



COMP130015.02

软件工程

6. 软件复用

复旦大学计算机科学技术学院

沈立炜

shenliwei@fudan.edu.cn



软件复用 (Reuse)

- 从零开始的开发方式 (Develop from Scratch) 已经无法满足软件开发在快速交付、成本控制、质量保障等方面的要求
- 软件复用已经成为一种重要的软件开发手段
 - ✓ 制造业、建筑业：建立在规范化、标准化设计之上的大规模零部件复用和组装式制造和建造
 - ✓ 软件业：通过源代码、模块、组件、框架和平台等各个层次上的复用实现提高软件开发质量和效率的目标
- 重复利用已有知识、经验或软件制品来开发新软件产品
- 能够有效提高开发效率、降低开发成本、提高软件质量，并且缩短产品发布周期

软件复用带来的变化

• 有力推动了通用软件资产的沉淀

- ✓ 不同软件项目在需求、设计、代码等层面上都有可能存在一些共性和通用的部分
- ✓ 这些共性和通用的部分可以逐渐沉淀形成不同形式的通用软件资产，例如软件组件库、技术框架、开放平台等

• 极大促进了软件行业的分工

- ✓ 软件开发围绕通用的软件平台、框架、组件库以及面向特定需求的软件应用形成了企业内以及社会化的分工合作
- ✓ 例：开源社区可以共同开发并维护一个通用开发框架，一些企业开发并销售报表、数据分析等通用功能组件，而另一些企业则利用通用开发框架和通用组件开发应用软件
- ✓ 例：基于一些互联网企业推出的开放开发平台，小型软件企业和个人开发者可以开发一些通用组件或应用，并依托平台获得使用流量和收益

软件复用在软件开发中的比重






- 各种形式的软件复用已经成为软件开发的重要支撑手段
- 通过复用方式获得的软件组件和其他软件制品在软件项目中已经占据了相当的比重，如下例
 - ✓ 复用的对象大部分为第三方库，包括对高层框架和平台的使用
 - ✓ 可见组件级的复用已经成为软件开发中的重要组成部分

	自研	复用
代码量	2.8亿行	1.2亿行
组件数量	600+	400+

某国内大型IT企业一款产品中自研与复用的代码和组件数量对比

软件复用层次

需求复用意味着与之对应的设计方案和实现代码都可能随之复用
软件设计产生的体系结构设计和组件级设计也可能具有可复用性

需求分析	软件设计	编码	测试
 软件特性 功能定义 使用场景 业务流程	 体系结构设计  组件级设计		 测试规程  测试用例

编码阶段所产生的源代码及其各种打包形式(例如Jar包)是更常见的复用形式

组件	 Apache LOG4J  fastjson  HTTP COMPONENTS Apache HttpClient	 编程指南
框架	 spring boot  MyBatis  Vue.js	
平台 (小程序)	 微信  支付宝  WeLink WeLink	

但一般不推荐通过代码复制粘贴和修改实现软件复用,因为这将导致相似代码散布在很多地方,难以进行跟踪和统一维护

推荐的代码复用形式是经过良好封装以及清晰的API(Application Programming Interface, 即应用程序接口)定义的软件组件,例如以Jar包形式提供的Java组件

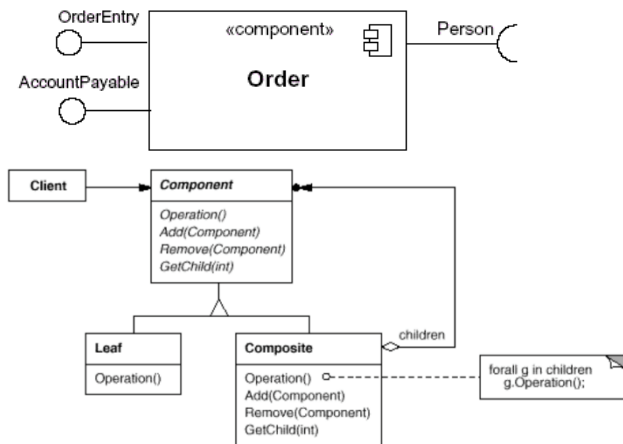
组件级软件复用

- 主要目的是实现特定功能和特性，形式主要是软件组件（或称软件构件）
- 这些软件组件实现特定的功能并明确定义了API，软件开发人员可以通过API调用的方式复用相关的功能实现
- 按照可复用范围分为三类
 - ✓ 一方库：当前项目内的通用功能模块，一般通过项目内的接口定义为项目中的多个模块提供共性的功能实现
 - ✓ 二方库：同一企业内所积累的通用功能模块，一般建议封装成软件组件并提供API描述同时由专人进行维护和升级，
 - ✓ 三方库：由开源社区或企业所提供的广泛授权使用的通用功能模块，一般都有规范的发布包和对应的API描述

