第1章 绪论

广东财经大学信息学院
罗 东 俊 博士
ZSUJONE@126.COM

（内部资料，请勿外传）

目的和要求

- 了解软件构件技术产生的背景。
- 掌握软件构件的定义、要素及表现形式。
- 理解软件架构、框架、中间件等相关概念。
- 掌握反射和动态代理的机制。

主要内容

- 1.1 软件构件概述
- 1.2 架构、框架与中间件

1.1 软件构件概述

- 1.1.1 软件构件技术的产生
- 1.1.2 软件构件的定义
- 1.1.3 软件构件的要素
- 1.1.4 软件构件的粒度

1.1.1 软件构件技术的产生

- 软件开发方法的螺旋式演进——从结构化到构件化
  - ✧结构化开发方法
  - ✧面向对象开发方法
  - ✧分布式对象方法
  - ✧基于构件的开发方法

结构化开发方法

- 着眼于系统功能，要点是系统的功能分解，即将系统的功能作为划分模块的标准，程序设计从高层模块开始就与当前系统的需求紧密结合，并且使数据结构的设计服从于功能实现的要求。

面向对象开发方法

- 面向对象方法提供了强有力的抽象机制，它使用类和对象作为分析和实现的主要结构。
- 面向对象方法使功能设计人员和开发人员能够把注意力集中到相当稳定的单元上，即系统的类和对象，这些单元与业务概念匹配，可以直接在信息系统中表示。

分布式对象方法

- 分布式对象方法扩展了面向对象方法，使其能够跨越地址空间边界调用对象，一般是利用对象请求代理，例如CORBA对象请求代理。

基于构件的开发方法

- 软件构件（Component）的概念共生于软件复用。
- 软件复用是在软件开发过程中避免重复劳动的解决方案。
  - ✧通过软件复用，可提高软件开发的质量和效率。
  - ✧面向对象技术的产生和发展，为软件复用提供了基本的技术支持。

基于构件的开发方法

- 创建和利用可复用的软件构件来解决应用软件的开发问题，只要遵循构件技术的规范，各个软件开发商就可以用自己方便的语言去实现构件，应用程序的开发人员就可以挑选构件组合新的应用软件，这样的应用软件系统不再是一种固化的整体系统，而是通过构件间相互提出请求和返回服务结果的协同工作机制来达到系统目标。
- 目前市场上已经有大量面向**GUI**、数据库和网络的**ActiveX**构件、**Java Beans**构件、以及众多的类库、**DLL**接口和**API**

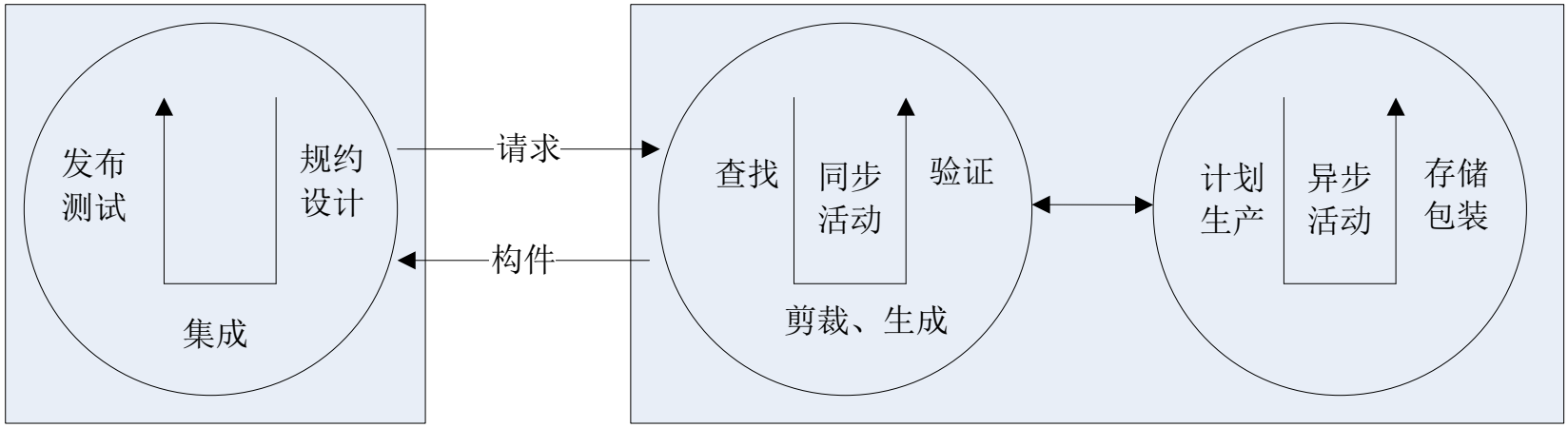
为什么要应用软件构件技术

- 开发工作构建在已有成果的基础上。
- 控制开发复杂性。
- 控制软件系统部署复杂性。
- 简化整个软件需求和开发周期内工作。
- 便于系统升级。
- 较好地利用本组织的最佳方法。
- 降低开发费用。
- 缩短产品投放市场时间。

