1 框架

让程序员专注于业务逻辑, 进而提升开发效率.

框架的主要作用是帮助开发人员快速、高效地开发应用程序，提供一套完整的系统结构、规范的开发流程、通用的功能和模块、配置文件管理、错误和异常管理以及数据库支持等，为开发人员提供了便利的开发工具和方法。

- Java相关框架：`Spring`
- Python相关框架：`Django、Flask、Tornado`

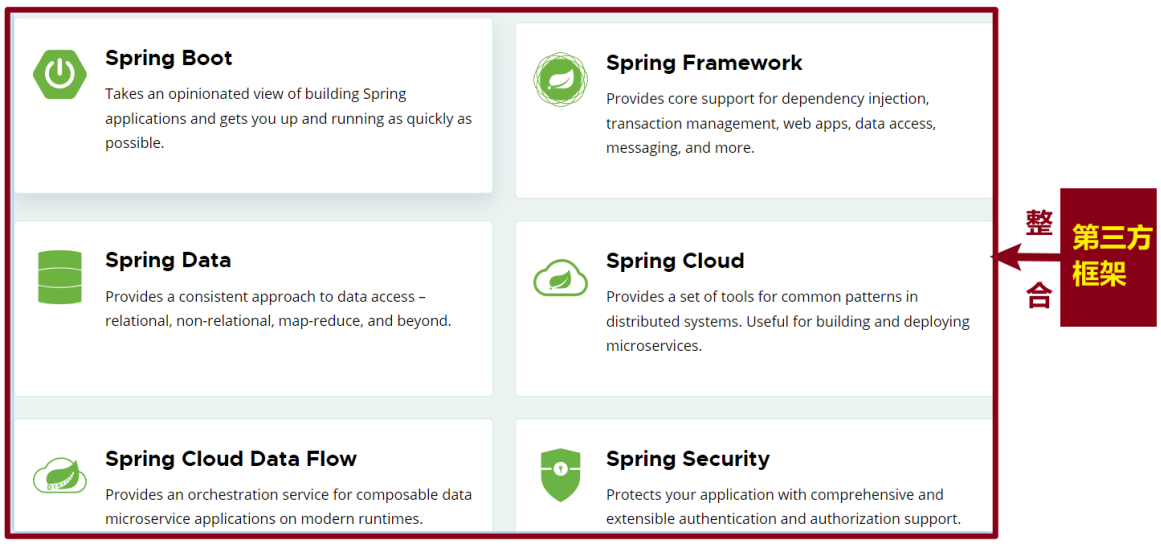
2 Spring框架

官网: https://spring.io

2.1 定义

我们平时所说的spring框架,指的就是 **Spring Framework**

1.Spring是一个资源整合的框架,整合一切可以整合的资源[Spring自身 + 优秀的第三方框架];
2.Spring Framework是整个Spring生态的核心,平时说Spring框架指的是Spring Framework;
3.Spring Framework有两个核心：IoC控制反转 和 AOP面向切面编程.



2.2 Spring版本

- Spring6: 只支持 `JDK17` 及以上版本
- Spring5: 支持 `JDK8-15` 版本,课程中采用: **5.3.24**

2.3 使用流程-注解方式

- 第1步: 添加依赖, 刷新 `maven`;
- 第2步: 在指定类上添加注解: `@Component`
- 第3步: 创建测试类

```
//1.创建IoC容器;
ApplicationContext context = new AnnotationConfigApplicationContext("包路径");
//2.获取Bean;
context.getBean(类型.class);
```

3 对象

- `Java` 对象: 开发人员手动创建的对象, 叫做 `Java` 对象.
- `Spring Bean` 对象: 由 `Spring` 框架创建的对象叫做 `Spring Bean` 对象.

  这两种对象在使用上没有任何差别, 只是为了区分对象创建的方式.

4 相关概念

- IoC

  控制反转的编程思想, 反转资源的获取方向;

  把对象的创建和管理交由框架来完成, 而不是由开发人员手动创建和管理.

- IoC容器

  实现IoC控制反转思想的一种技术手段.

  创建IoC容器

```
ApplicationContext context = new AnnotationConfigApplicationContext("包路径");
ApplicationContext context = new ClasspathXmlConfigApplicationContext("xxx.xml");
```

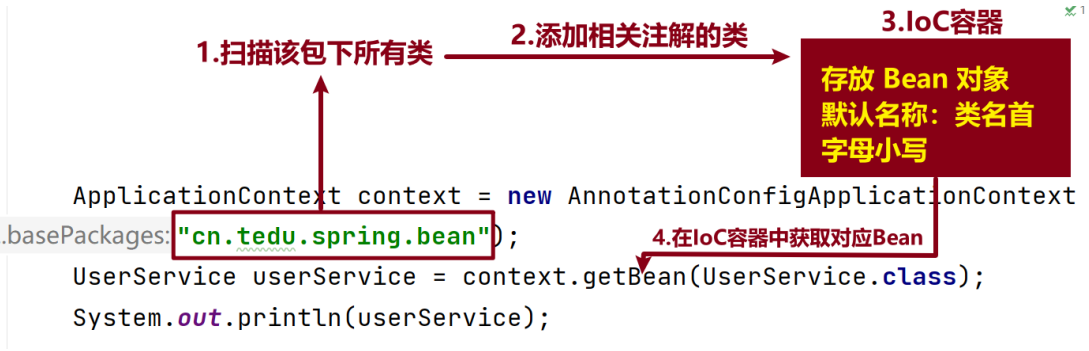
- 依赖注入-DI

  给 `Spring Bean` 对象的属性赋值.

  IoC是控制反转思想, IoC容器和DI依赖注入是实现IoC控制反转思想的两种技术手段.

5 IoC容器实现

- **实现: ApplicationContext 接口.**
- 实现方式
  - 注解方式构建IoC容器: `AnnotationConfigApplicationContext`
  - xml配置文件方式构建IoC容器: `ClasspathXmlApplicationContext`



6 常用注解

6.1 标识为Spring Bean注解

- **Component**
  - 添加在类上，创建Spring Bean对象；
  - 不分层。
- **Controller**
  - 添加在类上，创建Spring Bean对象；
  - 控制器层：负责接收请求并返回响应。
- **Service**
  - 添加在类上，创建Spring Bean对象；
  - 业务层：负责具体业务逻辑处理。
- **Repository**
  - 添加在类上，创建Spring Bean对象；
  - 数据访问层：负责和数据库[**MySQL和MariaDB**]交互。

6.2 DI依赖注入注解

- `@Value`

  添加在属性上, set方法上

  简单类型依赖注入.

- `@Autowired`

  添加在属性上，添加在set方法上，构造方法上

  对象类型或者接口类型依赖注入.

- `@Qualifier`

  根据 `Bean` 对象的名称进行注入, 经常配合 `@Autowired` 注解一起使用;

  当一个接口有多个实现类时, 可以通过 `@Qualifier(value="Bean对象名称")` 指定需要注入的对象.

常用快捷键

- 复制完整路径
  - Windows
    - `Ctrl + Shift + Alt + C`
    - `Fn + Ctrl + Shift + Alt + C`

- Mac
  - `Command + Shift + c`