Spring-boot入门:

网址:https://start.spring.io/

一.什么是spring-boot:

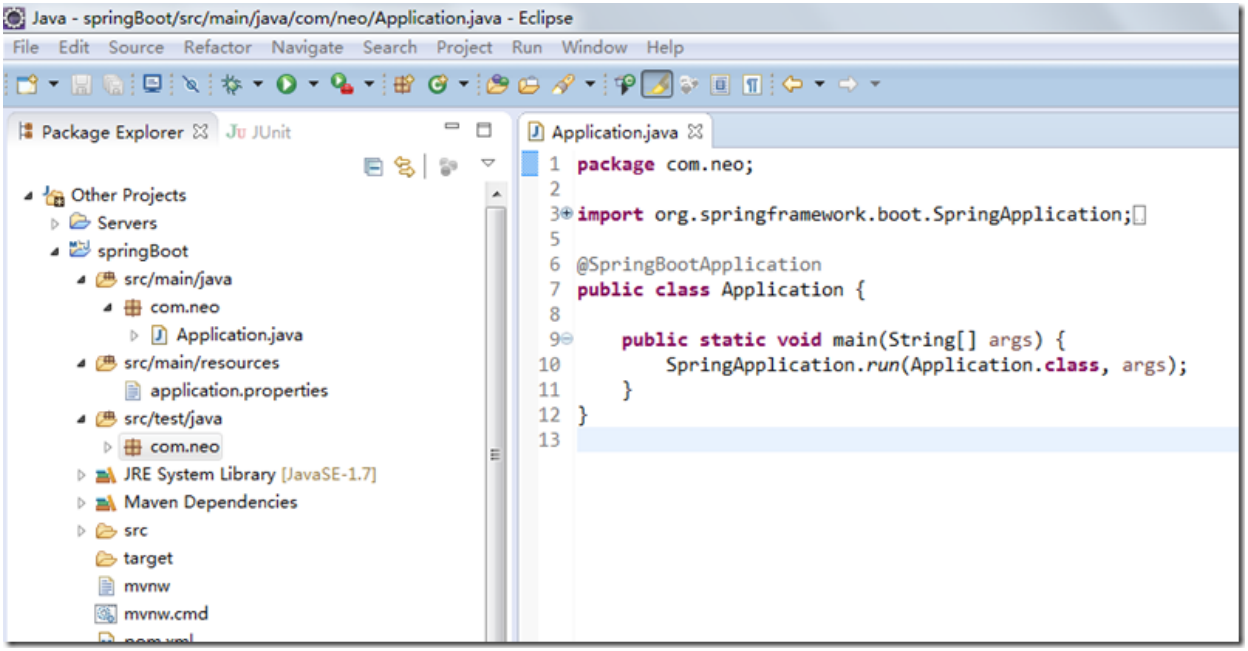
```
Spring Boot是由Pivotal团队提供的全新框架，其设计目的是用来简化新
Spring应用的初始搭建以及开发过程。
```

二.好处:

```
非常少的几个配置就可以迅速方便的搭建起来一套web项目或者是构建一个微服务！
```

三.maven构建项目:

```
maven构建项目
·          1、访问http://start.spring.io/
·          2、选择构建工具Maven Project、Spring Boot版本1.3.6以及一些工程基本信息，
点击“Switch to the full version.”java版本选择1.7.
·          3、点击Generate Project下载项目压缩包
·          4、解压后，使用eclipse，Import ->
 Existing Maven Projects ->
 Next ->选择解压后的文件夹-> Finsh，OK done!
```



如上图所示，Spring Boot的基础结构共三个文件:

```
·          src/main/java 程序开发以及主程序入口
·          src/main/resources 配置文件
```

```
3 ·          src/test/java 测试程序
```

另外，spingboot建议的目录结果如下：

root package结构：

```
com.example.myproject
com
  +- example
    +- myproject
      +- Application.java
      |
      +- domain
      |  +- Customer.java
      |  +- CustomerRepository.java
      |
      +- service
      |  +- CustomerService.java
      |
      +- controller
      |  +- CustomerController.java
      |
```

1、Application.java 建议放到跟目录下面,主要用于做一些框架配置

2、domain目录主要用于实体（Entity）与数据访问层（Repository）

3、service 层主要是业务类代码

4、controller 负责页面访问控制

采用默认配置可以省去很多配置，当然也可以根据自己的喜欢来进行更改

最后，启动Application main方法，至此一个java项目搭建好了！

图示:(启动类)

UserController.java | SpringBootDemo01Application.java | application.properties | SpringBootDemo01ApplicationTests.java

```java
package com.lanou.springBootdemo01;

import org.springframework.boot.SpringApplication;

@SpringBootApplication
public class SpringBootDemo01Application {

    public static void main(String[] args) {
        SpringApplication.run(SpringBootDemo01Application.class, args);
        //启动类
    }

}
```

四.引入web模块

1、pom.xml中添加支持web的模块：

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

pom.xml文件中默认有两个模块：

`spring-boot-starter` **：核心模块，包括自动配置支持、日志和YAML；**

`spring-boot-starter-test` **：测试模块，包括JUnit、Hamcrest、Mockito。**

2、编写controller内容：

```java
@RestControllerpublic class HelloWorldController {
    @RequestMapping("/hello")
    public String index() {
        return "Hello World";
    }
}
```

