1.SpringBoot 底层实现

springboot相当于大的工具箱（各种各样的组件），可以根据自己的项目所需要的工具（组件）直接从springboot里面拿，而不需要大量的配置。即开箱即用，自动装配，组件自动整合在一起

@SpringBootApplication

- @SpringBootConfiguration

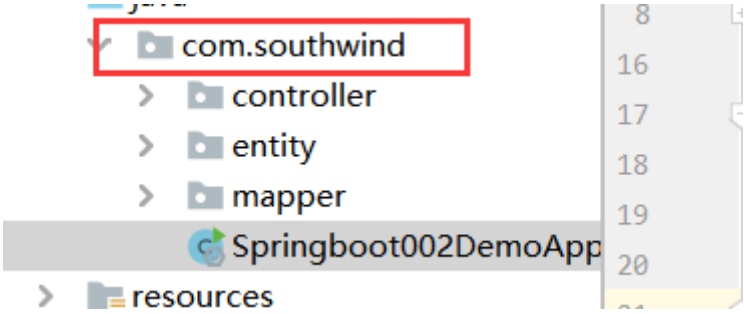
本质上就是一个@Configuration注解，作用是一样的。把一个类标注为配置类

- @EnableAutoConfiguration

```
@AutoConfigurationPackage
@Import({AutoConfigurationImportSelector.class})
```

@Import注解的作用是根据传入的参数Class所返回的信息，将对应的组件进行注入

@Import({Registrar.class}) Registerj中将启动类所在的包作为基础包进行扫描，所以工程中的组件必须被启动类所覆盖，即启动类和组件在一个包下



@Import({AutoConfigurationImportSelector.class})

查找项目路径下所有依赖的**META-INF/spring-factories**(

Springboot启动后找到所需依赖下的配置文件META-INF/spring-factories,从而找到配置类，配置类执行，配置文件中的东西就被读取到，然后被springboot加载进去，即为自动装配

里面存储各种组件的配置类的信息，如mybatis中的配置类，该注解扫描mybatis中的spring—factories文件，springboot将其中mybatis的配置类加载进去)

- @ComponentScan

@ComponentScan注解的作用是根据指定的路径进行扫描，把需要装载的组件进行注入，使用的时候是通过制定具体的包名进行扫描

```
@Data
@Component
public class AccountVo {
    private Integer  id;
    private String name;
}
```

```
@SpringBootConfiguration
@EnableAutoConfiguration
@ComponentScan({"com.southwind.entity","com.southwind.vo"})
public class SpringbootO02DemoApplication {

    public static void main(String[] args) {

        ConfigurableApplicationContext run =
SpringApplication.run(SpringbootO02DemoApplication.class, args);
        System.out.println(run.getBean(Account.class));
        System.out.println(run.getBean(AccountVo.class));

    }
}
```

```
Account(id=null, name=null, money=null, password=null)
AccountVo(id=null, name=null)
```

自定义Import注入：

1.创建实体类

```
import lombok.Data;

@Data
public class Account {
    private Integer id;
    private String name;
    private String money;
    private String password;
}
```

2.创建AccountImportSelector

```
import org.springframework.context.annotation.ImportSelector;
import org.springframework.core.type.AnnotationMetadata;

public class AccountImportSelector implements ImportSelector {
    @Override
    public String[] selectImports(AnnotationMetadata importingClassMetadata) {
        return new String[]{"com.southwind.entity.Account"};
    }
}
```

3.创建AccountImportConfiguration

```
import org.springframework.context.annotation.Import;

@Import(AccountImportSelector.class)
public class AccountImportConfiguration {
}
```

4.创建META-INF下的 spring.factories

```
org.springframework.boot.autoconfigure.EnableAutoConfiguration=\
com.southwind.autoconfigure.AccountImportConfiguration
```

