**中国矿业大学计算机学院**

**2014 级本科生课程报告**

课程名称 软件架构设计

报告时间 2017年5月12日

学生姓名 赵勤博

学 号 08143427

专 业 计算机科学与技术

任课教师 袁冠

任课教师评语

任课教师评语（①对课程基础理论的掌握；②对课程知识应用能力的评价；③对课程报告相关实验、作品、软件等成果的评价；④课程学习态度和上课纪律；⑤课程成果和报告工作量；⑥总体评价和成绩；⑦存在问题等）：

成 绩： 任课教师签字：

年 月 日

摘 要

软件体系结构是软件工程领域中一个重要的研究课题，是大型软件开发中的关键技术。经过十多年的发展，软件体系结构也超出了传统的对软件设计阶段的支持，逐渐扩展到整个软件生命周期。通过对现有软件体系结构和风格和框架、中间件的研究，应用于软件设计中。设计中使用了SSM框架，很大程度上提高了开发效率及系统的可靠性和易用性。介绍了软件中的总体架构、安全架构、权限设计和并发设计。

关键字： 软件体系结构；设计风格；框架；中间件；SOA

ABSTRACT

Software architecture is an important research topic in the field of software engineering, and it is the key technology in large-scale software development. After more than 10 years of development, the software architecture is beyond the traditional support for the software design phase, and gradually extended to the entire software life cycle. Through the existing software architecture and style and framework, middleware research, applied to software design. The design uses the SSM framework, which greatly improves the development efficiency and system reliability and ease of use. Introduced the overall architecture of the software, security architecture, authority design and concurrent design.

**KEYWORDS:** software architecture; design style; framework; middleware; SOA

目 录

[1 概述 1](#_Toc482297787)

[1.1引言 1](#_Toc482297788)

[1.2软件体系结构性质 1](#_Toc482297789)

[1.3软件体系结构的研究意义 3](#_Toc482297790)

[1.4软件体系结构的研究目标 3](#_Toc482297791)

[2 设计风格、模式与应用框架 4](#_Toc482297792)

[2.1软件体系结构风格 4](#_Toc482297793)

[2.2软件设计模式 4](#_Toc482297794)

[2.3软件应用框架 5](#_Toc482297795)

[2.4设计模式与软件框架 5](#_Toc482297796)

[3 中间件 6](#_Toc482297797)

[3.1中间件定义 6](#_Toc482297798)

[3.2中间件的分类 6](#_Toc482297799)

[3.3Tomcat中间件 7](#_Toc482297800)

[3.4Mybatis中间件 7](#_Toc482297801)

[4 面向服务的体系结构 8](#_Toc482297802)

[4.1 SOA概述 8](#_Toc482297803)

[4.2利用SOA架构开发的优点 8](#_Toc482297804)

[4.3Web Service 9](#_Toc482297805)

[4.4Web Service协议集 10](#_Toc482297806)

[4.5WebService主要协议 10](#_Toc482297807)

[4.5.1XML+XSD 10](#_Toc482297808)

[4.6.1SOAP 11](#_Toc482297809)

[4.6.2WSDL 11](#_Toc482297810)

[5 云库课文件管理系统 12](#_Toc482297811)

[5.1总体架构 12](#_Toc482297812)

[5.1.1 B/S结构 12](#_Toc482297813)

[5.1.2四层架构 12](#_Toc482297814)

[5.1.3RESTful风格架构 13](#_Toc482297815)

[5.2应用框架 14](#_Toc482297816)

[5.2.1SSM框架简介 14](#_Toc482297817)

[5.2.2SSM框架特点 14](#_Toc482297818)

[5.3权限架构 15](#_Toc482297819)

[5.3.1使用的权限架构 15](#_Toc482297820)

[5.3.2其它权限架构 15](#_Toc482297821)

[5.4并发设计 16](#_Toc482297822)

[5.4.1数据库索引与优化 16](#_Toc482297823)

[5.4.2 Mybatis数据库连接池 17](#_Toc482297824)

[5.4.3其它高并发策略 18](#_Toc482297825)

[5.5安全架构 18](#_Toc482297826)

[5.5.1使用的安全框架 18](#_Toc482297827)

[5.5.2数据库关键字加密 19](#_Toc482297828)

[5.5.3其它安全框架 19](#_Toc482297829)

[参考文献 20](#_Toc482297830)

# 1 概述

## 1.1引言

随着软件规模和复杂度不断地扩大和增加，软件开发的成败在很大程度上取决于软件体系结构的设计。将那些公认的、被多次成功使用的软件体系结构称为软件体系结构风格，比如REST(Representational State Transfer)风格、多级正交风格、SOA(Service-Oriented Architecture) 风格、RIA(RichInternet Application)风格等等。软件体系结构可以看成是软件体系结构风格的实例，软件体系结构风格是描述某一特定应用领域中系统组织方式的惯用设计模式，是一组约束的集合。面向对象，面向组件，面向服务已是当今主流的软件开发思想，同时以演化和增量为基础的迭代开发模式也是主流的开发过程。尽管如此，如果软件体系结构设计不当，就可能引起系统重构，甚至整个体系结构重新设计，大量实践也证明了：一个成功的软件系统往往都有一个好的软件体系结构。

## 1.2软件体系结构性质

若干软件体系结构学者提出了自己的概念与定义，我们将介绍几个具有代表性的定义，并分析其共同点。然后，根据我们对软件体系结构的理解，给出能够反映软件体系结构基本特征的定义。

定义 1. Garlan & Shaw 模型:

SA={components， connectors， constrains}.

构件(component)可以是一组代码，如程序的模块；也可以是一个独立的程序，如数据库的 SQL 服务器。连接器(connector)表示构件之间的相互作用。它可以是过程调用、管道、远程过程调用等。一个软件体系结构还包括某些限制(constrain)。该模型视角是程序设计语言，构件主要是代码模块。

定义 2. Perry & Wolf 模型:

SA={elements， form， rational}.

软件体系结构是由一组元素(elements)构成。这组元素分成 3 类:处理元素(processing elements)、数据元素(data elements)和连接元素(connecting elements)。软件体系结构形式(form)是由专有特性(properties)和关系(relationship)组成。专有特性用于限制软件体系结构元素的选择，关系用于限制软件体系结构元素组合的拓扑结构。而在多个体系结构方案中选择合适的体系结构方案往往基于一组准则(rational)。

定义 3. CFRP 模型:

SA={elements， interfaces， connections， connection semantics}.

软件系统由一组元素构成(elements)。这组元素分成处理元素和数据元素。每个元素有一个接口(interface)，一组元素的互连(connection)构成系统的拓扑。元素互连的语义是:静态互连语义(如数据元素的互连)，描述动态连接的信息转换的协议(如过程调用，管道等)。

定义 4. Vestal 模型:

SA={component， idioms/styles， common patterns of interaction}.