`@RestController` **的意思就是controller里面的方法都以json格式输出，不用再写什么jackjson配置的了！**

```
3、启动主程序，打开浏览器访问http://localhost:8080/hello，
 就可以看到效果了，有木有很简单！
```

五.如何做单元测试

打开的src/test/下的测试入口，编写简单的http请求来测试；使用mockmvc进行，利用MockMvcResultHandlers.print()打印出执行结果。

```
@RunWith(SpringJUnit4ClassRunner.class)
@SpringApplicationConfiguration(classes = MockServletContext.class)
@WebAppConfigurationpublic class HelloWorldControlerTests {
    private MockMvc mvc;
    @Before
    public void setUp() throws Exception {
        mvc = MockMvcBuilders.standaloneSetup(new HelloWorldController()).build();
    }
    @Test
     public void getHello() throws Exception {
     mvc.perform(MockMvcRequestBuilders.get("/hello").accept(MediaType.APPLICATION_JSON))
                .andExpect(MockMvcResultMatchers.status().isOk())
                .andDo(MockMvcResultHandlers.print())
                .andReturn();
     }
}
```

```
@RunWith(SpringRunner.class)
@SpringBootTest(classes= {SpringBootApplication.class})
@Slf4j
public class SpringBootDemo01ApplicationTests {

    @Test
    public void contextLoads() {
            log.info("11111");
    }

}
```

```
使用Lombok来优雅的编码:https://www.cnblogs.com/qnight/p/8997493.html
```

```
<dependency>
 <groupId>org.projectlombok</groupId>
 <artifactId>lombok</artifactId>
 </dependency>
```

怎样设置:

在escipese中插入lombok.jar,然后在eclipse.ini中添加:

```
-javaagent:lombok.jar
```

```
2 -Xbootclasspath/a:lombok.jar
```

安装插件

由于 Lombok 采取的注解形式的，在编译后，自动生成相应的方法，为了不让 ide 疯了，需要下载插件了支持它。
以 idea 为例：查找插件 lombok plugin 安装即可。

用我的 User 实体类为例（set,get,toString 方法），

```
@Getter
@Setter
@ToString
public class SysUserEntity implements Serializable
```

在按快捷键 Ctrl + F12，可以查找到set,get,toString 方法。

注解

写点常用的，其余的 api 的打开 Jar 包一目了然

@Getter

@Setter

@ToString

@EqualsAndHashCode

构造函数

@AllArgsConstructor

会生成一个包含所有变量，同时如果变量使用了NotNull annotation ， 会进行是否为空的校验，
全部参数的构造函数的自动生成，该注解的作用域也是只有在实体类上，参数的顺序与属性定义的顺序一致。

@NoArgsConstructor

无参构造函数

@RequiredArgsConstructor

会生成一个包含常量（final），和标识了@NotNull的变量 的构造方法。

怎么使用

它们都有三个参数可以设置

1. String staticName() default "";

如果设置了它，将原来的构造方法的访问修饰符将会变成 **私有的**，而外添加一个**静态构造方法**，参数相同，名字是**设置的字符串的名字**，访问修饰符为**公有的**。

1. AnyAnnotation[] onConstructor() default {};
   在构造方法上添加**注解**。使用方法@RequiredArgsConstructor(onConstructor=@__({@AnnotationsGoHere}))

   例如我们在 Spring 项目中需要注入多个值，写很多个 @Autowired 很麻烦，就可以使用这种方式：

```
@Service
@RequiredArgsConstructor(onConstructor = @__(@Autowired))
public class UserServiceImpl implements IUserService {
   private final IUserRepository userRepository;
   private final IOrderRepository orderRepository;
   ……
```

2. AccessLevel access() default lombok.AccessLevel.PUBLIC;
   构造函数访问修饰符；

3. @NoArgsConstructor无参构造函数中还有个注解 boolean force() default false;
   作者的注释是 *If {@code true}, initializes all final fields to 0 / null / false. Otherwise, a compile time error occurs.*

   设置为 true 的时候，初始化所有的参数为默认值，否则编译错误。

@Data

我自己尝试了下，我们使用 @Data 注解就可以有下面几个注解的功能： @ToString、@Getter、@Setter、@EqualsAndHashCode、@NoArgsConstructor 。

> 注意的是，同时使用@Data 和 @AllArgsConstructor 后 ，默认的无参构造函数失效，如果需要它，要重新设置 @NoArgsConstructor

@Slf4j

```
//类上面注解了，直接调用 log 即可：
log.info(xxxx);
```

六.开发环境的调试(热部署).减少了开启服务器,关闭服务器所浪费的时间.

热启动在正常开发项目中已经很常见了吧，虽然平时开发web项目过程中，改动项目启重启总是报错；但springBoot对调试支持很好，修改之后可以实时生效，需要添加以下的配置：

```
<dependencies>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-devtools</artifactId>
        <optional>true</optional>
    </dependency></dependencies>
<build>
    <plugins>
        <plugin>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-maven-plugin</artifactId>
            <configuration>
                <fork>true</fork>
            </configuration>
        </plugin></plugins></build>
```

该模块在完整的打包环境下运行的时候会被禁用。如果你使用java -jar启动应用或者用一个特定的classloader启动，它会认为这是一个“生产环境”。

七.怎样刷新(浏览器F12相当于检查选NetWork,

ctrl+shift+del:清除js缓存.)