框架级软件复用

- 主要目的是获得支撑应用运行的整体性框架，形式主要是各种软件开发框架
- 在共性实现的基础上支持定制和扩展
 - ✓ 定义并实现了面向特定类型应用或其局部（例如Web应用的前端、后端和数据库访问）的整体设计及共性部分
 - ✓ 同时支持应用开发进行定制和扩展
- 与组件级复用的区别
 - ✓ 组件级复用：应用代码通过调用组件的API实现软件复用
 - ✓ 框架级复用：按照框架规范实现的应用代码被框架调用，因此其中经常会用到类似于依赖注入这样的设计思想
- 常用开发框架：Vue.js（web前端）、Spring Boot（web后端）、MyBatis（对象关系映射）

组件、模式、框架

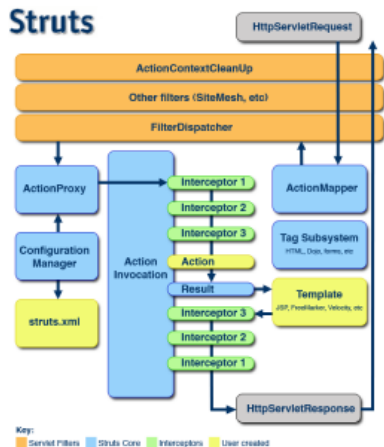


软件组件

(如图片处理、打印组件)

设计模式

(如组合模式、观察者模式)

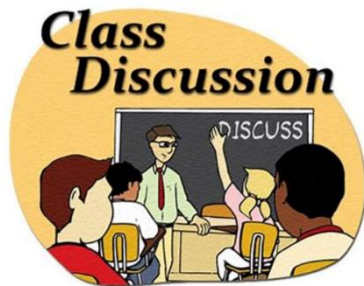


设计框架

(如Struts、JhotDraw)



课堂讨论：组件、模式与框架



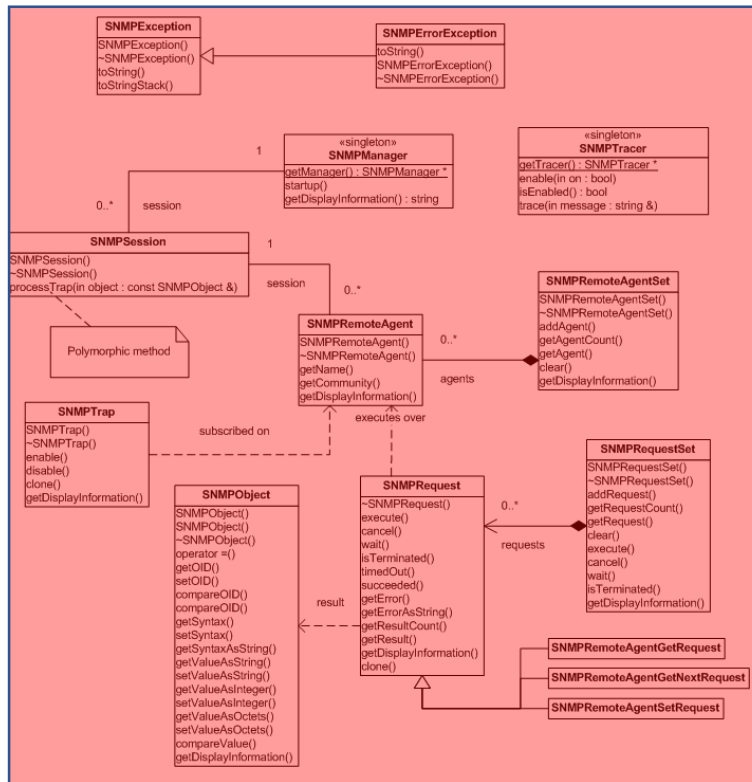
课堂讨论：软件组件、模式与框架都可以被复用，但它们在复用的内容以及方式上有什么区别？