为什么要应用软件构件技术

- 软件构件技术是软件工厂得以实现的重要理论基础和技术保证。
  - ✧软件工厂是指一种大规模生产软件的组织和方法。
  - ✧软件工厂有两个要素：
    - 软件“元器件”技术
    - 软件“元器件”的组装、链接、合成技术。

一种基于构件的软件工厂模式



1.1.2 软件构件的定义

- 从构件技术提出至今，出现了若干软件构件的定义，其中有代表性的定义包括：
  - ✧软件构件是一个具有规范接口和确定的上下文依赖的组装单元，软件构件能够被独立部署和被第三方组装——1996年ECOOP会议上提出的定义
  - ✧软件构件是可单独生产、获取、部署的二进制单元，它们之间可以互相作用构成一个功能系统——Szyperski在1998年给出的定义

软件构件的特征

- 构件不是完整的应用程序，需要组装
- 构件的特征在于实现软件复用，需要规范
- 构件是预制的知识服务，需要封装

1.1.3 软件构件的要素

- 通常，构件建立于面向对象和分布式基础之上，它有5个要素:
  - ✧规格说明
  - ✧一个或多个实现
  - ✧受约束的构件标准（软件构件模型）
  - ✧包装方法
  - ✧部署方法

软件构件模型

- 软件构件模型是关于开发可重用构件和构件之间相互通信的一组标准的描述，它定义了一种软件构件组装的模式。
  - ✧软件构件存在于构件模型的环境中才能相互协同工作
  - ✧构件模型提供了一套服务，并且定义了构件利用这些服务必须遵守的一套规则，例如提供创建和实现构件的指导原则。
  - ✧构件模型规定了构件接口的结构以及构件与构件之间、构件与软件架构之间的交互机制。
- 典型的构件模型有国际对象管理组织**OMG**的公共对象请求代理结构**CORBA**、**SUN**公司的**EJB**和微软公司的**COM/DCOM/COM+/Web Service**三大类。

1.1.4 软件构件的粒度

- 软件构件粒度的大小是影响软件成本和软件重用质量的重要因素，它表征了构件的规模与复杂程度。
- 它是一个难以清晰定义的概念。
  - ✧一般认为，构件的粒度是用来描述构件所提供特征的粗细程度的量化，或者说是构件所提供特征的计算量的大小，但通常难以精确度量。

