析6

Expert Java Leader.us

@EnableAutoConfiguration

spring-boot-autoconfigure-1.4.2.RELEASE.jar

- configurations= ArrayList<E> (id=40)
  - elementData= Object[138] (id=51)
    - [0...99]
      - [0]= "org.springframework.boot.autoconfigure.admin.SpringApplicationAdminJmxAutoConfiguration" (id=54)
      - [1]= "org.springframework.boot.autoconfigure.aop.AopAutoConfiguration" (id=58)
      - [2]= "org.springframework.boot.autoconfigure.amqp.RabbitAutoConfiguration" (id=59)
      - [3]= "org.springframework.boot.autoconfigure.MessageSourceAutoConfiguration" (id=60)
      - [4]= "org.springframework.boot.autoconfigure.PropertyPlaceholderAutoConfiguration" (id=61)
      - [5]= "org.springframework.boot.autoconfigure.batch.BatchAutoConfiguration" (id=62)
      - [6]= "org.springframework.boot.autoconfigure.cache.CacheAutoConfiguration" (id=63)
      - [7]= "org.springframework.boot.autoconfigure.cassandra.CassandraAutoConfiguration" (id=64)
      - [8]= "org.springframework.boot.autoconfigure.cloud.CloudAutoConfiguration" (id=65)
      - [9]= "org.springframework.boot.autoconfigure.context.ConfigurationPropertiesAutoConfiguration" (id=66)
      - [10]= "org.springframework.boot.autoconfigure.couchbase.CouchbaseAutoConfiguration" (id=67)
      - [11]= "org.springframework.boot.autoconfigure.dao.PersistenceExceptionTranslationAutoConfiguration" (id=68)
      - [12]= "org.springframework.boot.autoconfigure.data.cassandra.CassandraDataAutoConfiguration" (id=69)
      - [13]= "org.springframework.boot.autoconfigure.data.cassandra.CassandraRepositoriesAutoConfiguration" (id=70)
      - [14]= "org.springframework.boot.autoconfigure.data.couchbase.CouchbaseDataAutoConfiguration" (id=71)
      - [15]= "org.springframework.boot.autoconfigure.data.couchbase.CouchbaseRepositoriesAutoConfiguration" (id=167)
      - [16]= "org.springframework.boot.autoconfigure.data.elasticsearch.ElasticsearchAutoConfiguration" (id=168)
      - [17]= "org.springframework.boot.autoconfigure.data.elasticsearch.ElasticsearchDataAutoConfiguration" (id=217)
      - [18]= "org.springframework.boot.autoconfigure.data.elasticsearch.ElasticsearchRepositoriesAutoConfiguration" (id=218)
      - [19]= "org.springframework.boot.autoconfigure.data.jpa.JpaRepositoriesAutoConfiguration" (id=219)
      - [20]= "org.springframework.boot.autoconfigure.data.mongo.MongoDataAutoConfiguration" (id=220)
      - [21]= "org.springframework.boot.autoconfigure.data.mongo.MongoRepositoriesAutoConfiguration" (id=221)
      - [22]= "org.springframework.boot.autoconfigure.data.neo4j.Neo4jDataAutoConfiguration" (id=222)
      - [23]= "org.springframework.boot.autoconfigure.data.neo4j.Neo4jRepositoriesAutoConfiguration" (id=223)
      - [24]= "org.springframework.boot.autoconfigure.data.solr.SolrRepositoriesAutoConfiguration" (id=224)