软件由构件(component)组成，构件之间通过通用的互操作模式相连。体系结构风格(style)描述了一种通用的设计模式，可满足特定系列的应用需求。

定义 5. IEEE 610。12-1990 软件工程标准词汇中的定义:

Architecture={component， connector， environment， principle}。

体系结构是以构件、构件之间的关系、构件与环境之间的关系为内容的某一系统的基本组织结构，以及指导上述内容设计与演化的原理。

定义 6. Boehm 模型:

SA={components， connections， constraints， stakeholders’ needs， rationale}.

软件体系结构包含系统构件、连接件、约束的集合；反应不同人员需求的集合；以及能够展示由构件、连接件和约束所定义的系统在实现时如何满足系统不同人员需求的原理的集合。

比较上述各种体系结构的定义可以发现:定义 1和定义 3 都强调了体系结构是由构件、连接件及其约束(或连接语义)，即从构造的角度来审视软件体系结构。而定义 2 和定义 4 侧重于从体系结构风格、模式和规则等角度来考虑软件体系结构，采用的是一种俯瞰的视角。定义 5 不仅强调了体系结构系统的基本构成，同时还强调了体系结构的环境即与外界的交互。定义 6 则强调了软件体系结构是一组概念以及关于软件系统结构的设计决策，用来使待开发的系统在体系结构上满足重要的功能与质量需求。尽管各种定义都从不同的角度关注软件体系结构，但其核心内容都是软件系统的结构，并且都涵盖了如下一些实体:构件、构件之间的交互关系、限制、构件和连接件构成的拓扑结构、设计原则与指导方针。

根据我们对软件体系结构的理解，将软件体系结构定义为“在软件密集的大规模系统或具有类似需求和结构的软件产品线的开发中，必须从一个较高的层次来考虑组成系统的构件、构件之间的交互，以及由构件与构件交互形成的拓扑结构，这些要素应该满足一定的限制，遵循一定的设计规则，能够在一定的环境下进行演化。

而且，软件体系结构应能反映系统开发中具有重要影响的设计决策，便于各种人员的交流， 反映多种关注，据此开发的系统能完成系统既定的功能和性能需求。”

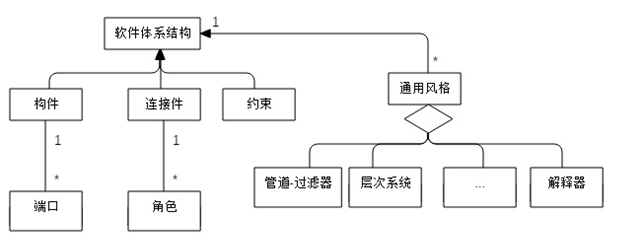


图1.1 软件体系结构

## 1.3软件体系结构的研究意义

自NATO于 1968 年提出软件工程概念以来，软件工程界已经提出了一系列的理论、方法、语言和工具，解决了软件开发过程中的若干问题。但是，软件固有的复杂性、易变性和不可见性，使得软件开发周期长、代价高和质量低的问题依然存在。大量实践统计表明：大系统软件开发中70％的错误是由需求和软件设计阶段引入的；而且错误在系统中存在的时间愈长则愈难发现，解决这些错误的代价也愈高。

为了提高软件需求和软件设计的质量，软件工程界提出了需求分析工程技术和各种软件建模技术。但是在需求与设计之间仍存在一条很难逾越的鸿沟，即缺乏能够反映做决策的中间过程，从而很难有效地将需求转换为相应的设计。为此，软件体系结构概念应运而生，并试图在软件需求与软件设计之间架起一座桥梁，着重解决软件系统的结构和需求向实现平坦地过渡的问题。

从机器语言、汇编语言、过程式程序设计语言、面向对象程序设计语言、形式化(半形式化)规格说明语言(如体系结构描述语言)发展过程中，可以发现：计算机语言越来越适合于开发人员的思维活动模型，代码复用的级别也在不断地提升，如图1.2所示。体系结构技术的研究，使软件复用从代码复用发展到设计复用和过程复用。

鉴于软件体系结构的重要性，D.E.Perry 将软件体系结构视为软件开发中第1类重要的设计对象[1]，而 Barry Boehm 明确指出：“在没有设计出体系结构及其规则时，那么整个项目不能继续下去，而且体系结构应该看做是软件开发中可交付的中间产品”[2] 。

由此可见，体系结构在软件开发中为不同的人员提供了共同交流的语言，体现并尝试了系统早期的设计决策，并作为系统设计的抽象，为实现框架和构件的共享与复用、基于体系结构的软件开发提供了有力的支持。

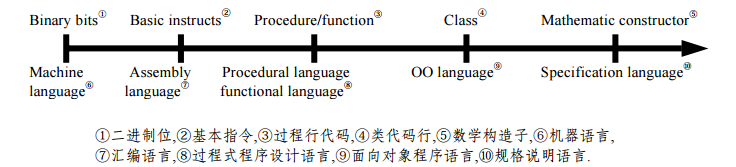


图1.2 计算机语言的变迁与复用的提升图示

## 1.4软件体系结构的研究目标

在基于构件和体系结构的软件开发方法下，程序开发模式也相应地发生了根本变化。软件开发不再是“算法+数据结构”，而是“构件开发+基于体系结构的构件组装”。那么，如何描述、表示、设计体系结构?如何验证一个体系结构是否符合期望的系统需求?如何基于体系结构开发实例化的软件系统?将是基于构件和体系结构的软件开发方法所必须研究和解决的核心问题。归纳现有体系结构的研究活动，主要包括如下几个方面：

(1) 体系结构理论模型的研究；比如：多视图表示

(2) 体系结构描述研究，主要研究体系结构描述语言及其支持环境、体系结构描述规范；

(3) 体系结构设计研究：包括体系结构设计方法、体系结构风格、体系结构设计空间等内容；

(4) 体系结构分析与验证：研究如何将软件的非功能特性转化为体系结构的需求，如何分析体系结构满足期望的需求的属性，对体系结构的语法、语义、类型失配等进行检查与验证的研究；

(5) 体系结构演化与复用研究：研究产品线中软件体系结构演进的理论与方法，从已有文档、系统设计和代码中逆向提取软件体系结构、体系结构复用等；

(6) 动态体系结构研究：研究软件系统由于特殊需要必须在连续运营情况下的体系结构变化与支撑平台；

(7) 基于体系结构的软件开发：研究引入体系结构后的软件开发过程、基于体系结构开发与中间技术集成、基于体系结构的程序框架自动生成技术等。

(8) 基于软件体系结构的部署研究。部署方案评估以及生成最优部署方案等等。

# 2 设计风格、模式与应用框架

## 2.1软件体系结构风格

通过对软件体系结构风格的研究和实践促进了对设计和过程的重用，经过实践证实的解决方案可以可靠地用于解决新的问题。基于软件体系结构风格构建的系统可使别的设计者很容易地理解系统的体系结构。例如：如果某人把系统描述为 B/S 模式，则不必给出设计细节，我们立刻就会明白系统是如何组织和工作的。