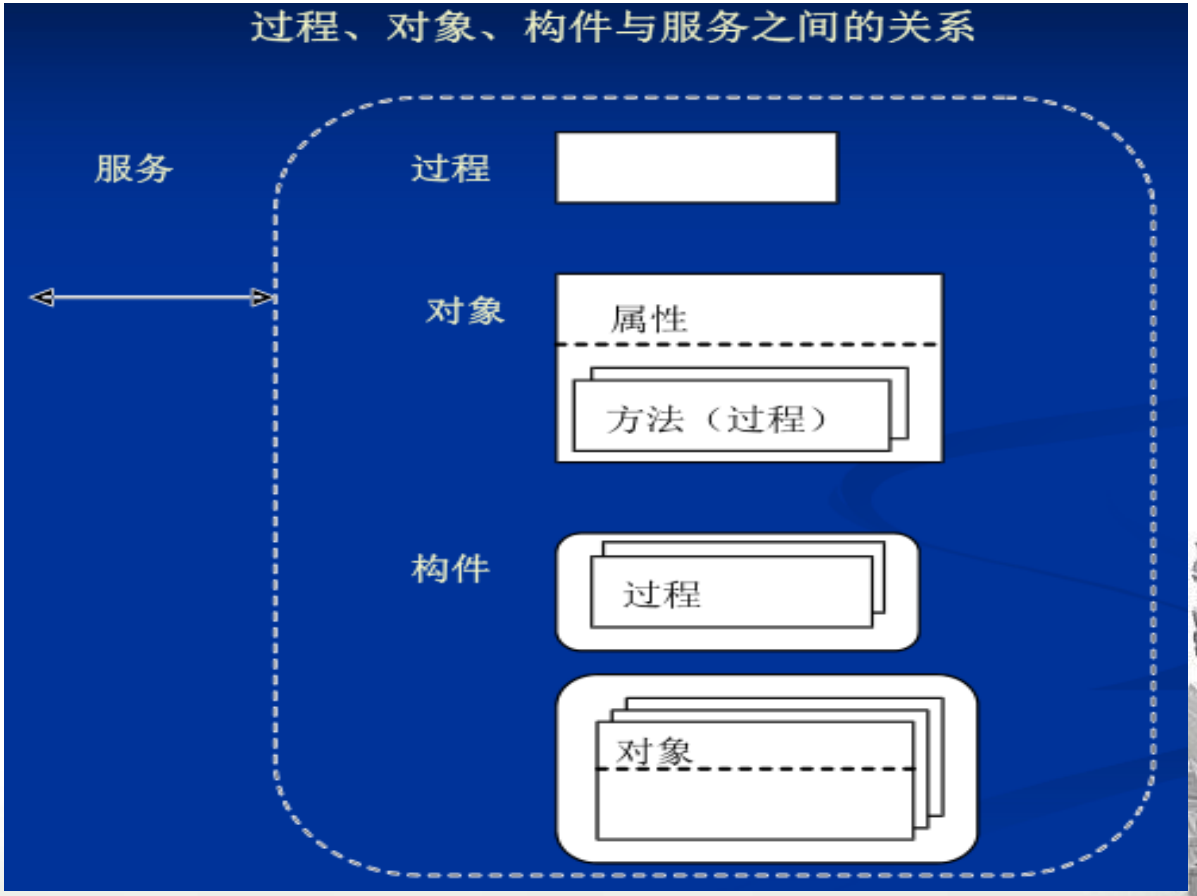
不同粒度的软件构件

- 过程（Procedure）
  - ✧最基本的软件模块
  - ✧是大型程序从混沌向结构化走出的第一步
  - ✧每个过程都有具体的调用格式
    - 在C/C++中，该格式用头文件来说明
- 对象（Object）
  - ✧方法与数据的封装体
  - ✧是类（class）的实例
  - ✧其中的方法与过程有直接的对应关系
  - ✧在一些基于对象的系统中，对象仅仅在编程阶段存在，编译器将对象映射为传统的过程，这样，在运行阶段不再保持对象实体了

不同粒度的软件构件

- 构件（Component）
  - ✧更大粒度的构造模块
  - ✧通常在运行阶段保持构件形态
  - ✧通常构件是由一个或多个类组成的实体
  - ✧也可以直接由一个或多个过程组成
- 服务（Service）
  - ✧更松散的软件实体
  - ✧服务强调的是软件实体的外在表现
  - ✧其内在实现则也是由某个构件、某个对象、甚至某个过程完成