软件组件：针对特定功能提供可复用的实现，开发人员在自身已有的实现方案基础上调用组件实现局部功能

设计模式：一种抽象的设计思想，往往体现为参考设计方案（如用UML图表示），本身并没有代码实现，需要针对具体问题、参考其设计思想进行实现

软件框架：本身包含相对完整的设计以及核心实现，提供扩展和定制能力，开发人员针对特定应用的实现通过扩展点插入框架中，一般由框架来调用形成完整的应用实现

软件组件复用示意（关键词：调用）



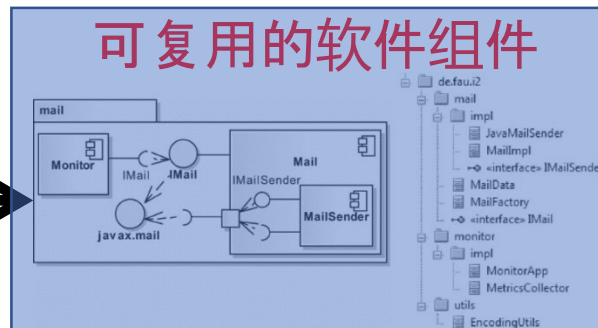
特定应用实现



被复用的部分



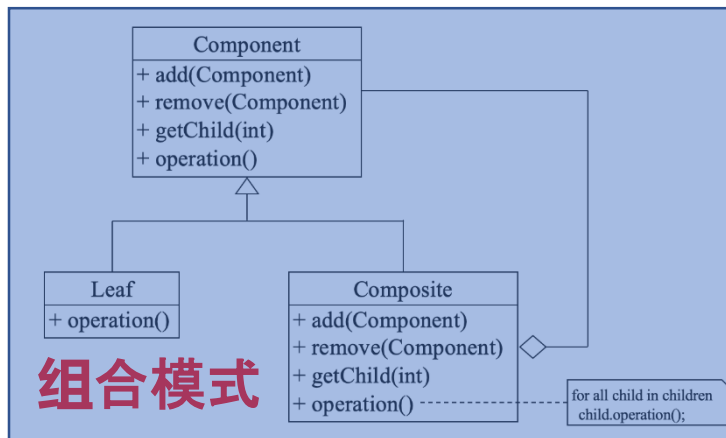
特定应用实现的部分



实现具体功能(如邮件发送)，有自己的实现并通过接口提供访问

特定应用实现调用软件组件实现局部功能，组件对系统整体设计影响不大

设计模式复用示意（关键词：实例化）



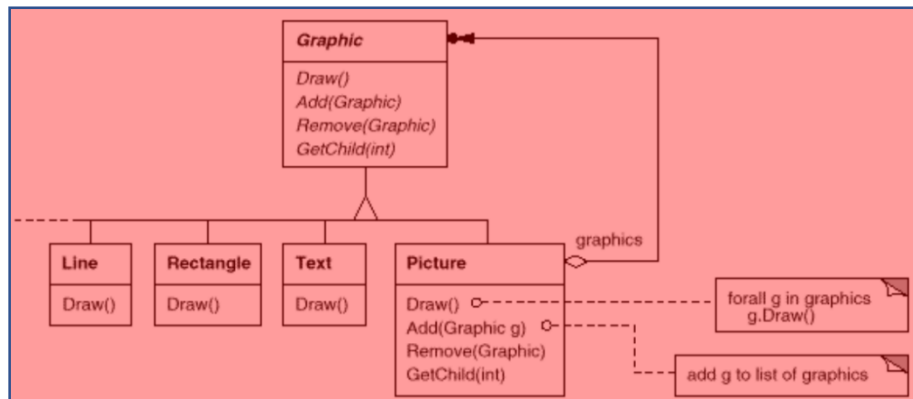
被复用的部分



特定应用实现的部分