随着软件研发技术的不断进步，近些年接连总结出了几种新型的软件体系结构风格，如：正交软件体系结构风格、REST 软件体系结构风格、面向服务的SOA 软件体系结构风格、基于 WEB 的插件体系结构风格、RIA 富客户端风格等等，关于软件体系结构风格的研究还处于起步阶段。

## 2.2软件设计模式

设计模式是面向对象技术的“刀刃”部分。设计模式的概念最早来源于建筑学，由建筑师 Alexander 提出来的 ，它是记录、提炼存在于软件开发人员头脑中或文档中的一些反复出现的共性问题及其经过多次验证的成功解， 它表达了在特定上下文情形下产生的 、反复出现的典型性问题以及相应的一整套解决方案之间的关系。设计模式关注的是特定设计问题及其解决方案，它描述了如何利用面向对象的基本概念和机制来解决可扩展的软件设计中经常出现的问题，并针对设计问题给出可复用的解决方案，而这个解决方案是对反复出现的设计结构进行识别和抽象得到的。同时，每个模式都伴有定义的语境和强度，语境解释了模式的适用情况，强度是语境中的元素。如果问题的环境与模式的语境和强度相匹配，该模式便可应用。

每一个设计模式都系统地命名、解释和评价了面向对象系统中的一个重要的和重复出现的设计。这样，只要搞清楚这些设计模式，就可以完全或者很大程度上吸收了那些蕴含在模式中的宝贵经验， 对面向对象的系统能够有更完善的了解。更为重要的是，这些模式可直接用来指导面向对象系统中至关重要的对象建模问题。只要问题有相同的背景，就可以直接套用这些设计模式，从而大大减轻软件开发人员的工作量。

关于软件设计模式，目前的研究方向主要有设计模式与其他面向对象设计方法(如特定领域的框架)的关系，它们各自的优劣和适应范围。除此以外，人们还在各个方面总结设计模式(如通信领域等)，以及研究如何让设计模式的使用更加自动化。

## 2.3软件应用框架

《设计模式》对框架有如下定义：“一个框架就是一组相互协作的类，对于特定的一类软件，框架构成了一种可重用的设计”。这个定义虽然主要着眼于面向对象的软件开发，但已经基本上给出了这个词的核心含义：框架是软件系统的设计、开发过程中的一个概念，它强调对己完成的设计、代码的重复使用，并且，一个框架主要适用于实现某一特定类型的软件系统。不同领域的软件项目有着不同的框架类型。框架的作用在于：由于提取了特定领域软件的共性部分，因此在此领域内新项目的开发过程中代码不需要从头编写，只需要在框架的基础上进行一些开发和调整便可满足要求；对于开发过程而言，这样做会提高软件的质量，降低成本，缩短开发时间，使开发越做越轻松，效益越做越好，形成一种良性循环。

框架不是工具包/类库/API。目前流行的很多框架中，包括了大量的类库和API，但是调用API并不就是在使用框架开发。仅仅使用API时，开发者完成系统的主体部分，并不时地调用类库实现特定任务。而框架构成了通用的、具有一般性的系统主体部分，“二次开发者”只是像做填空题一样，根据具体业务，完成特定应用系统中特殊的部分。

面向对象框架是设计复用和代码复用相结合的一种最为重要的形式。说它是代码复用的一种形式，是因为它是由一种程序设计语言写成，是一个应用的半成品(应用可以通过扩展框架来实现所需的功能)，且常常包括具体的立即可用的构件与子类。但面向对象框架更强调设计复用，它定义了应用的整体结构，框架中各个构件的主要责任和构件之间的协作。因此也就记录了其应用领域中共同的设计决策。

框架强调的是软件的设计重用性和系统的可扩充性，以缩短大型应用软件系统的开发周期，提高开发质量。与传统的基于类库的面向对象重用技术比较，框架更注重于面向专业领域的软件重用。框架具有领域相关性，构件根据框架进行复合而生成可运行的系统。框架的粒度越大，其中包含的领域知识就更加完整。

框架涉及到的概念和技术方法较多，包括领域工程、设计模式、体系结构、类库、构件技术等。一方面，当与模式、类库和构件结合在一起使用时，框架能明显提高软件质量，降低应用开发的难度；另一方面，进行领域工程，研究发掘出应用领域可复用的设计模式，基于发掘出的设计模式开发应用框架是一条完整解决框架复用问题的有效途径。

## 2.4设计模式与软件框架

框架是一个应用程序，但是它却是可复用的。软件开发人员通过对框架的定制去生成客户所需求所满足的应用程序。从一般用户使用的角度去看的话，这些用户通过定制一般的框架生成了满足他们所需要的软件。虽然过程是比较简单的设计，但是框架的内部复杂结构需要对说使用的用户处于隐藏状态。从软件设计重用性的角度去看的话，框架很好地解释了某个领域里面的设计概念，可以更加的适应此领域内的用户的需求。从达到软件重用性的角度去看的话，软件设计人员可使用继承或代理的方法，通过这样使框架中的抽象类达到软件复用的实现目的。其主要优点是：可扩张性；模块化；可复用性。

如今，很多人经常把软件框架和设计模式混淆。实际上框架与设计模式大多是来源于对解决问题的一般应用，来源于很多应用中所获得的实践经验总结。但是软件框架与设计模式却是两个不同的概念，当然有不同的含义。

软件框架应该是设计模式的特殊化、实例化，它是针对于某个特定的领域中使用的。设计模式表示的是软件开发人员在开发的过程中在特定的场景下去解决重复发生问题的方案。每个设计模式都集中在某个特定的面向对象设计问题，描述了设计人员该在什么情况下使用它，以及使用了以后的效果。一个系统如果使用了设计模式的框架，则比不使用设计模式的框架更可能取得高层次上的软件设计复用。设计模式与软件框架主要的不同在于:

(1)设计模式相比于框架，更为抽象。因为软件框架能用代码表示出来，而设计模式只能是实例化后才可以表示为代码。框架的魅力在于它完全可以用程序的设计语言来编写。不但可以用来被学习，还可以被直接执行和重用。如果从某种意义上来说的话，框架是个物理实体，而设计模式则是个逻辑实体。框架可以被人看成是一个设计模式解决方案的物理实现，而模式则可以看是指导如何去实现这些方案的逻辑实现。

(2)设计模式是相比与框架，具有更为小的结构元素。通常情况下，比较典型的框架包含一个或者多个设计模式，而反过来却不一定是这样。

(3)框架相比于设计模式，则更加的特殊化、实例化。因为框架往往是针对一些特定的应用领域，而设计模式则可以在任何应用被使用。

# 3 中间件

## 3.1中间件定义

IDC给出的一个定义：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。

## 3.2中间件的分类

中间件的产品种类很多，综合其特点来看，大致划分为以下七种：

(1)数据库中间件 ( DM , Database Middleware )