100个@Configuration 现成可用！！！目测还在极速膨胀，感谢Spring Boot，可少加班了

@SpringBootApplication注解分析7

Expert Java Leader.us

spring-boot-autoconfigure-1.4.2.RELEASE.jar

```
# Initializers
org.springframework.context.ApplicationContextInitializer=\
org.springframework.boot.autoconfigure.SharedMetadataReaderFactoryContextInitializer,\
org.springframework.boot.autoconfigure.logging.AutoConfigurationReportLoggingInitializer

# Application Listeners
org.springframework.context.ApplicationListener=\
org.springframework.boot.autoconfigure.BackgroundPreinitializer

# Auto Configure
org.springframework.boot.autoconfigure.EnableAutoConfiguration=\
org.springframework.boot.autoconfigure.admin.SpringApplicationAdminJmxAutoConfiguration,\
org.springframework.boot.autoconfigure.aop.AopAutoConfiguration,\
org.springframework.boot.autoconfigure.amqp.RabbitAutoConfiguration,\
org.springframework.boot.autoconfigure.MessageSourceAutoConfiguration,\
org.springframework.boot.autoconfigure.PropertyPlaceholderAutoConfiguration,\
org.springframework.boot.autoconfigure.batch.BatchAutoConfiguration,\
org.springframework.boot.autoconfigure.cache.CacheAutoConfiguration,\
org.springframework.boot.autoconfigure.cassandra.CassandraAutoConfiguration,\
org.springframework.boot.autoconfigure.cloud.CloudAutoConfiguration,\
org.springframework.boot.autoconfigure.context.ConfigurationPropertiesAutoConfiguration,\
org.springframework.boot.autoconfigure.couchbase.CouchbaseAutoConfiguration,\
org.springframework.boot.autoconfigure.dao.PersistenceExceptionTranslationAutoConfiguration,\
org.springframework.boot.autoconfigure.data.cassandra.CassandraDataAutoConfiguration,\
org.springframework.boot.autoconfigure.data.cassandra.CassandraRepositoriesAutoConfiguration,\
org.springframework.boot.autoconfigure.data.couchbase.CouchbaseDataAutoConfiguration,\
org.springframework.boot.autoconfigure.data.couchbase.CouchbaseRepositoriesAutoConfiguration,\
org.springframework.boot.autoconfigure.data.elasticsearch.ElasticsearchAutoConfiguration,\
org.springframework.boot.autoconfigure.data.elasticsearch.ElasticsearchDataAutoConfiguration,\
org.springframework.boot.autoconfigure.data.elasticsearch.ElasticsearchRepositoriesAutoConfiguration,\
org.springframework.boot.autoconfigure.data.jpa.JpaRepositoriesAutoConfiguration,\
org.springframework.boot.autoconfigure.data.mongo.MongoDataAutoConfiguration,\
org.springframework.boot.autoconfigure.data.mongo.MongoRepositoriesAutoConfiguration,\
org.springframework.boot.autoconfigure.data.neo4j.Neo4jDataAutoConfiguration,\
org.springframework.boot.autoconfigure.data.neo4j.Neo4jRepositoriesAutoConfiguration,\
org.springframework.boot.autoconfigure.data.solr.SolrRepositoriesAutoConfiguration,\
org.springframework.boot.autoconfigure.data.redis.RedisAutoConfiguration,\
org.springframework.boot.autoconfigure.data.redis.RedisRepositoriesAutoConfiguration,\
org.springframework.boot.autoconfigure.data.rest.RepositoryRestMvcAutoConfiguration,\
org.springframework.boot.autoconfigure.data.web.SpringDataWebAutoConfiguration,\
org.springframework.boot.autoconfigure.elasticsearch.jest.JestAutoConfiguration,\
org.springframework.boot.autoconfigure.freemarker.FreeMarkerAutoConfiguration,\
org.springframework.boot.autoconfigure.gson.GsonAutoConfiguration,\
org.springframework.boot.autoconfigure.h2.H2ConsoleAutoConfiguration,\
org.springframework.boot.autoconfigure.web.DispatcherServletAutoConfiguration,\
org.springframework.boot.autoconfigure.web.EmbeddedServletContainerAutoConfiguration,\
org.springframework.boot.autoconfigure.web.ErrorMvcAutoConfiguration,\
org.springframework.boot.autoconfigure.web.HttpEncodingAutoConfiguration,\
org.springframework.boot.autoconfigure.web.HttpMessageConvertersAutoConfiguration,\
org.springframework.boot.autoconfigure.web.MultipartAutoConfiguration,\
org.springframework.boot.autoconfigure.web.ServerPropertiesAutoConfiguration,\
org.springframework.boot.autoconfigure.web.WebClientAutoConfiguration,\
org.springframework.boot.autoconfigure.web.WebMvcAutoConfiguration,\
org.springframework.boot.autoconfigure.websocket.WebSocketAutoConfiguration,\
org.springframework.boot.autoconfigure.websocket.WebSocketMessagingAutoConfiguration,\
org.springframework.boot.autoconfigure.webservices.WebServicesAutoConfiguration

# Failure analyzers
org.springframework.boot.diagnostics.FailureAnalyzer=\
org.springframework.boot.autoconfigure.diagnostics.analyzer.NoSuchBeanDefinitionFailureAnalyzer,\
org.springframework.boot.autoconfigure.jdbc.DataSourceBeanCreationFailureAnalyzer,\
org.springframework.boot.autoconfigure.jdbc.HikariDriverConfigurationFailureAnalyzer

# Template availability providers
org.springframework.boot.autoconfigure.template.TemplateAvailabilityProvider=\
org.springframework.boot.autoconfigure.freemarker.FreeMarkerTemplateAvailabilityProvider,\
org.springframework.boot.autoconfigure.mustache.MustacheTemplateAvailabilityProvider,\
org.springframework.boot.autoconfigure.groovy.template.GroovyTemplateAvailabilityProvider,\
org.springframework.boot.autoconfigure.thymeleaf.ThymeleafTemplateAvailabilityProvider,\
org.springframework.boot.autoconfigure.velocity.VelocityTemplateAvailabilityProvider,\
org.springframework.boot.autoconfigure.web.JspTemplateAvailabilityProvider
```