不同粒度的软件构件



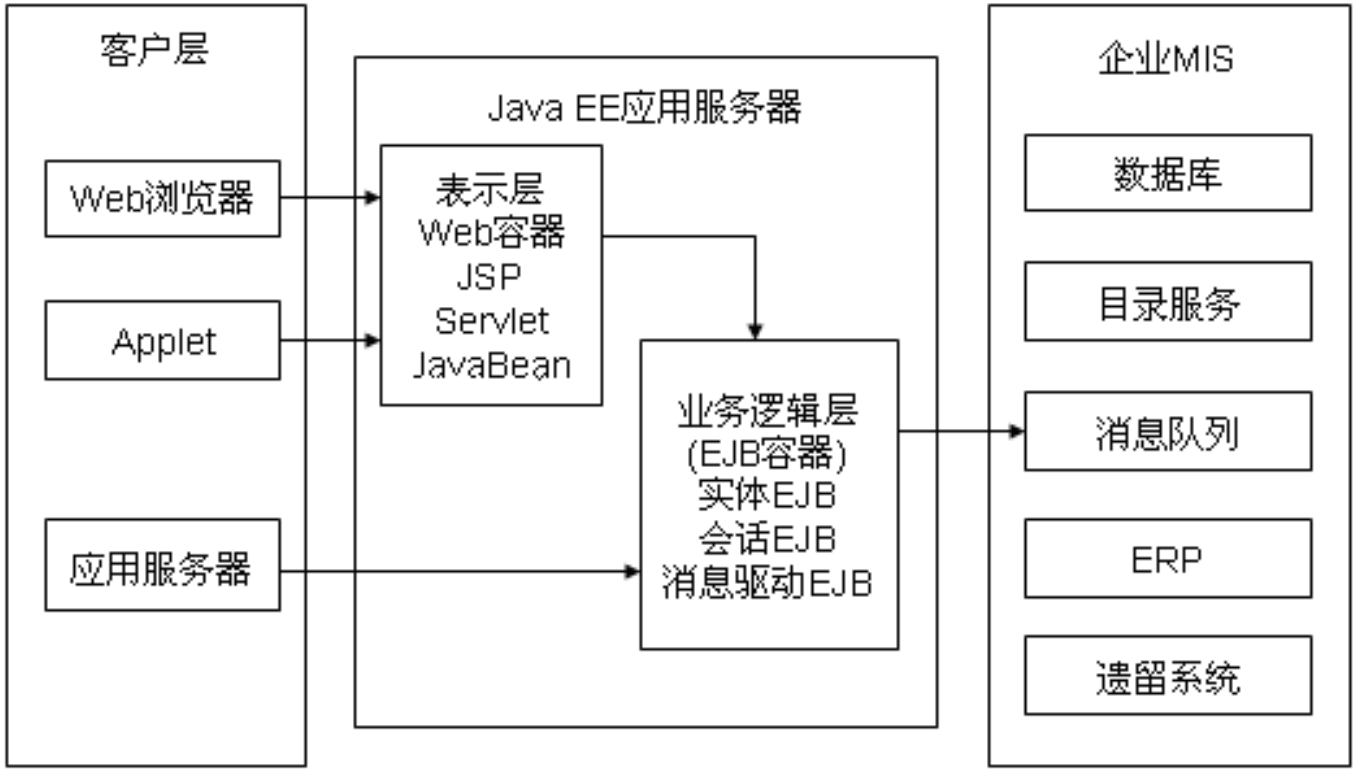
1.2 架构、框架与中间件

- 1.2.1 软件架构
- 1.2.2 软件框架
- 1.2.3 中间件

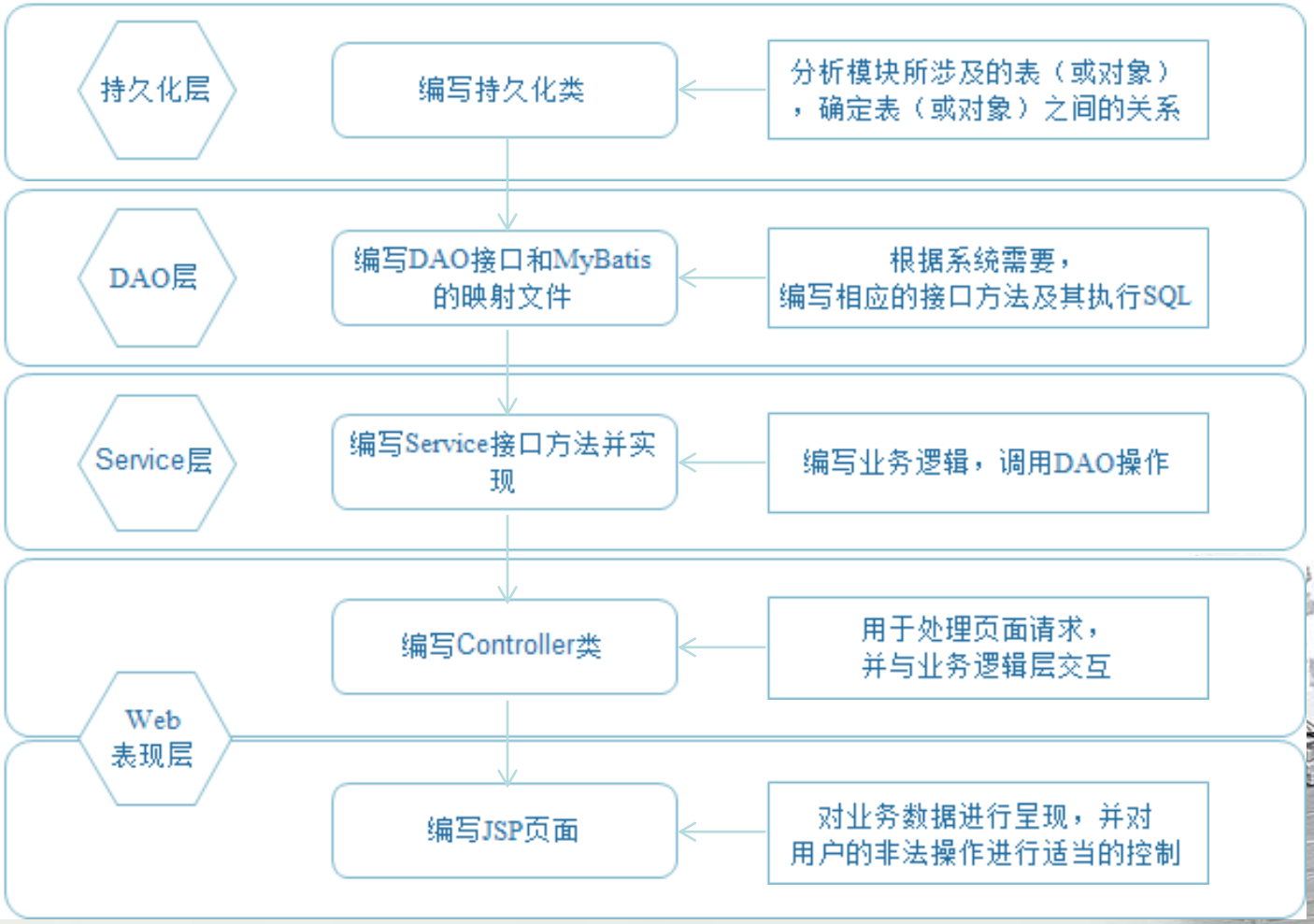
1.2.1 软件架构

- 软件架构，也称为软件体系结构。
- 简单地说，软件架构就是一个蓝图，是一种设计方案，将客户的不同需求抽象成为抽象组件，并且能够描述这些抽象组件之间的通信和调用。
- 它是对软件系统的系统组织，是对构成系统的构件的接口、行为模式、协作关系等体系问题的决策总和。