(2)远程过程调用 中间件 ( RPC , Remote Procedure Call )

(3)面向消息中间件 ( MOM , Message Oriented Middleware)

(4)对象请求代理 ( ORB , Object Request Brokers )

(5)事务处理管程 ( TPM , Transaction Processing Monitor )

(6)专用中间件 ( PM , Proprietary Middleware )

(7)WEB服务中间件

## 3.3Tomcat中间件

Tomcat是由Apache软件基金会下属的Jakarta项目开发的一个Servlet容器，按照Sun Microsystems提供的技术规范，实现了对Servlet和JavaServer Page（JSP）的支持，并提供了作为Web服务器的一些特有功能，如Tomcat管理和控制平台、安全域管理和Tomcat阀等。由于Tomcat本身也内含了一个HTTP服务器，它也可以被视作一个单独的Web服务器。

Tomcat作为运行Servlet的容器，主要目标就是接收和解析客户发送来的请求。并且将请求发送给Servlet。最后将Servlet处理结果传送给客户。

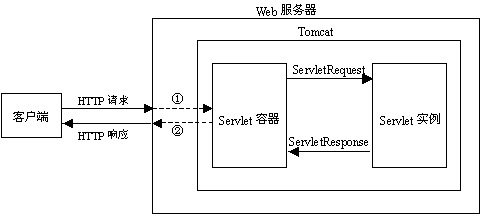


图3.1 Tomcat工作原理

## 3.4Mybatis中间件

MyBatis 是支持定制化 SQL、存储过程以及高级映射的数据库中间件。MyBatis 避免了几乎所有的 JDBC 代码和手动设置参数以及获取结果集。MyBatis 可以对配置和原生Map使用简单的 XML 或注解，将接口和 Java 的 POJOs(Plain Old Java Objects,普通的 Java对象)映射成数据库中的记录。

MyBatis应用程序根据XML配置文件创建SqlSessionFactory，SqlSessionFactory在根据配置，配置来源于两个地方，一处是配置文件，一处是Java代码的注解，获取一个SqlSession。SqlSession包含了执行sql所需要的所有方法，可以通过SqlSession实例直接运行映射的sql语句，完成对数据的增删改查和事务提交等，用完之后关闭SqlSession。

Mybatis实现了DAO接口与xml映射文件的绑定，自动生成接口的具体实现，使用起来变得更加省事和方便。

# 4 面向服务的体系结构

## 4.1 SOA概述

面向服务架构(SOA)以其独特的优势越来越受到企业的重视，它可以根据需求通过网络对松散耦合的粗粒度应用组件进行分布式部署、组合和使用。服务层是SOA的基础，可以直接被应用调用，从而有效控制系统中与软件代理交互的人为依赖性。SOA的开发方法一般主要有开源的dubbo、dubbox、mule、wso2、cxf，以及付费的Oracle soa、ibm soa等。

SOA是一种粗粒度、松耦合服务架构，服务之间通过简单、精确定义接口进行通讯，不涉及底层编程接口和通讯模型。SOA可以看作是B/S模型、XML（标准通用标记语言的子集）,Web Service技术之后的自然延伸。

SOA将能够帮助软件工程师们站在一个新的高度理解企业级架构中的各种组件的开发、部署形式，它将帮助企业系统架构者以更迅速、更可靠、更具重用性架构整个业务系统。较之以往，以SOA架构的系统能够更加从容地面对业务的急剧变化。

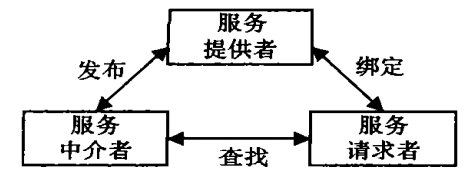


图4.1 SOA模型

## 4.2利用SOA架构开发的优点

(1)更易维护

　　业务服务提供者和业务服务使用者的松散耦合关系及对开放标准的采用确保了该特性的实现。建立在以 SOA基础上的信息系统，当需求发生变化的时候，不需要修改提供业务服务的接口，只需要调整业务服务流程或者修改操作即可，整个应用系统也更容易被维护。

(2)更高的可用性

　　该特点是在于服务提供者和服务使用者的松散耦合关系上得以发挥与体现。使用者无须了解提供者的具休实现细节。

(3)更好的伸缩性

　　依靠业务服务设计、开发和部署等所采用的架构模型实现伸缩性。使得服务提供者可以互相彼此独立地进行调整，以满足新的服务需求。

## 4.3Web Service

WebService是一种跨编程语言和跨操作系统平台的远程调用技术。Web service是一个平台独立的，低耦合的，自包含的、基于可编程的web的应用程序，可使用开放的XML（标准通用标记语言下的一个子集）标准来描述、发布、发现、协调和配置这些应用程序，用于开发分布式的互操作的应用程序。

Web Service技术，能使得运行在不同机器上的不同应用无须借助附加的、专门的第三方软件或硬件，就可相互交换数据或集成。依据Web Service规范实施的应用之间， 无论它们所使用的语言、平台或内部协议是什么，都可以相互交换数据。Web Service是自描述、 自包含的可用网络模块， 可以执行具体的业务功能。Web Service也很容易部署，因为它们基于一些常规的产业标准以及已有的一些技术，诸如标准通用标记语言下的子集XML、HTTP。Web Service减少了应用接口的花费。Web Service为整个企业甚至多个组织之间的业务流程的集成提供了一个通用机制。

Web服务有两层含义：

一是指封装成单个实体并发布到网络上的功能集合体；二是指功能集合体被调用后所提供的服务。简单地讲，Web服务是一个URL资源，客户端可以通过编程方式请求得到它的服务，而不需要知道所请求的服务是怎样实现的，这一点与传统的分布式组件对象模型不同。

Web服务的体系结构是基于Web服务提供者、Web服务请求者、Web服务中介者三个角色和发布、发现、绑定三个动作构建的。简单地说，Web服务提供者就是Web服务的拥有者，它耐心等待为其他服务和用户提供自己已有的功能；Web服务请求者就是Web服务功能的使用者，它利用SOAP消息向Web服务提供者发送请求以获得服务；Web服务中介者的作用是把一个Web服务请求者与合适的Web服务提供者联系在一起，它充当管理者的角色，一般是UDDI。这三个角色是根据逻辑关系划分的，在实际应用中，角色之间很可能有交叉：一个Web服务既可以是Web服务提供者，也可以是Web服务请求者，或者二者兼而有之。显示了Web服务角色之间的关系:其中，“发布”是为了让用户或其他服务知道某个Web服务的存在和相关信息；“查找（发现）”是为了找到合适的Web服务；“绑定”则是在提供者与请求者之间建立某种联系。