反映的是抽象的设计思想，且只关注于某个局部设计问题，其中的类都是与特定应用无关的抽象角色，没有实现代码

特定应用实现对设计模式中的角色及其关系进行实例化，得到特定用的设计和实现



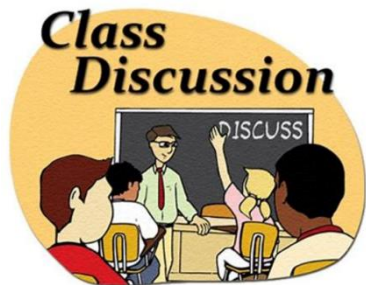
基于设计模式的实例化实现，其中的类都是针对特定应用的设计类，存在对应的实现代码

应用了组合模式的一个设计实例

平台级软件复用

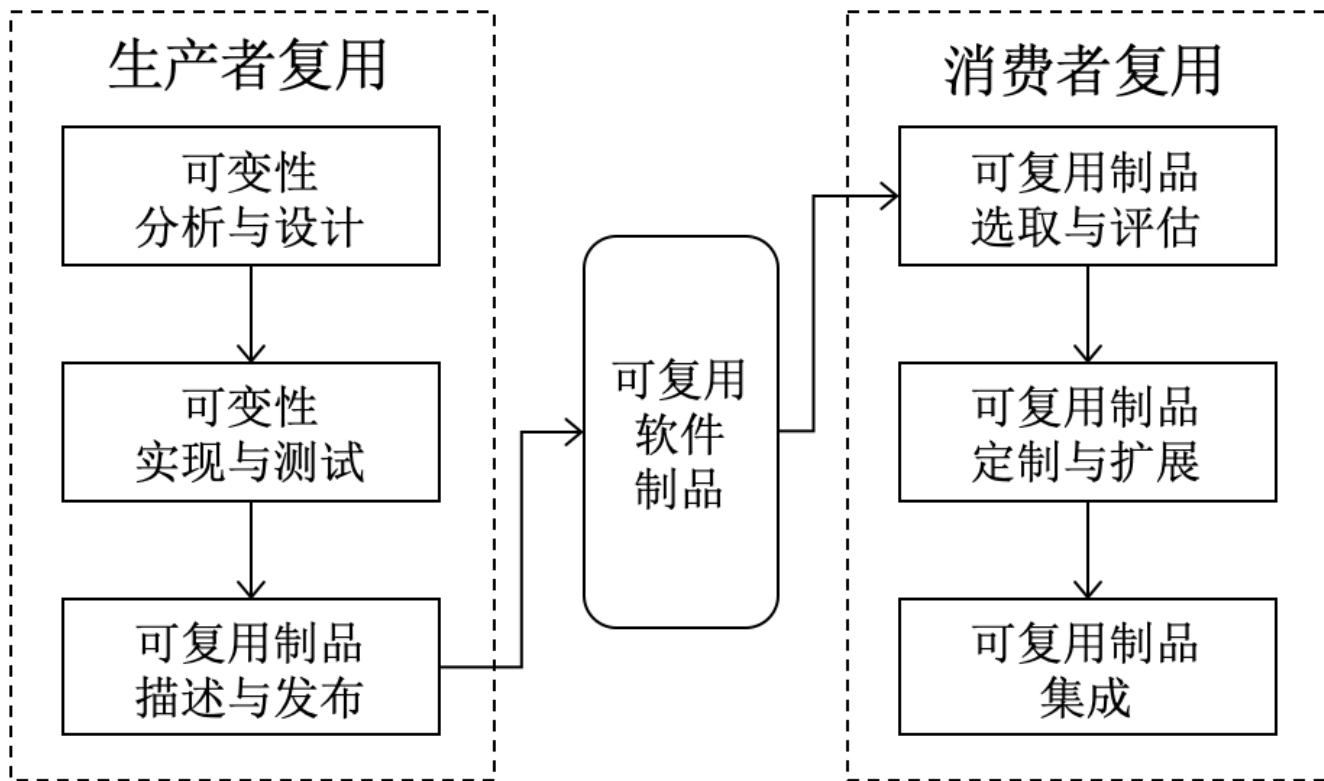
- 主要目的不仅包括获得特定功能和特性的实现以及整体框架支持，还包括获得软件应用的部署和运行支撑
- 以小程序等形式支持应用开发、部署和运行
 - ✓ 平台为应用开发提供了整体开发框架以及相应的开发规范
 - ✓ 以服务的形式提供了身份认证、二维码扫描、用户管理、支付、消息发送等通用功能
 - ✓ 为应用提供基于云的计算、存储和网络资源支持
 - ✓ 应用可以以一种透明的方式实现云化部署和运维支撑，甚至平台还可以提供应用推广、收益分成等运营支持
- 典型代表是各种互联网应用开放平台，例如：微信、支付宝、WeLink 平台