- 它不仅涉及到结构与行为，而且还涉及到系统的使用、功能、性能、适应性、重用性、可理解性、经济性和技术约束的权衡和美学考虑。

传统的JavaEE架构



典型的JavaEE架构

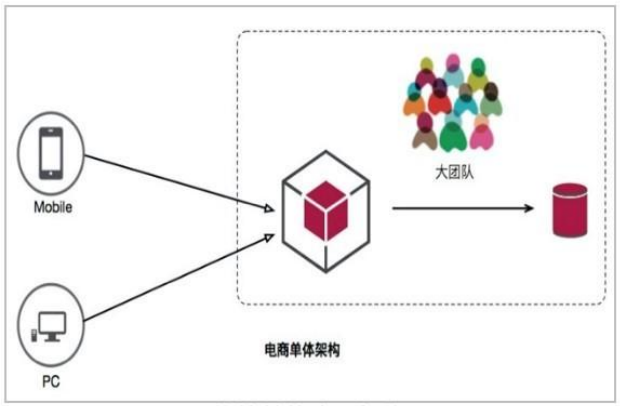


互联网应用架构的演变

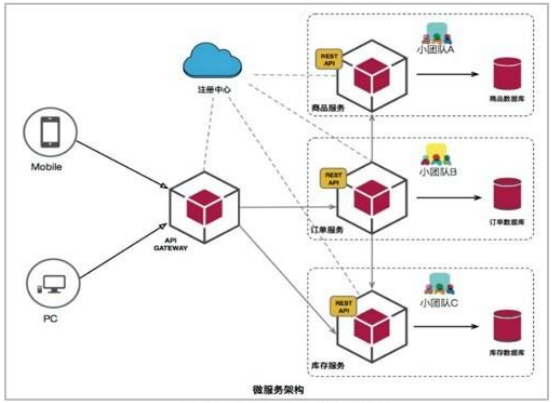
- 随着互联网的发展，网站应用的规模也在不断的扩大。
- 从互联网早起到现在，系统架构大体经历了下面几个过程：
  - ✧单体应用架构—>垂直应用架构—>分布式架构—> SOA架构—>微服务架构，当然还有悄然兴起的Service Mesh(服务网格化)。

