

Springboot2系列[5]--常见注解及实现原理简单理解



安安熊

计算机

1 人赞同了该文章

在常见的配置中，我们这里主要选择下面三个进行说明

@SpringBootApplication

@Configuration

@EnableAutoConfiguration

@ComponentScan

一、@SpringBootApplication

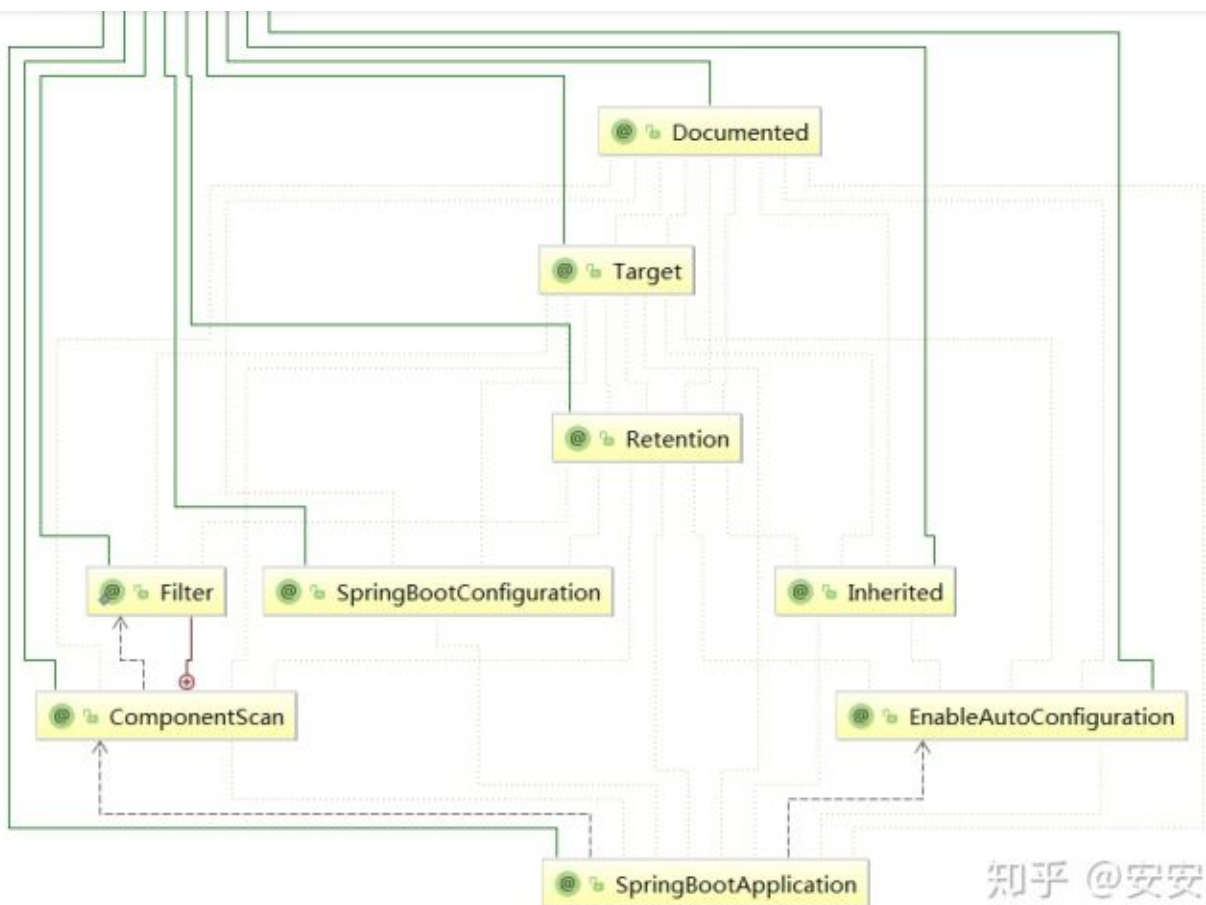
这个注解为springboot项目的启动注解类，用在springboot项目main方法中，为入口类的注解。

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

知乎 @安安熊

可以看到这个注解引入的主要注解有：

SpringBootConfiguration, EnableAutoConfiguration、ComponentScan 主要依赖关系如下：



1) SpringBootApplication

注解如下： 引入的为Configuration注解，指定需要初始化的bean，可参考：

Springboot2系列[4]-- Bean的注入

```
/*
 *
 */
@Target(ElementType.TYPE)
@Retention(RetentionPolicy.RUNTIME)
@Documented
@Configuration
public @interface SpringBootApplication {

}
```