现在可以明白为什么可执行的Spring Web

For example, If you have tomcat-embedded.jar on your classpath you are likely to want a TomcatEmbeddedServletContainerFactory (unless you have defined your own EmbeddedServletContainerFactory bean).

- spring-boot-starter-tomcat-1.4.2.RELEASE.jar
- tomcat-embed-core-8.5.6.jar
- tomcat-embed-el-8.5.6.jar
- tomcat-embed-websocket-8.5.6.jar

Spring Boot Dependencies

```
<dependency>
    <groupId>org.apache.tomcat.embed</groupId>
    <artifactId>tomcat-embed-core</artifactId>
    <version>${tomcat.version}</version>
</dependency>
<dependency>
    <groupId>org.apache.tomcat.embed</groupId>
    <artifactId>tomcat-embed-el</artifactId>
    <version>${tomcat.version}</version>
</dependency>
<dependency>
    <groupId>org.apache.tomcat.embed</groupId>
    <artifactId>tomcat-embed-jasper</artifactId>
    <version>${tomcat.version}</version>
</dependency>
<dependency>
    <groupId>org.apache.tomcat.embed</groupId>
    <artifactId>tomcat-embed-websocket</artifactId>
    <version>${tomcat.version}</version>
</dependency>
```

EmbeddedServletContainerAutoConfiguration

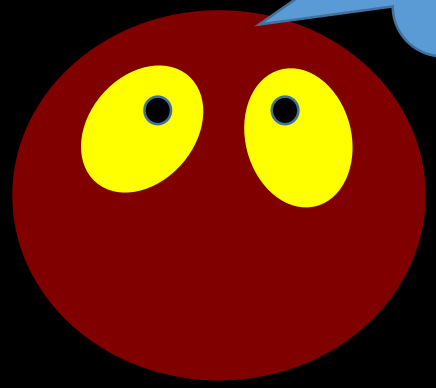
```
@AutoConfigureOrder(Ordered.HIGHEST_PRECEDENCE)
@Configuration
@ConditionalOnWebApplication
@Import(BeanPostProcessorsRegistrar.class)
public class EmbeddedServletContainerAutoConfiguration {

    /**
     * Nested configuration if Tomcat is being used.
     */
    @Configuration
    @ConditionalOnClass({ Servlet.class, Tomcat.class })
    @ConditionalOnMissingBean(value = EmbeddedServletContainerFactory.class, search = SearchStr
    public static class EmbeddedTomcat {

        @Bean
        public TomcatEmbeddedServletContainerFactory tomcatEmbeddedServletContainerFactory() {
            return new TomcatEmbeddedServletContainerFactory();
        }

    }
```