互联网应用架构的演变

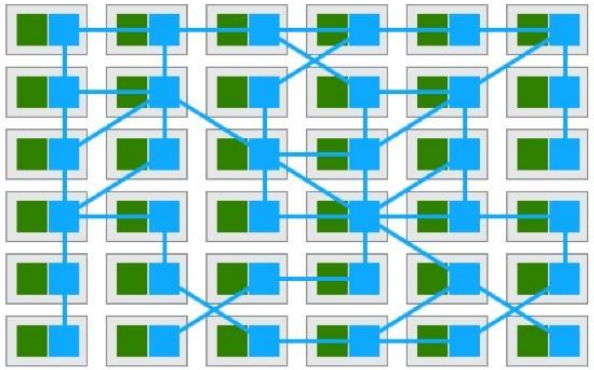
单体应用 → 微服务框架 → Service Mesh



传统单体应用架构



服务化拆分后的架构



- 开发、测试、部署简单
- 项目复杂、扩展性差

- 灵活性、扩展性强
- 引入RPC框架、治理、监控、安全等设施的建设

- 业务无侵入、支持热升级、语言无关
- 技术较新，体系较大

1.2.2 软件框架

- 软件框架是项目软件开发过程中提取特定领域软件的共性部分形成的体系结构。
- 框架不是现成可用的应用系统，而是一个半成品，是一个提供了诸多服务，供开发人员进行二次开发，实现具体功能的应用系统。
  - ✧特别强调，框架是一个可供二次开发的程序实体
  - ✧基于框架的开发极大地提高了软件构造的效率。
- 如：**Spring、Spring MVC、Spring Boot、MyBatis、Struts、Hibernate、JUnit**