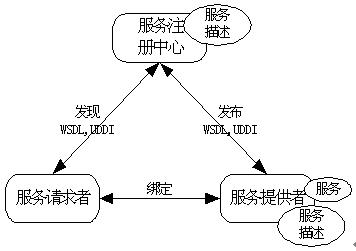


图4.2 Web service的体系结构

实现一个完整的Web服务包括以下步骤：

(1)Web服务提供者设计实现Web服务，并将调试正确后的Web服务通过Web服务中介者发布，并在UDDI注册中心注册； （发布）

(2)Web服务请求者向Web服务中介者请求特定的服务，中介者根据请求查询UDDI注册中心，为请求者寻找满足请求的服务； （发现）

(3)Web服务中介者向Web服务请求者返回满足条件的Web服务描述信息，该描述信息用WSDL写成，各种支持Web服务的机器都能阅读；（发现）

(4)利用从Web服务中介者返回的描述信息生成相应的SOAP消息，发送给Web服务提供者，以实现Web服务的调用；（绑定）

(5) Web服务提供者按SOAP消息执行相应的Web服务，并将服务结果返回给Web服务请求者。（绑定）

## 4.4Web Service协议集

为了实现这个模型，Web Service 使用了一系列协议，这些协议组成了图 4.3所示的协议栈。 首先，服务提供者所能提供的服务接口用 WSDL和 WSF L 描述出来，然后使用 UDDI在服务中介者处将这些服务接口进行注册；服务中介者则使用UDDI注册这些服务接口， 并接受服务请求者的查询；服务请求者通过使用 U DDI 在服务中介者处查询，得到所需的服务接口的描述文件，然后绑定到这些服务，最终完成调用。参与者之间的三种操作都采用 SOAP( Simple Object Access Protocol) 完成。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Routing, Reliability and Transaction |  | Management | Quality  of  Service | Security |
| Workflow | WSFL |
| Service Discovery, Integration | UDDI |
| Service Description | WSDL |
| Messaging | SOAP |
| Transport | HTTP, FTP, SMTP |
| Internet | IPv4, IPv6 |
| Wire / Wireless |  |

图 4.3 Web Service 协议栈

## 4.5WebService主要协议

### 4.5.1XML+XSD

WebService采用HTTP协议传输数据，采用XML格式封装数据（即XML中说明调用远程服务对象的哪个方法，传递的参数是什么，以及服务对象的返回结果是什么）。XML是WebService平台中表示数据的格式。除了易于建立和易于分析外，XML主要的优点在于它既是平台无关的，又是厂商无关的。无关性是比技术优越性更重要的：软件厂商是不会选择一个由竞争对手所发明的技术的。

XML解决了数据表示的问题，但它没有定义一套标准的数据类型，更没有说怎么去扩展这套数据类型。例如，整形数到底代表什么？16位，32位，64位？这些细节对实现互操作性很重要。XML Schema(XSD)就是专门解决这个问题的一套标准。它定义了一套标准的数据类型，并给出了一种语言来扩展这套数据类型。WebService平台就是用XSD来作为其数据类型系统的。当你用某种语言(如.NET或Java,C语言)来构造一个Web service时，为了符合WebService标准，所有你使用的数据类型都必须被转换为XSD类型。你用的工具可能已经自动帮你完成了这个转换

### 4.6.1SOAP

WebService通过HTTP协议发送请求和接收结果时，必须要有一定的格式，并不是说通过Http随便发送一个请求就可以的。发送的请求内容和结果内容都采用XML格式封装，并增加了一些特定的HTTP消息头，以说明HTTP消息的内容格式，这些特定的HTTP消息头和XML内容格式就是SOAP协议。SOAP提供了标准的RPC方法来调用Web Service。

SOAP协议 = HTTP协议 + XML数据格式

SOAP协议定义了SOAP消息的格式，SOAP协议是基于HTTP协议的，SOAP也是基于XML和XSD的，XML是SOAP的数据编码方式。Http是基本协议，而SOAP在基础协议上添加了一系列的格式限制。打个比喻：HTTP就是普通公路，XML就是中间的绿色隔离带和两边的防护栏，SOAP就是普通公路经过加隔离带和防护栏改造过的高速公路。

### 4.6.2WSDL

WebService客户端要调用一个WebService服务，首先要先知道这个服务的地址在哪，以及这个服务里有什么方法可以调用，所以，WebService务器端首先要通过一个WSDL文件来说明自己家里有啥服务可以对外调用，服务是什么（服务中有哪些方法，方法接受的参数是什么，返回值是什么），服务的网络地址用哪个url地址表示，服务通过什么方式来调用。WSDL是WebService的描述。就好比你买了一辆散装自行车，总得附一个产品说明书吧，告诉你怎么把自行车怎么组装起来，各个零件的作用。

WSDL(Web Services Description Language)就是这样一个基于XML的语言，用于描述Web Service及其函数、参数和返回值。它是WebService客户端和服务器端都能理解的标准格式。因为是基于XML的，所以WSDL既是机器可阅读的，又是人可阅读的，这将是一个很大的好处。一些最新的开发工具既能根据你的Web service生成WSDL文档，又能导入WSDL文档，生成调用相应WebService的代理类代码。

WSDL文件保存在Web服务器上，通过一个url地址就可以访问到它。客户端要调用一个WebService服务之前，要知道该服务的WSDL文件的地址。WebService服务提供商可以通过两种方式来暴露它的WSDL文件地址：1.注册到UDDI服务器，以便被人查找；2.直接告诉给客户端调用者。

# 5 云库课文件管理系统

## 5.1总体架构

### 5.1.1 B/S结构

B/S结构（Browser/Server，浏览器/服务器模式），是WEB兴起后的一种网络结构模式，WEB浏览器是客户端最主要的应用软件。这种模式统一了客户端，将系统功能实现的核心部分集中到服务器上，简化了系统的开发、维护和使用。客户机上只要安装一个浏览器（Browser），服务器安装MYSQL等数据库。浏览器通过Web Server 同数据库进行数据交互。

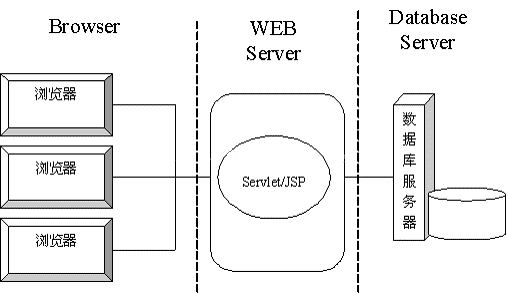


图5.1 B/S结构

### 5.1.2四层架构

分层设计的软件系统，由于把相似功能的类或组件放在同一层里。所以好的分层系统应该具备：层的内部“高内聚”，层与层之间是“松耦合”的。只有符合这样的一个设计原则的软件系统，才能具有可复用性和可扩展性。因此，降低层间“耦合度”就成软件设计的目标，能够设计出“松耦合”的系统，就意味着我们的系统具备可复用性和可扩展性，这样的系统就能够满足的用户不断变化的需求。从设计角度我们可以将一个应用系统（一般是信息系统）分成四层结构如图5.2所示。