如果当前Spring环境是Web Application，并且没有定义Tomcat工厂类，而且存在Tomcat.class以及Servlet.class，就自动创建

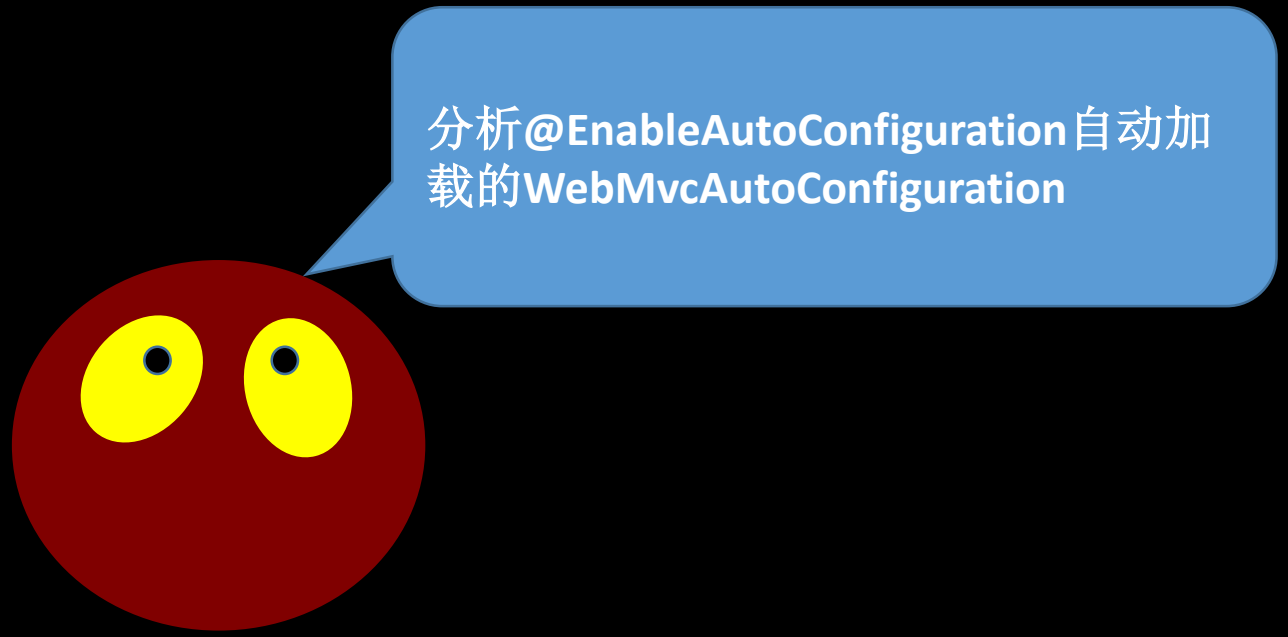


- classes
- lib
- lib-provided
  - spring-boot-starter-tomcat-1.4.2.RELEASE
  - tomcat-embed-core-8.5.6.jar
  - tomcat-embed-el-8.5.6.jar
  - tomcat-embed-websocket-8.5.6.jar