知乎 @安安熊

2) EnableAutoConfiguration

赞同 1

添加评论

分享

收藏

...

```
@Target(ElementType.TYPE)
@Retention(RetentionPolicy.RUNTIME)
@Documented
@Inherited
@AutoConfigurationPackage
@Import(AutoConfigurationImportSelector.class)
public @interface EnableAutoConfiguration {

    String ENABLED_OVERRIDE_PROPERTY = "spring.boot.enableautoconfiguration";

    /**
     * Exclude specific auto-configuration classes such that they will never be applied.
     * @return the classes to exclude
     */
    Class<?>[] exclude() default {};

    /**
     * Exclude specific auto-configuration class names such that they will never be
     * applied.
     * @return the class names to exclude
     * @since 1.3.0
     */
    String[] excludeName() default {};
}
```

知乎 @安安熊

AutoConfigurationPackage :

这个注解继承的为: @Import(AutoConfigurationPackages.Registrar.class)

类包数据的加载

Import :

这个注解为: @Import(AutoConfigurationImportSelector.class)

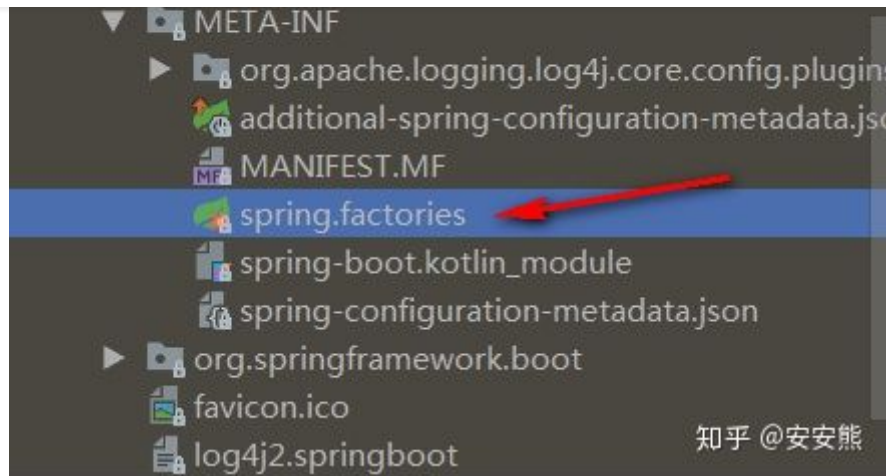
这个类核心部分为将所有符合条件的@Configuration配置都加载到当前SpringBoot创建并使用的IoC容器. 大致流程为, 找到jar包下面META-INF/spring.factories 实例化符合条件的bean,

知乎



首发于

springboot快速入门



知乎 @安安熊

文件内容：

```
org.springframework.boot.SpringApplicationRunListener=\
org.springframework.boot.context.event.EventPublishingRunListener

# Error Reporters
org.springframework.boot.SpringBootExceptionHandler=\
org.springframework.boot.diagnostics.FailureAnalyzers

# Application Context Initializers
org.springframework.context.ApplicationContextInitializer=\
org.springframework.boot.context.ConfigurationWarningsApplicationContextInitializer,\
org.springframework.boot.context.ContextIdApplicationContextInitializer,\
org.springframework.boot.context.config.DelegatingApplicationContextInitializer,\
org.springframework.boot.web.context.ServerPortInfoApplicationContextInitializer

# Application Listeners
org.springframework.context.ApplicationListener=\
org.springframework.boot.ClearCachesApplicationListener,\
org.springframework.boot.builder.ParentContextCloserApplicationListener,\
org.springframework.boot.context.FileEncodingApplicationListener,\
org.springframework.boot.context.config.AnsiOutputApplicationListener,\
org.springframework.boot.context.config.ConfigFileApplicationListener,\
org.springframework.boot.context.config.DelegatingApplicationListener,\
org.springframework.boot.context.logging.ClasspathLoggingApplicationListener,\
org.springframework.boot.context.logging.LoggingApplicationListener,\
org.springframework.boot.liquibase.LiquibaseServiceLocatorApplicationListener

# Environment Post Processors
```