(1)表示层

表示层是用户与系统交互的组件集合，用户通过这一层向系统提交请求或发出指令，系统通过这一层接收用户请求或指令，然后，根据请求指令调用服务层，再根据调用的结果，将相应的内容展现到表示层。表示层应该是轻薄的，不应该具有业务逻辑的。我们的系统是一个Web系统，采用的技术有HTML、JSP和Servlet等，也使用Spring-MVC框架技术。这时将表示层称为Web层更准确一点。无论采用什么技术表示层都应该是轻薄的，不应该具有业务逻辑。

(2)服务层

服务层是系统的核心业务处理层，负责接收表示层的指令和数据，根据业务逻辑的需要调用相应的持久层，并将结果返回给表示层。为了降低表示层和服务层的耦合问题，我们会在两层之间引入接口，然后在运行期注入依赖关系。服务层一般使用的技术有会话EJB、消息EJB和JavaBean。

(3)数据持久层

数据持久层用于访问数据库，数据持久层中一般是通过DAO（数据访问对象设计模式）访问数据库的，也是为了降低耦合度，DAO被设计为接口，DAO实现可以是JDBC，也可以使用一些ORM框架如Hibernate、Mybatis等。除了使用DAO以外，我们也可以使用EJB中的实体EJB实现数据持久化。

(4)EIS层

EIS层即企业信息系统层，是系统的数据来源层。它们可以是数据库、文件和其它的系统，多数情况下是指的是数据库。

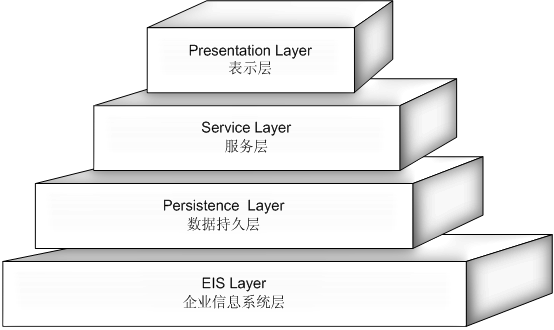


图5.2 四层结构

### 5.1.3RESTful风格架构

RESTful架构风格最初由Roy T. Fielding（HTTP/1.1协议专家组负责人）在其2000年的博士学位论文中提出。HTTP就是该架构风格的一个典型应用。从其诞生之日开始，它就因其可扩展性和简单性受到越来越多的架构师和开发者们的青睐。一方面，随着云计算和移动计算的兴起，许多企业愿意在互联网上共享自己的数据、功能；另一方面，在企业中，RESTful API（也称RESTful Web服务）也逐渐超越SOAP成为实现SOA的重要手段之一。时至今日，RESTful架构风格已成为企业级服务的标配。

REST即Representational State Transfer的缩写，可译为"表现层状态转化”。REST最大的几个特点为：资源、统一接口、URI和无状态。

RESTful的原则：

(1)URL表示资源。

(2)HTTP方法表示操作。

(3)GET只是用来请求操作，GET操作永远都不应该修改服务器的状态。但是这个也要具体情况进行分析，例如一个页面中的计数器，每次访问的时候确实引起了服务器数据的改变，但是在商业上来说，这并不是一个很重要的改变，所以仍然可以接收使用GET的方式来修改数据。

(4)服务应该是无状态的。

(5)在有状态的会话中，服务器可以记录之前的信息。而RESTful风格中是不应该让服务器记录状态的，只有这样服务器才具备可扩展性。当然，我们可以在客户端使用cookie，而且只能用在客户端向服务器发送请求的时候。

(6)服务应当是“幂等”的。

“幂等”表示可以发送消息给服务，然后可以再次毫不费力的发送同样的消息给服务。例如，发送一个“删除第995场比赛”的消息，可以发送一次，也可以连续发送十次，最后的结果都会保持一致。当然，RESTful的GET请求通常是幂等的，因为基本上不会改变服务器的状态。注意：POST请求不能被定义为“幂等”，特别是在创建新资源的时候，一次请求创建一个资源，多次请求会创建多个资源。

(7)拥抱超链接。

(8)服务应当自我说明。

例如 http://www.yunkuke.cn/yunkuke/user/userInfo/995 请求了一个具体的个人信息，但是 http://www.yunkuke.cn/yunkuke/user/userInfo 并没有对任何实体进行请求，因此，应当返回一些介绍信息。

## 5.2应用框架

### 5.2.1SSM框架简介

SSM框架是Spring MVC、Spring、MyBatis的整合。

Spring MVC属于SpringFrameWork的后续产品，已经融合在Spring Web Flow里面。Spring 框架提供了构建 Web 应用程序的全功能 MVC 模块。使用 Spring 可插入的 MVC 架构，从而在使用Spring进行WEB开发时，可以选择使用Spring的SpringMVC框架或集成其他MVC开发框架，如Struts1，Struts2等。

Spring Framework是一个开源的Java／Java EE全功能栈的应用程序框架，以Apache许可证形式发布，也有.NET平台上的移植版本。Spring Framework提供了一个简易的开发方式，这种开发方式，将避免那些可能致使底层代码变得繁杂混乱的大量的属性文件和帮助类。

MyBatis是一个Java持久化框架，它通过XML描述符或注解把对象与存储过程或SQL语句关联起来。MyBatis是在Apache许可证 2.0下分发的自由软件，是iBATIS 3.0的分支版本。其维护团队也包含iBATIS的初创成员。

### 5.2.2SSM框架特点

Spring MVC

(1) 通过策略接口，Spring 框架是高度可配置的，而且包含多种视图技术，例如 JavaServer Pages（JSP）技术、Velocity、Tiles、iText和POI。Spring MVC 框架并不知道使用的视图，所以不会强迫您只使用 JSP 技术。Spring MVC 分离了控制器、模型对象、过滤器以及处理程序对象的角色，这种分离让它们更容易进行定制。