框架与架构关系

- 框架不是架构，框架比架构更具体，更偏重于技术，而架构偏重于设计。
- 架构可以通过多种框架来实现。

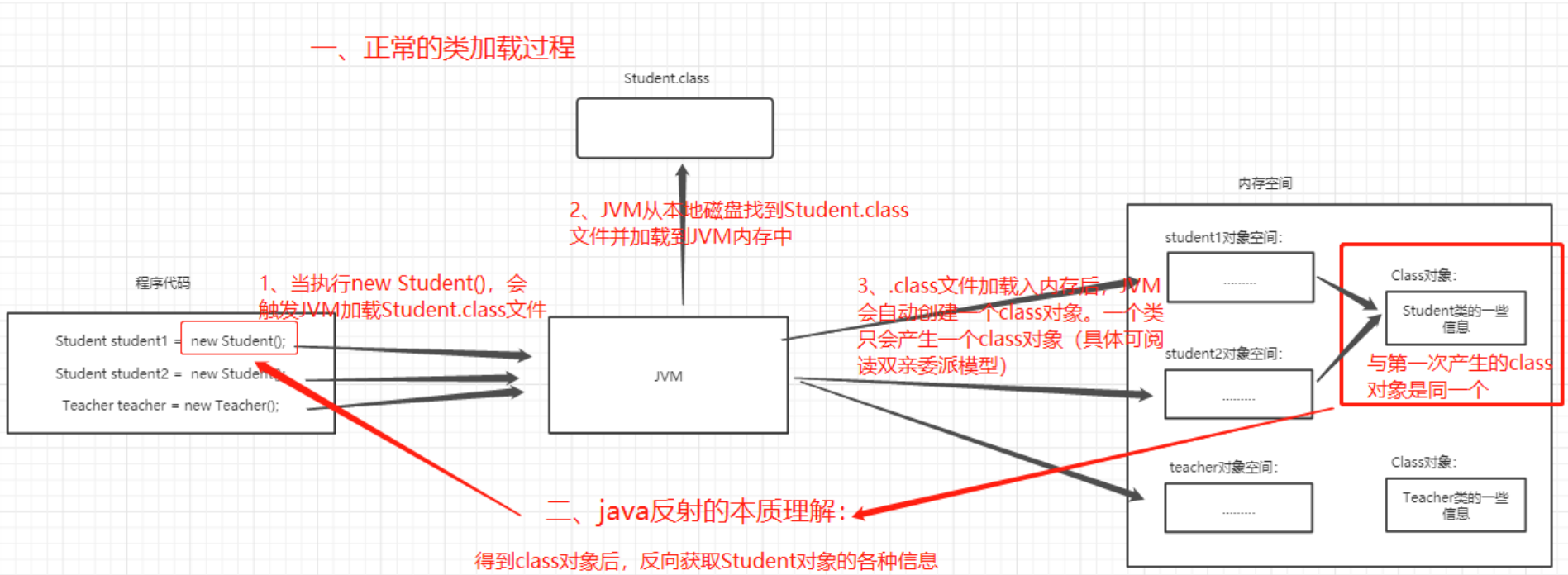
框架运行机制

- 1）反射机制
- 2）动态代理

1）反射机制

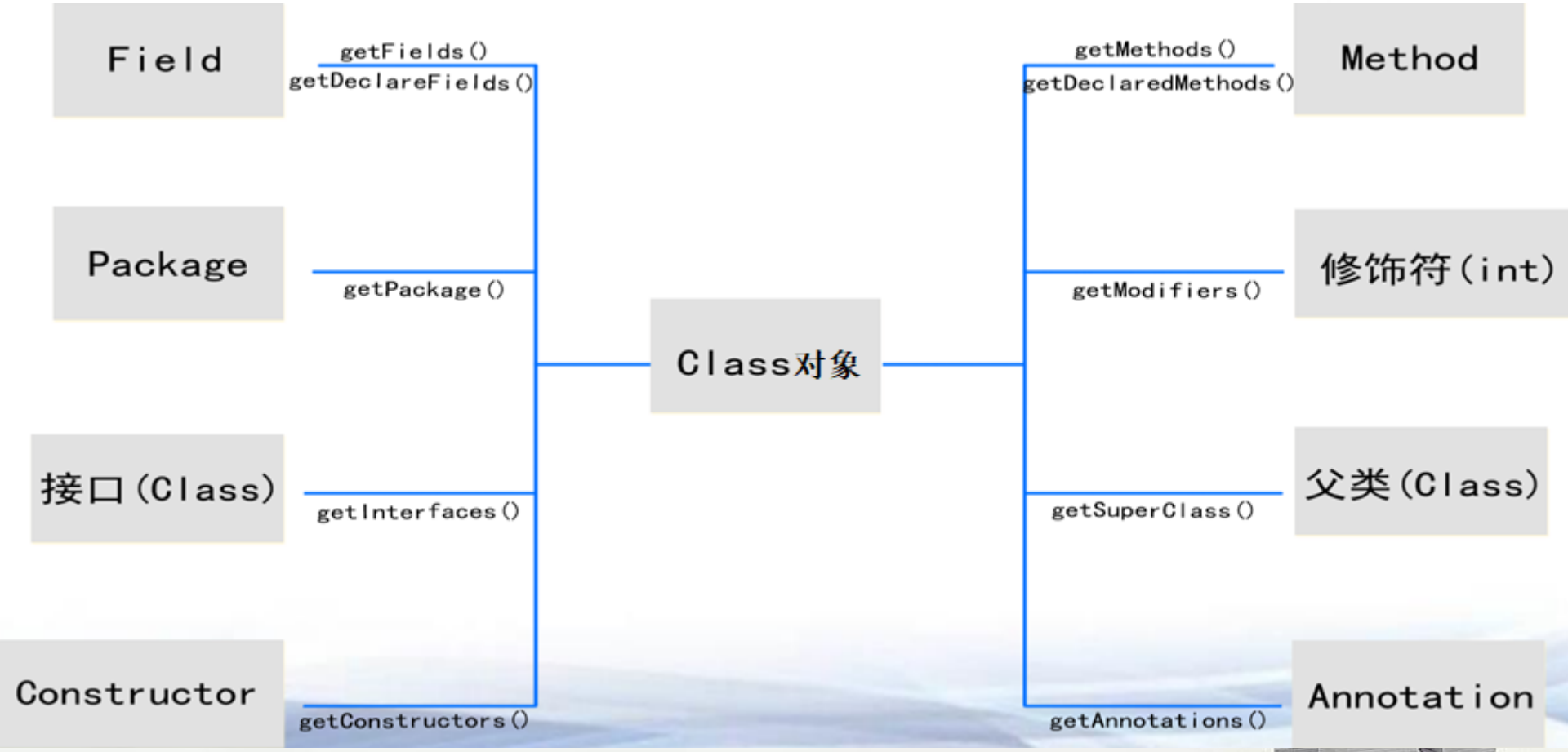
- **Java**反射机制的核心是在程序运行时动态加载类并获取类的详细信息，从而操作类或对象的属性和方法。
  - ✧本质是**JVM**得到**class**对象之后，再通过**class**对象进行反编译，从而获取对象的各种信息。
- **Spring**框架利用**Java**反射机制，使用配置文件把框架的各个组件串联起来，实现代码和配置文件的相互独立。

反射的原理



JVM会为每个类创建一个Class对象（ java.lang.Class类对象），通过该Class对象就可以获取这个类的信息，然后通过使用java.lang.reflect包下的API以达到各种动态需求。

可获取的类信息


Field
getFields()
getDeclareFields()
Package
getPackage()
接口(Class)
getInterfaces()
Constructor
getConstructors()
Class对象
getMethods()
getDeclaredMethods()
Method
getModifiers()
修饰符(int)
getSuperClass()
父类(Class)
getAnnotations()
Annotation