课堂讨论：平台级复用



课堂讨论：你是否在互联网开放平台上开发过小应用？如何理解平台在开发、测试、部署、运行等方面提供的支持？

软件复用过程

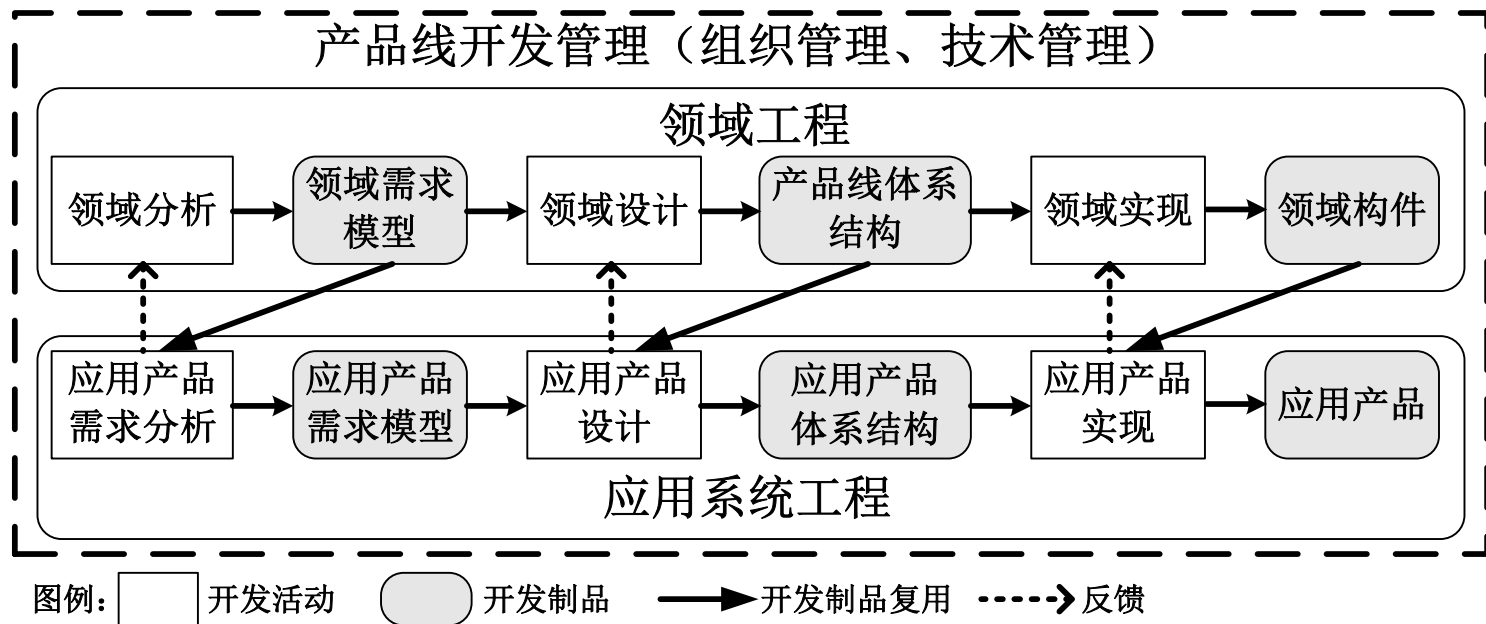


软件产品线

- 汽车等制造业中已经广泛实现了大规模定制化生产，即在标准化产品设计的基础上兼顾大规模生产和个性化定制
 - ✓ 针对特定车型的生产线可以实现大规模汽车组装生产
 - ✓ 同时为客户提供一系列定制选项（如追求舒适还是运动性能、是否加装导航仪等）
- 软件工程领域逐渐也形成了软件产品线的思想
 - ✓ 针对特定领域中一系列具有共性特征的软件应用，通过对领域共性和可变性的把握构造一系列领域核心资产
 - ✓ 面向应用需求的软件应用可以在这些核心资产基础上通过定制和扩展快速、高效地构造出来
 - ✓ 例如：支持面向不同客户进行定制的财务管理系统产品线、支持面向不同型号硬件进行定制的通信系统产品线

软件产品线工程框架

领域工程是其中的核心部分，是领域核心资产(领域需求模型、产品线体系结构、领域组件等)的生产阶段，体现生产者复用的

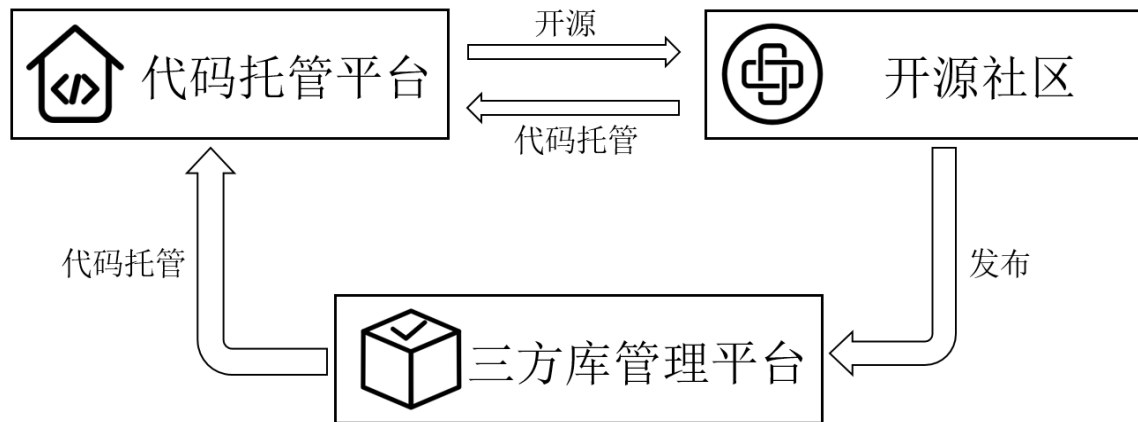


应用系统工程在领域核心资产基础上通过定制化实现特定应用开发
产品线管理从技术和组织两个方面为软件产品线的构建和长期发展提供管理支持

开源软件复用

代码托管平台支持对源代码、文档和其他软件制品进行存档和版本管理，开发者依赖成熟的代码托管平台对自己的项目进行开发、维护和版本控制

开源社区根据相应的许可证协议公布软件源代码，由拥有共同兴趣爱好的人所组成，同时也为网络成员提供一个自由学习和交流的空间



如果开源软件属于通用第三方库，开发人员可以将其成熟版本发布到三方库管理平台，平台保存编译和打包后的库文件同时提供三方库在代码托管平台中的地址，从而方便开发者获取源代码

主流代码托管平台

托管平台	概述
GitHub	2008年正式上线，已成为全球最流行的代码托管平台，目前属于微软旗下产品。GitHub中个人用户居多。根据2020年度Octoverse报告，GitHub已经拥有超过2亿个仓库，并吸引全球超过5600万的开发者。
GitLab	GitLab支持无限的开源和私有项目，因此在企业开发中应用广泛。相比Github，团队对仓库拥有更多权限控制，并可部署自己的服务器。
Bitbucket	Atlassian公司提供的基于web的版本库托管服务。Bitbucket既提供免费帐号，也提供商业付费方案。一般面向专业开发者，能够与Atlassian的其他产品（例如Jira, HipChat等）集成。
CodeHub	代码托管（CodeHub）源自华为千亿级代码管理经验，基于Git，提供企业代码托管的全方位服务，为软件开发者提供基于Git的在线代码托管服务，包括代码克隆/下载/提交/推送/比较/合并/分支/代码评审等功能。
Gitee	又称为“码云”，是开源中国社区团队推出的在线代码托管平台，为开发者提供云端软件开发协作能力。码云支持个人、团队、企业实现代码托管、项目管理和协作开发。