知乎 @安安熊

赞同 1

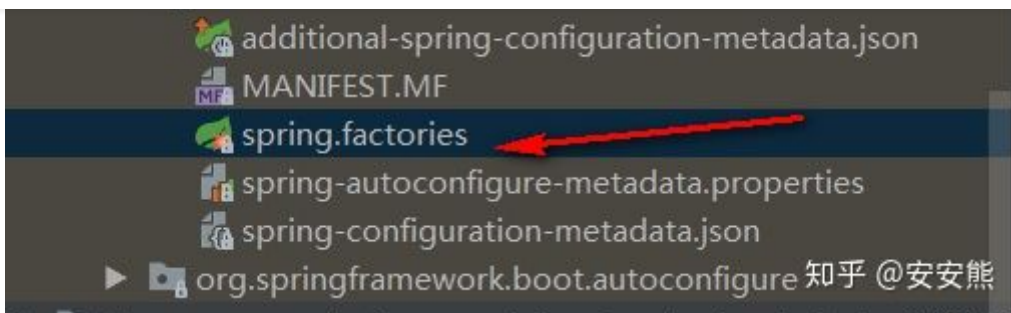


添加评论

分享

★ 收藏





可以看到autoconfigure的包，spring.factories 内容如下：

```
# Auto Configuration Import Listeners
org.springframework.boot.autoconfigure.AutoConfigurationImportListener=\
org.springframework.boot.autoconfigure.condition.ConditionEvaluationReportAutoConfigurationImportListener

# Auto Configuration Import Filters
org.springframework.boot.autoconfigure.AutoConfigurationImportFilter=\
org.springframework.boot.autoconfigure.condition.OnClassCondition

# 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.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.context.MessageSourceAutoConfiguration,\
org.springframework.boot.autoconfigure.context.PropertyPlaceholderAutoConfiguration,\
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.CassandraReactiveDataAutoConfiguration,\
org.springframework.boot.autoconfigure.data.cassandra.CassandraReactiveRepositoriesAutoConfiguration,\
org.springframework.boot.autoconfigure.data.cassandra.CassandraRepositoriesAutoConfiguration,\
org.springframework.boot.autoconfigure.data.couchbase.CouchbaseDataAutoConfiguration,\
```

其中配置的key为注解名称，值为对应的configuration配置类，如：
WebMvcAutoConfiguration

知乎



首发于

springboot快速入门

```
@ConditionalOnMissingBean(WebMvcConfigurationSupport.class)
@AutoConfigureOrder(Ordered.HIGHEST_PRECEDENCE + 10)
@AutoConfigureAfter({ DispatcherServletAutoConfiguration.class,
    validationAutoConfiguration.class })
public class WebMvcAutoConfiguration {

    public static final String DEFAULT_PREFIX = "";

    public static final String DEFAULT_SUFFIX = "";
```

知乎 @安安熊

这样借助于 @EnableAutoConfiguration 和 SpringFactoriesLoader 可以将我们需要的Bean初始化，并加载到IOC容器中，从而完成上下文的初始化工作。

二、@Configuration

Springboot2系列[4]-- Bean的注入

三、@ComponentScan

@ComponentScan的功能其实就是自动扫描并加载符合条件的组件（比如@Component和@Repository等）或者bean定义，最终将这些bean定义加载到IoC容器中。

我们可以通过basePackages等属性来细粒度的定制@ComponentScan自动扫描的范围，如果不指定，则默认Spring框架实现会从声明@ComponentScan所在类的package进行扫描。

我们可以自定义扫描路径：

1) 指定包名进行扫描

```
@ComponentScan(basePackages = {"org.sp.test"})
```

2) 可以自定义过滤，过滤掉不需要加载的类

怎么做？分两步

步骤一：自定义过滤器类

▲ 赞同 1 ▼

● 添加评论

➦ 分享

★ 收藏

...

知乎



首发于

springboot快速入门

```
* @author:xiongyongjie
* @date:2018年3月7日
* @description:
*/
public class MyTypeFilter implements TypeFilter{

    @Override
    public boolean match(MetadataReader metadataReader, MetadataReaderFactory metadataReaderFactory)
        throws IOException {
        if(metadataReader.getResource().getFilename().contentEquals("ConditionConfigBean.class"))
            return true;
        }
        return false;
    }
}
```

步骤二：配置过滤器到 ComponentScan 注解中

```
@ComponentScan(excludeFilters={@Filter(type=FilterType.CUSTOM,value=MyTypeFilter.class)}
//@EnableAspectJAutoProxy
public class SpringbootaopApplication { ... }
```

上述例子过滤掉了类名为：ConditionConfigBean 的类，此类不会被初始化。

四、重点

- 常见注解的使用
- 简单了解springboot加载启动原理（后续会有专题讲解）

编辑于 2018-06-13

[Spring Boot](#)[Spring Cloud](#)[Spring MVC](#)[赞同 1](#)[添加评论](#)[分享](#)[收藏](#)

知乎



首发于

springboot快速入门



springboot快速入门

全面介绍springboot相关内容

[进入专栏](#)

推荐阅读



10分钟详解Spring全家桶7大知识点

拉勾·开悟训练营



浅析SpringIOC容器

韩数

5 分

程序

还没有评论

写下你的评论...



▲ 赞同 1 ▼

● 添加评论

➦ 分享

★ 收藏