Expert Java Leader.us

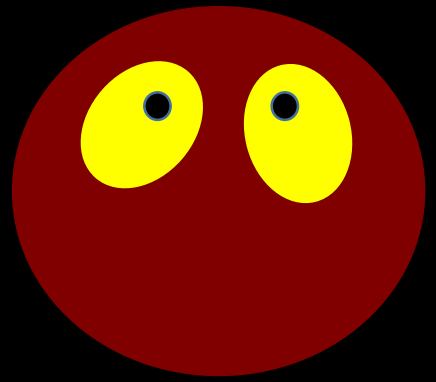
Spring boot & Spring MVC的秘密

`org.springframework.boot.autoconfigure.web.WebMvcAutoConfiguration`



Expert Java Leader.us

Springboot扫描到了WebMvcAutoConfiguration ？



Spring Boot Web Starter is a Starter for building web, including RESTful, applications using Spring MVC. Uses Tomcat as the default embedded container

spring-boot-starter-web.pom

```
<dependencies>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter</artifactId>
    </dependency>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-tomcat</artifactId>
    </dependency>
    <dependency>
        <groupId>org.hibernate</groupId>
        <artifactId>hibernate-validator</artifactId>
    </dependency>
    <dependency>
        <groupId>com.fasterxml.jackson.core</groupId>
        <artifactId>jackson-databind</artifactId>
    </dependency>
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-web</artifactId>
    </dependency>
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-webmvc</artifactId>
    </dependency>
</dependencies>
```

依赖了spring-webmvc的JAR包

spring-webmvc-4.3.4.RELEASE.jar

因为@EnableAutoConfiguration

Expert Java Leader.us

所以一切变得简单

一个Spring MVC+MySQL数据库+Mybatis的典型Web框架的配置，已经简单的不成样子了！！！

- Spring Boot Starter Web
- mybatis-spring-boot-starter
- mysql-connector-java

main/java/leader/App.java

```
@SpringBootApplication(scanBasePackages = "leader.**")
@MapperScan("leader.mapping")
public class App extends SpringBootServletInitializer {

    @Override
    protected SpringApplicationBuilder configure(SpringApplicationBuilder applicat
        return application.sources(App.class);
    }

    public static void main(String[] args) throws Exception {

        ConfigurableApplicationContext ctx = SpringApplication.run(App.class, args

    }
```

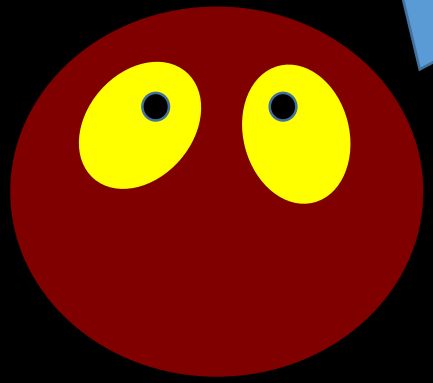
main/resources/application.properties

```
application.hellowmsg:  Leader Spring Boot
logging.level.=INFO
#logging.level.org.springframework.web.servlet.DispatcherServlet=DEBUG
spring.datasource.url=jdbc:mysql://localhost:3306/leaderspring?useUnicod
spring.datasource.username=root
spring.datasource.password=123456

spring.datasource.driverClassName=com.mysql.jdbc.Driver
spring.datasource.max-idle=20
spring.datasource.min-idle=5
spring.datasource.initial-size=10
spring.datasource.test-on-borrow=false
spring.datasource.test-on-return=false
spring.datasource.test-while-idle=true
spring.datasource.max-wait-millis=30000
spring.datasource.validation-query=SELECT 1
spring.datasource.time-between-eviction-runs-millis=180000
spring.datasource.min-evictable-idle-time-millis=600000
server.port=8080
```