(2) 易于同其它View框架（Tiles等）无缝集成，采用IOC便于测试。

Spring

(1) 轻量—从大小与开销两方面而言Spring都是轻量的。Spring非侵入式的,应用中的对象不依赖于Spring特定类。

(2) 控制反转—Spring通过一种称作控制反转的技术促进了低耦合。不是对象从容器中查找依赖，而是容器在对象

初始化时不等对象请求就主动将依赖传递给它。

(3) 面向切面—Spring提供了面向切面编程的丰富支持，通过分离应用的业务逻辑与系统级服务进行内聚性的开发。

(4) 容器—Spring包含并管理应用对象的配置和生命周期，在这个意义上它是一种容器。

Mybatis

(1) 易于上手和掌握。

(2) sql写在xml里，便于统一管理和优化。

(3) 解除sql与程序代码的耦合。

(4) 提供xml标签，支持编写动态sql。

## 5.3权限架构

### 5.3.1使用的权限架构

用户登录后读取数据库，将用户权限等级存入session中。若用户未达到权限要求，则过滤器不通过，达到控制权限的目的。这样做实现简单，但不适应与关系复杂的系统。

一个Web应用组件部署多个过滤器，这些过滤器组成一个过滤器链，每个过滤器只执行某个特定的操作或者检查。这样请求在到达被访问的目标之前，需要经过这个过滤器链。

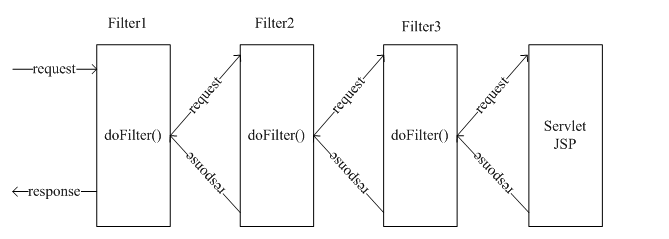


图5.3 过滤器的链式结构

### 5.3.2其它权限架构

Apache Shiro 是 Java 的一个安全框架。我们经常看到它被拿来和 Spring 的 Security 来对比。他们都可以实现权限的管理。



图5.4 Shiro工作流程

可以看到：应用代码直接交互的对象是Subject，也就是说Shiro的对外API核心就是Subject；其每个API的含义：

Subject：主体，代表了当前“用户”，这个用户不一定是一个具体的人，与当前应用交互的任何东西都是Subject，如网络爬虫，机器人等；即一个抽象概念；所有Subject都绑定到SecurityManager，与Subject的所有交互都会委托给SecurityManager；可以把Subject认为是一个门面；SecurityManager才是实际的执行者；

SecurityManager：安全管理器；即所有与安全有关的操作都会与SecurityManager交互；且它管理着所有Subject；可以看出它是Shiro的核心，它负责与后边介绍的其他组件进行交互，如果学习过SpringMVC，你可以把它看成DispatcherServlet前端控制器；

Realm：域，Shiro从从Realm获取安全数据（如用户、角色、权限），就是说SecurityManager要验证用户身份，那么它需要从Realm获取相应的用户进行比较以确定用户身份是否合法；也需要从Realm得到用户相应的角色/权限进行验证用户是否能进行操作；可以把Realm看成DataSource，即安全数据源。

也就是说对于我们而言，最简单的一个Shiro应用：

(1)应用代码通过Subject来进行认证和授权，而Subject又委托给SecurityManager；

(2)我们需要给Shiro的SecurityManager注入Realm，从而让SecurityManager能得到合法的用户及其权限进行判断。

## 5.4并发设计

### 5.4.1数据库索引与优化

对于任何DBMS，索引都是进行优化的最主要的因素。对于少量的数据，没有合适的索引影响不是很大，但是，当随着数据量的增加，性能会急剧下降。

如果对多列进行索引(组合索引)，列的顺序非常重要，MySQL仅能对索引最左边的前缀进行有效的查找。例如：假设存在组合索引it1c1c2(c1,c2)，查询语句select \* from t1 where c1=1 and c2=2能够使用该索引。查询语句select \* from t1 where c1=1也能够使用该索引。但是，查询语句select \* from t1 where c2=2不能够使用该索引，因为没有组合索引的引导列，即要想使用c2列进行查找，必需出现c1等于某值。索引是在存储引擎中实现的，而不是在服务器层中实现的。所以，每种存储引擎的索引都不一定完全相同，并不是所有的存储引擎都支持所有的索引类型。索引类型有：B-Tree索引，Hash索引，聚簇索引B-Tree索引，Hash索引，聚簇索引，覆盖索引等。

### 5.4.2 Mybatis数据库连接池

连接池所处的位置如下图：

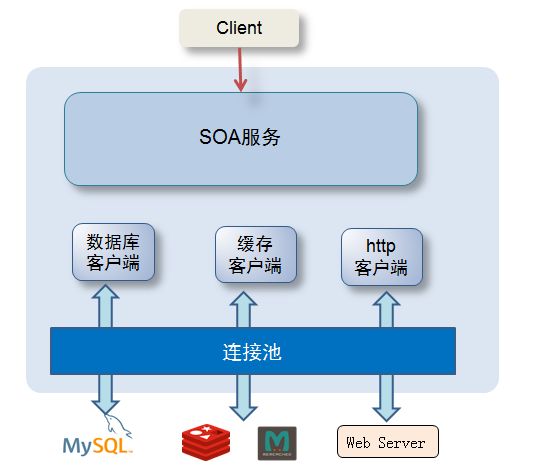


图5.5 数据库连接池位置

应用框架的业务实现一般都会访问数据库，缓存或者HTTP服务。为什么要在访问的地方加上一个连接池呢？

下面以访问MySQL为例，执行一个SQL命令，如果不使用连接池，将经历五个流程，TCP建立连接的三次握手，MySQL认证的三次握手，真正的SQL执行，MySQL的关闭，TCP的四次握手关闭。可以看到，为了执行一条SQL，却多了非常多我们不关心的网络交互。

优点：实现简单。缺点：网络IO较多，数据库的负载较高，响应时间较长及QPS较低，应用频繁的创建连接和关闭连接，导致临时对象较多，GC频繁，在关闭连接后，会出现大量TIME\_WAIT 的TCP状态（在2个MSL之后关闭）。

使用数据库连接池后，第一次访问的时候，需要建立连接。 但是之后的访问，均会复用之前创建的连接。优点：较少了网络开销。系统的性能会有一个实质的提升没了麻烦的TIME\_WAIT状态。当然，现实往往是残酷的，当我们解决了一个问题的时候，同时伴随着另外一个问题的产生。使用连接池面临的最大挑战： 连接池的性能。

有连接池（POOLED）– 这种数据源的实现利用“池”的概念将 JDBC 连接对象组织起来，避免了创建新的连接实例时所必需的初始化和认证时间。这是一种使得并发 Web 应用快速响应请求的流行处理方式。