开源社区

- 开源社区对于推动开源软件发展起着重要作用

- ✓ 汇聚志同道合的开发者并进行交流和沟通的重要途径
- ✓ 通过pull-request机制允许对项目感兴趣的开发者提出其发现的bug或者新增需求，实现开放和社会化开发
- ✓ 汇聚大量开源项目并支持开源项目的检索、使用和推广

- 知名开源社区

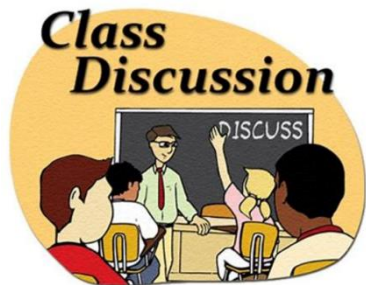
- ✓ 国际：Apache、GitHub、GitLab、Sourceforge、kernel.org、OpenSource、OpenOffice、Mozilla等
- ✓ 国内：木兰社区、开源中国、Linux中国、LUPA、共创软件联盟、Openharmony、OpenEuler、OpenGauss等

- 开源软件并不意味着可以完全自由地使用其源代码，开源项目一般都有对应的许可证协议，其中对开发者使用开源软件的行为进行了约束

常用的开源软件许可证协议

许可证类型	触发开源义务的条件	开源要求和范围	典型软件
BSD类 (Apache/ BSD/MIT等)	无	无	Tomcat, OpenSSL
MPL类 (MPL/EPL等)	产品集成使用该软件, 并 对外分发或销售; 产品对 该软件进行了修改。	若无修改, 则无需开源; 若对其进行了修改, 需 将修改的部分开源。	FireFox, Eclipse
GPL类: GPL	产品集成使用该软件, 并 对外分发或销售。	软件本身须开源。具有传染性, 与其有链接关系 的代码都必须以GPL许可对外开源, 即与该软件 在同一进程中运行的代码都必须开源。	linux kernel, Busybox
GPL类: LGPL	产品集成使用该软件, 并 对外分发或销售。	软件本身须开源。具有传染性, 与其静态链接部 分的代码也必须以LGPL许可开源; 动态链接则 不被传染。	Hibernate, glibc
GPL类: AGPL	产品集成使用该软件。	在GPL上增加了一条限制: 即便不对外分发, 只 要在网络服务器上使用AGPL软件提供网络服务, 就需要履行相关开源义务。	Berkeley_D B
GPL类: SSPL	产品将该软件做为服务或 利用该软件的能力向公司 外的第三方提供服务。	软件本身须开源。具有传染性, 使用开源软件相 关的服务组件也要开源。相对AGPL, 任何试图 将开源软件作为服务加以利用的组件, 都必须开 放用于提供此类服务的软件的源代码。	MongoDB

课堂讨论：开源社区及开源项目



课堂讨论：你是否了解、使用或参加过开源项目？如何理解开源项目的运作机制及其价值？

三方库管理平台

- 如果一个开源软件属于三方库时，开发人员一般会选择将其成熟版本发布到三方库管理平台
- 其他软件开发者可以在项目配置文件中引入该三方库的特定版本并使用所引入三方库的API
- 面向Java语言三方库的Maven仓库
 - ✓ 当前使用最广泛的三方库管理平台之一，基于简单文件系统存储集中化管理Java API资源（Jar包）
 - ✓ 提供客户端为开发者提供资源检索与构建服务
 - ✓ 软件项目可以通过包管理器引用资源并加入项目进行构建
- 其他主流三方库管理平台还包括面向C/C++语言三方库的JFrog ConanCenter（与该平台相对应的包管理器是Conan）

组件级复用

• 软件开发库复用

- ✓ 实现通用功能且经过良好封装的软件组件经过大量复用后，不仅可以提高开发效率、节约开发成本，而且其质量较高
- ✓ 一般以开发库的形式提供，有着明确定义的API及相关的文档描述，应用开发者可以方便地进行复用和集成

• 在线服务复用

- ✓ 有些通用功能与提供方的内部数据和业务密切相关，只能以在线服务的方式提供
- ✓ 应用开发人员通过远程服务调用的方式进行使用

• 接口描述规范

- ✓ 围绕各种类型的软件组件形成的接口描述规范使得组件接口可以被规范描述
- ✓ 进一步还可以基于接口描述实现客户端桩（stub）和服务端骨架（skeleton）代码的自动生成

软件开发库复用

一般都是围绕API进行复用，要求开发人员将特定版本的开发库下载到本地参与项目的构建并将其作为依赖库进行管理

addExact

```
public static int addExact(int x,  
                           int y)
```

访问方式: 静态方法

Returns the sum of its arguments, throwing an exception if the result overflows an int.