Expert Java Leader.us

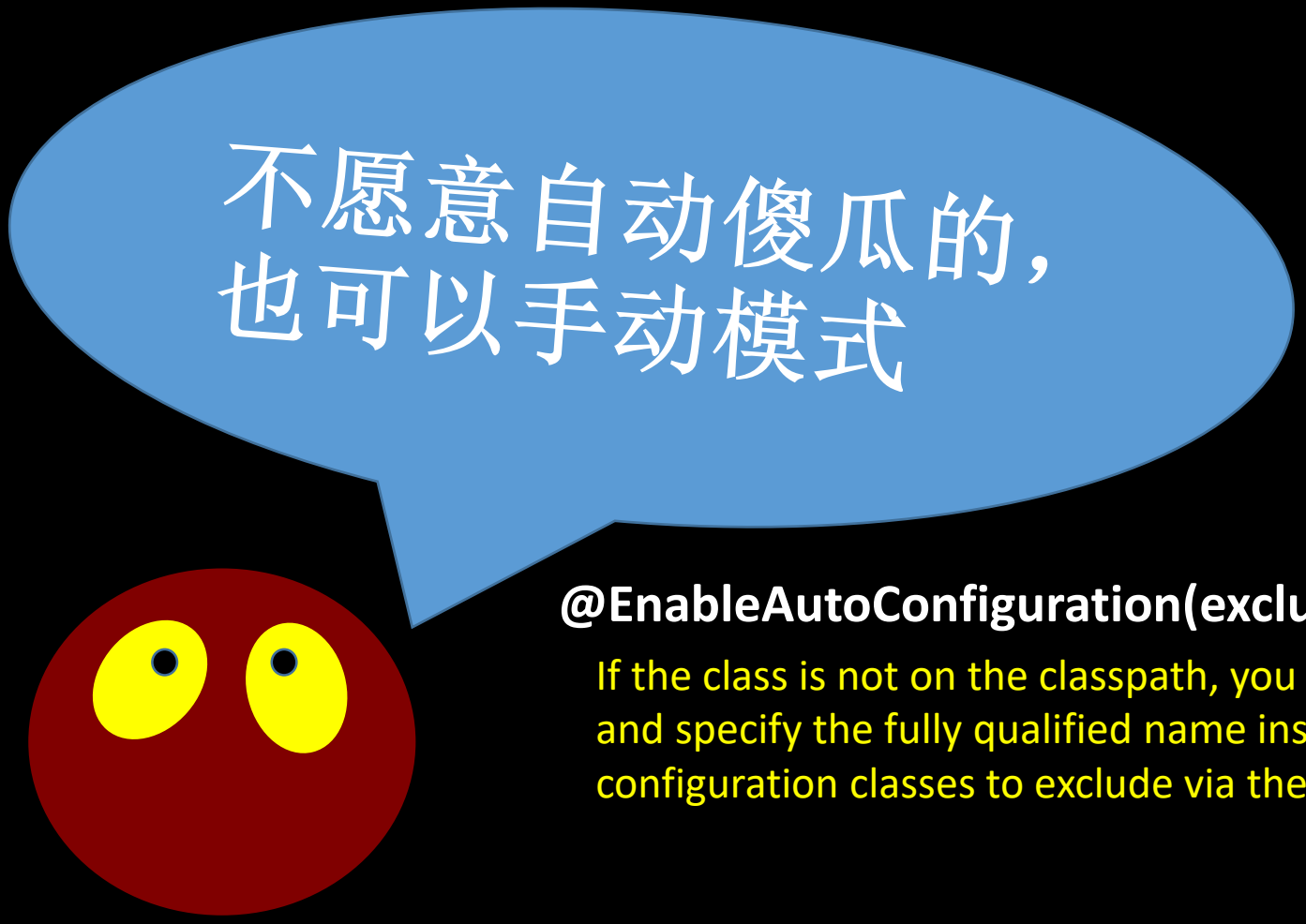
Spring Boot自动注册了多少个Bean



```
ConfigurableApplicationContext ctx = SpringApplication.run(App.class, args);
        Iterator<String> itors = ctx.getBeanFactory().getBeanNamesIterator();
        while (itors.hasNext()) {
                System.out.println(itors.next());
        }
```

作业题，整理并分类：每个Bean的名字，class，以及作用.....

Expert Java Leader.us



@EnableAutoConfiguration(exclude={DataSourceAutoConfiguration.class})

If the class is not on the classpath, you can use the excludeName attribute of the annotation and specify the fully qualified name instead. Finally, you can also control the list of auto-configuration classes to exclude via the spring.autoconfigure.exclude property.

But自动化是大势所趋.....

To Be Continued