除了上述无连接池（UNPOOLED）的属性外，会有更多属性用来配置有连接池（POOLED）的数据源：

poolMaximumActiveConnections – 在任意时间可以存在的活动（也就是正在使用）连接数量，默认值：10。

poolMaximumIdleConnections – 任意时间可能存在的空闲连接数。

poolMaximumCheckoutTime – 在被强制返回之前，池中连接被检出（checked out）时间，默认值：20000 毫秒（即 20 秒）。

poolTimeToWait – 这是一个低层设置，如果获取连接花费的相当长的时间，它会给连接池打印日志并重新尝试获取一个连接的机会（避免在误配置的情况下一直安静的失败），默认值：20000 毫秒（即 20 秒）。

poolPingQuery – 发送到数据的侦测查询，用来检验连接是否处在正常工作秩序中并准备接受请求。默认是“NO PING QUERY SET”，这会导致多数数据库驱动失败时带有一个恰当的错误消息。

poolPingEnabled – 是否启用侦测查询。若开启，也必须使用一个可执行的 SQL 语句设置 poolPingQuery 属性（最好是一个非常快的 SQL），默认值：false。

poolPingConnectionsNotUsedFor – 配置 poolPingQuery 的使用频度。这可以被设置成匹配标准的数据库连接超时时间，来避免不必要的侦测，默认值：0（即所有连接每一时刻都被侦测 — 当然仅当

poolPingEnabled 为 true 时适用。

### 5.4.3其它高并发策略

需求：数据查询时每次都需要从数据库查询数据，数据库压力很大，查询速度慢，因此设置缓存层，查询数据时先从redis中查询，如果查询不到，则到数据库中查询然后将数据库中查询的数据放到redis中一份，下次查询时就能直接从redis中查到，不需要查询数据库了。

Spring整合Redis，修改Spring配置文件，增加jedis 配置，redis服务器中心，cache配置，配置切面拦截方法。

Spring切面使用缓存

在不使用切面的时候如果我们想给某个方法加入缓存的话肯定是在方法返回之前就要加入相应的逻辑判断，只有一个或几个倒还好，如果有几十上百个的话那很困难了，而且维护起来也特别麻烦。Spring的AOP可以帮我们解决这个问题。

## 5.5安全架构

### 5.5.1使用的安全框架

GitHub上的安全框架，webSecurity。主要用servlet filter方式覆盖httpServletRequest和HttpServletResponse方式增加一些输入输出的过滤。

主要实现的安全包括：

(1)XSS过滤（获取用户输入参数和参数值进行XSS过滤，对Header和cookie value值进行XSS过滤（转码Script标签的< > 符号）。

(2)对Response的setStatus(int sc, String sm)方法 sm错误信息进行XSS过滤；

(3)对Header的CLRF进行过滤；

(4)对cookie大小和cookie的白名单进行验证；

(5)对文件上传后缀白名单进行验证；

(6)对只允许POST提交的url进行验证；

(7)CSRF攻击 tokenID防御支持；

(8)SESSION通过加密存储到cookie支持；

(9)静态资源路径去除../上级目录符号；

### 5.5.2数据库关键字加密

MD5加Salt加密用户密码。现在很多数据库设计都喜欢用单向加密的方式保存密码,验证时对提交的密码再次加密之后做密文对比，这种设计最初是防止被暴库之后黑客可以直接得到用户的密码而设计的,因为是单向加密,所以即便知道加密算法也无法得到用户的实际密码。但是所谓道高一尺魔高一丈, 在彩虹表出现之后, 单纯的md5也不安全了。彩虹表是一个用于加密散列函数逆运算的预先计算好的表, 常用于破解加密过的密码散列。一般主流的彩虹表都在100G以上。 查找表常常用于包含有限字符固定长度纯文本密码的加密。这是以空间换时间的典型实践, 在每一次尝试都计算的暴力破解中使用更少的计算能力和更多的储存空间，但却比简单的每个输入一条散列的翻查表使用更少的储存空间和更多的计算性能。使用加盐的KDF函数可以使这种攻击难以实现。

### 5.5.3其它安全框架

Spring Security是一个能够为基于Spring的企业应用系统提供声明式的安全访问控制解决方案的安全框架。它提供了一组可以在Spring应用上下文中配置的Bean，充分利用了Spring IoC，DI（控制反转Inversion of Control ,DI:Dependency Injection 依赖注入）和AOP（面向切面编程）功能，为应用系统提供声明式的安全访问控制功能，减少了为企业系统安全控制编写大量重复代码的工作。

使用众多的拦截器对url拦截，以此来管理权限。

登陆验证拦截器AuthenticationProcessingFilter流程如下：

用户登陆，会被AuthenticationProcessingFilter拦截，调用AuthenticationManager的实现，而且AuthenticationManager会调用ProviderManager来获取用户验证信息（不同的Provider调用的服务不同，因为这些信息可以是在数据库上，可以是在LDAP服务器上，可以是xml配置文件上等），如果验证通过后会将用户的权限信息封装一个User放到spring的全局缓存SecurityContextHolder中，以备后面访问资源时使用。

访问资源（即授权管理），访问url时，会通过AbstractSecurityInterceptor拦截器拦截，其中会调用FilterInvocationSecurityMetadataSource的方法来获取被拦截url所需的全部权限，在调用授权管理器AccessDecisionManager，这个授权管理器会通过spring的全局缓存SecurityContextHolder获取用户的权限信息，还会获取被拦截的url和被拦截url所需的全部权限，然后根据所配的策略（有：一票决定，一票否定，少数服从多数等），如果权限足够，则返回，权限不够则报错并调用权限不足页面。

# 参考文献

[1] Perry, D.E. Software engineering and software architecture. In: Feng, Yu-lin, ed. Proceedings of the International Conference on

Software: Theory and Practice. Beijing: Electronic Industry Press, 2000. 1~4.

[2] Boehm, B. Engineering context (for software architecture), invited talk, In: Garlan D., ed. Proceedings of the 1st International

Workshop on Architecture for Software Systems Seattle. New York: ACM Press, 1995. 1~8.

[3] Perry, D.E., Wolf, A.L. Foundations for the study of software architecture. ACM SIGSOFT Software Engineer Notes, 1992,17(4):40~50

[4] 孙昌爱,金茂忠,刘超等.软件体系结构研究综述[J].软件学报,2002,13(7):1228-1237.

[5]汪保杰,王如龙. 软件体系结构及其风格的现状与发展[J]. 计算机系统应用,2009,(11):201-204.

[6] 计春雷. 软件设计模式及其应用研究[J]. 上海电机学院学报,2006,(05):46-49+70.

[7]徐烈辉. 粒子系统基本理论及其应用[J]. 电脑与信息技术,2009,(03):9-10+39.

[8]黄宁,陈未如. 基于架构风格的软件可靠性评估[J]. 计算机系统应用,2009,(05):198-201.

[9]王艳清,陈红. 基于SSM框架的智能web系统研发设计[J]. 计算机工程与设计,2012,(12):4751-4757.

[10] 郑雪,徐亚娟. 中间件的概念、分类与应用[J]. 微型电脑应用,1999,(02):15-17.

[11] 张宏森,朱征宇. 四层B/S结构及解决方案[J]. 计算机应用研究,2002,(09):21-22.