2）动态代理

- 动态代理就是在程序运行期，创建目标对象的代理对象，并对目标对象中的方法进行功能性增强的一种技术。
- 有了动态代理的技术，那么就可以在不修改方法源码的情况下，增强被代理对象的方法的功能，在方法执行前后做任何你想做的事情。
  - ✧从而可以对同一类型Java对象进行统一配置和管理，提升开发效率。

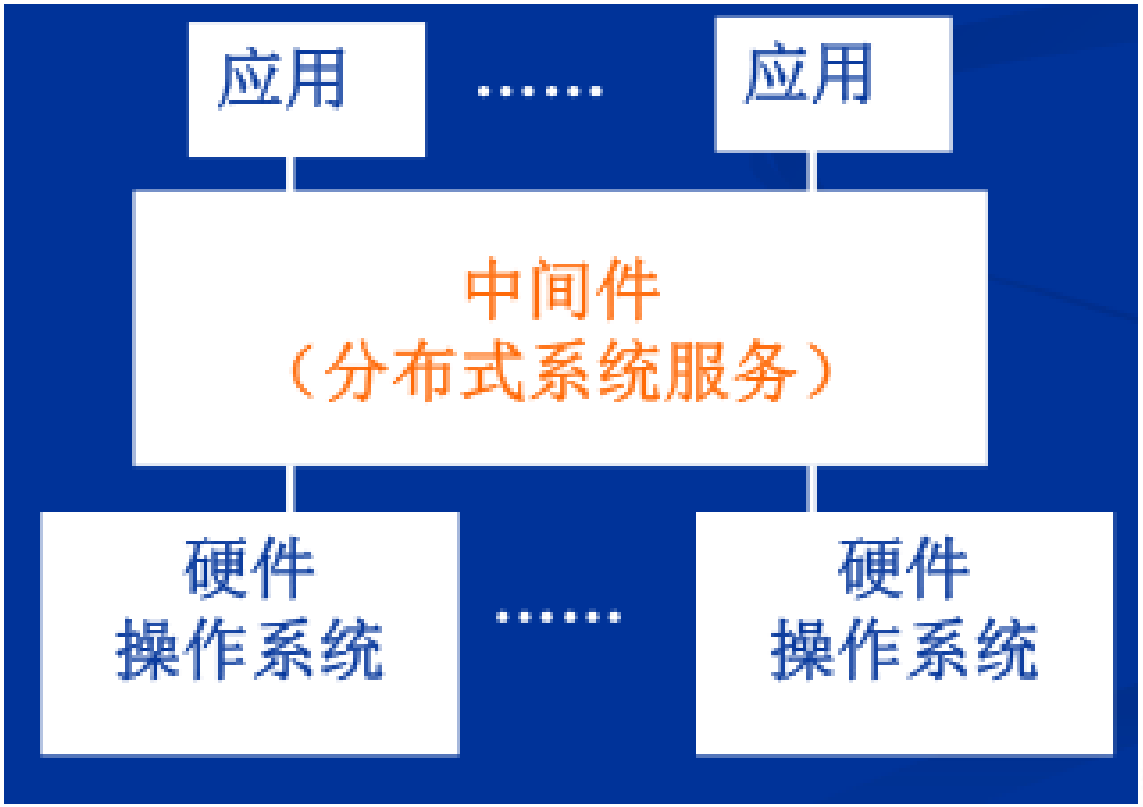
动态代理实现

- 有两类动态代理实现方法：
  - ①JDK动态代理
  - ②CGLIB代理

1.2.3 中间件

- 中间件是一种独立的系统软件或服务程序。
  - ✧分布式应用软件借助这种软件在不同的技术之间共享资源；
  - ✧中间件位于客户机、服务器的操作系统之上，管理计算资源和网络通信。
  - ✧如：**JDBC**、**Tomcat**应用服务器、**Redis**缓存中间件、**RocketMQ**消息中间件等。

中间件



中间件的作用

- 中间件主要是用来支持网络环境中软件实体之间的有效交互。
  - ✧ “软件实体”是指具有不同特征的软件模块，它们通常是具有不同粒度的软件构造模块
  - ✧ “有效”是指所支持的实体之间的交互必须具有可靠、安全、快速等特点
    - 所以要有良好的交互模式+交互质量

本章小结

- 本章具体讲解了：
  - ✧1.1 软件构件概述
  - ✧1.2 架构、框架与中间件


聰明等于勤奋
劳力就是幸福