功能描述

Parameters:

x - the first value

y - the second value

参数列表

Returns:

the result

返回值

Throws:

`ArithmeticException` - if the result overflows an int

抛出异常

Since:

1.8

使用范围

Java SE 1.8中Math类下的
addExact方法的描述文档

```
int sum = 0;  
  
try {  
    sum = Math.addExact(x, y);  
}  
  
catch (ArithmeticException e) {  
    System.err.println(e.getMessage());  
}
```

在客户端程序中，按照API使用规范正确调用方法，正确传入实参，并使用适合的处理机制来处理所抛出的溢出异常

在线服务复用

- 通过互联网提供的在线服务也是复用对象
 - ✓ 这些服务往往与服务提供者自身的业务属性密切相关
 - ✓ 例如，天气预报查询、航班信息查询、在线支付服务等
- 企业内以在线服务方式沉淀和积累通用业务或技术服务
 - ✓ “中台”的概念就在很大程度上与这种通用业务或技术服务的积累和复用密切相关
 - ✓ 例如，电商企业的中台服务可以包括优惠券管理、配送服务等业务服务以及内容存储、短信和邮件发送等技术服务
- 复用在线服务时无需下载到本地参与项目构建，可以直接进行在线服务的调用
- 广泛使用的接口风格是RESTful API

RESTful API

- 以资源为核心

- ✓ 一般是一个业务对象（例如图书、图书副本、读者）或者业务对象的集合（例如所有新到图书）
- ✓ 可以使用URL来作为唯一标识符并进行引用

- 通过HTTP动词来操作资源

- ✓ GET（请求读取资源内容）、POST（请求创建新资源）、PUT（请求更新资源）、DELETE（请求删除资源）
- ✓ 通过资源URL与HTTP动词的组合来定义RESTful API

- 一般使用HTTP协议作为进程间通信机制，并使用XML或JSON格式来表示资源信息

- 应用开发者需要编写客户端代码实现远程调用，例如使用Apache提供的HttpClient

RESTful API接口说明

URI表示RESTful API的调用入口，其中url表示服务地址
“orders”表示资源(即订单)

URI	接口说明	安全性	幂等性
GET http://url/orders/{id}	返回订单编号为id的 订单内容	安全	幂等
DELETE http://url/orders/{id}	删除订单编号为id的 订单	非安全	幂等
PUT http://url/orders/{id}	更新订单编号为id的 订单	非安全	幂等
POST http://url/orders/	增加一个订单	非安全	非幂等

如果一个API的执行对资源不进行修改则该API是安全的

如果一个API执行一次和多次所产生的结果是一样的，则该API是幂等的

接口描述规范

- RESTful API在网络化软件以及以微服务架构为特征的云原生软件中有着大量应用，因此接口规范化描述也成为一个问题
 - ✓ 开发人员不仅可以了解API定义从而确保正确调用，而且可以利用相关工具自动生成客户端桩（stub）和服务端骨架（skeleton）代码
 - ✓ 测试人员可以基于API定义进行相应的测试
- 目前使用最广泛的描述规范和框架是Swagger
 - ✓ 目前世界上最大的面向OpenAPI规范的API开发者框架
 - ✓ 支持从API设计、描述、测试和部署的整个生存周期过程
 - ✓ 也是一套接口描述规范，只要按照规范定义API及相关的信息就可以通过工具生成接口文档、客户端和服务端代码、在线接口调试页面等
 - ✓ 基于YAML格式的接口描述指定了与接口相关的关键属性

Swagger接口描述规范中的关键属性

属性名	描述
basePath	资源的基础路径
path	资源标识，对资源的完整的访问路径为 basePath + path
GET/DELETE/PUT /POST	对资源的操作类型
produces/consumes	资源产出和消费的制品的表达方式，例如XML或JSON
parameters	参数定义，包括参数名称、描述、模式
responses	响应结果定义，一般为HTTP规范中的响应返回码， 例如1xx表示临时响应，2xx表示成功响应，3xx表示转发，4xx表示客户端错误，5xx表示服务端错误

框架级复用

• 框架级复用

- ✓ 可以让开发人员获得软件应用或其一部分的整体基础框架
- ✓ 按照框架规范进行定制和扩展后获得的软件应用可以在很大程度上获得框架所带来的设计灵活、关注点分离、易于维护和扩展等方面的优势
- ✓ 开发框架一般还内置了一些通用的基础功能实现，避免在软件应用开发时重复地实现

• 应用开发可以关注于特定应用的业务实现

- 当前主流的软件应用（web应用、移动应用等）一般都由前端界面和后端服务构成
- 广泛采用MVC模式来设计应用结构，将界面与业务逻辑分离，提高可扩展性和可维护性