5.获取bean

```
@SpringBootApplication
@MapperScan("com.southwind.mapper")
public class Springboot002DemoApplication {

    public static void main(String[] args) {

        ConfigurableApplicationContext run =
SpringApplication.run(Springboot002DemoApplication.class, args);
        System.out.println(run.getBean(Account.class));
    }
}
```

```
2022-03-20 14:29:35.999  INFO 21012 --- [           main] o.s.b.w.embed
2022-03-20 14:29:36.006  INFO 21012 --- [           main] c.s.Springboo
Account(id=null, name=null, money=null, password=null)
```

2.Starter

各个组件都是通过Starter的形式自动注入到springboot中的

mysprngboot002-->springboot002_demo

1.导入自动装配依赖

```
<!--自动装配-->
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-autoconfigure</artifactId>
</dependency>
```

2.创建service

```
import lombok.extern.slf4j.Slf4j;

@Slf4j
public class Service {
    public Service(){
        log.info("这是通过无参构造创建的service");
    }
}
```

3.创建ServiceImportConfiguration

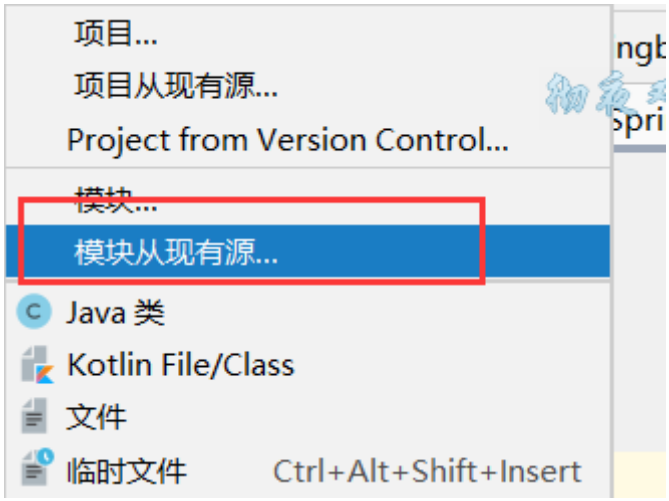
```
import com.ishang.service.Service;
import lombok.extern.slf4j.Slf4j;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
```

```
@Slf4j
@Configuration
public class serviceImportCongiguration{
    @Bean
    public Service service(){
        log.info("这是ServiceAutoConfigure创建了service对象");
        return  new Service();
    }
}
```

4.创建spring.factories

```
org.springframework.boot.autoconfigure.EnableAutoConfiguration=\
com.ishang.configure.serviceImportCongiguration
```

5.在springboot002_demo项目中导入子模块，通过pom导入



点击需要导入模块的pom文即可

6.在springboot002_demo的pom文件中导入子模块的依赖

```
<dependency>
    <groupId>com.ishang</groupId>
    <artifactId>myspringboot002</artifactId>
    <version>0.0.1-SNAPSHOT</version>
</dependency>
```

7.调用

```
@MapperScan("com.southwind.mapper")
@SpringBootApplication
public class Springboot002DemoApplication {

    public static void main(String[] args) {
         ConfigurableApplicationContext run =
SpringApplication.run(Springboot002DemoApplication.class, args);
         System.out.println(run.getBean(Service.class));

    }
}
```

```
    /              |
                     3.4.3.1
2022-03-20 15:48:51.129  INFO 28620 --- [           main] o.s.b.a.w.s.WelcomePageHandlerMapping    : Adding welcome page template: index
2022-03-20 15:48:51.221  INFO 28620 --- [           main] c.i.c.serviceImportCongiguration         : 这是ServiceAutoConfigure创建了service对象
2022-03-20 15:48:51.221  INFO 28620 --- [           main] com.ishang.service.Service               : 这是通过无参构造创建的service
2022-03-20 15:48:51.278  INFO 28620 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat started on port(s): 8080 (http) with context path ''
2022-03-20 15:48:51.286  INFO 28620 --- [           main] c.s.Springboot002DemoApplication         : Started Springboot002DemoApplication in 2.9 seconds (JVM running for 4.227)
```