MVC模式

查询数据并将
数据进行展现

截获用户请求并
操纵数据处理

封装数据及
业务逻辑



用户首先通过视图向控制器发送请求

该请求被控制器识别后根据预定义规则交由相应的模型业务逻辑进行处理

业务逻辑的处理结果体现为对模型内数据的更新，并将更新结果返回给控制器

控制器随后选择需要展示的视图并将模型数据在视图中进行渲染展示

Spring MVC请求处理流程

2. DispatcherServlet对请求URL进行解析, 基于所配置的HandlerMapping映射规则找到对应Controller并返回DispatcherServlet

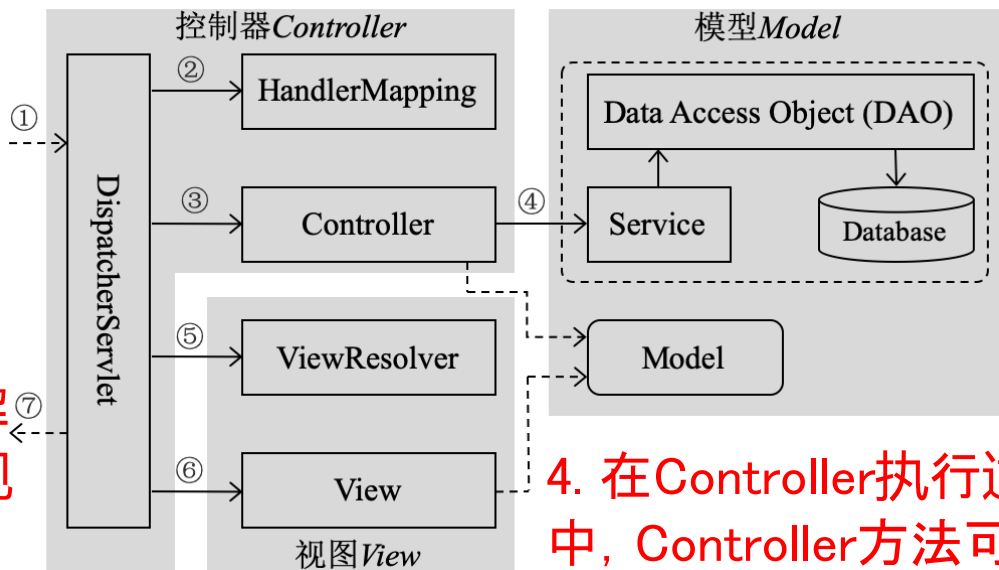
1. 用户通过浏览器向服务器发送请求, 被Spring MVC的前端控制器DispatcherServlet捕获

5. DispatcherServlet调用ViewResolver对视图名称进行解析, 得到要被渲染展示的具体视图

6. DispatcherServlet将模型中的数据填充至视图中的相应元素, 从而渲染视图

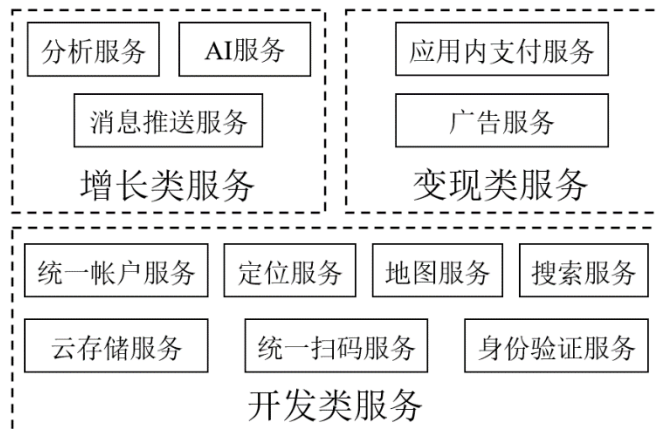
3. DispatcherServlet调用请求所映射的Controller

4. 在Controller执行过程中, Controller方法可能会调用后端复杂的业务逻辑以刷新模型



平台级复用

- 基于平台的小程序应用开发是一种高度基于复用的软件开发形式，提供了一系列支持
 - ✓ 开发和部署环境，包括应用框架、平台服务、一站式的开发工具服务，帮助开发者实现敏捷开发及持续集成等能力
 - ✓ 计算、存储、网络等资源以及相关的技术服务（借助于所依托的云计算系统）
 - ✓ 用户资源和访问流量



开发类服务主要面向开发者，为其提供统一的平台级基础支撑能力以搭建各具特色的小程序应用

增长类服务旨在为应用开发者提供面向应用维护、运营和优化的高级别服务能力

变现类服务为应用开发者提供基于平台的收入渠道

本章小结

- 各种层次、各种形态的软件复用已经成为一种必不可少的软件开发手段
- 组件级复用：获得应用中的一些局部功能实现
- 框架级复用
 - ✓ 可以让开发人员获得软件应用或其一部分的整体基础框架
 - ✓ 在此基础上按照框架规范进行定制和扩展
 - ✓ 基于框架开发的应用可以在很大程度上获得框架所带来的设计灵活、关注点分离、易于维护和扩展等方面的优势
- 平台级复用
 - ✓ 微信、支付宝、WeLink等流行的互联网开放平台
 - ✓ 基于平台的应用开发是一种高度基于复用的软件开发形式
 - ✓ 平台提供开发及部署环境、计算和网络等资源，以及用户资源和访问流量

COMP130015.02

软件工程

End

6. 软件复用