分类号_____ 密级_____

UDC _____ 编号_____

中国科学院研究生院

硕士学位论文

SkyMouse 天文服务搜索整合系统的设计与实现

孙 华 平

指导教师 赵永恒 研究员 中国科学院国家天文台

崔辰州 副研究员 中国科学院国家天文台

申请学位级别 硕士 学科专业名称 天文技术与方法

论文提交日期 2007 年 5 月 论文答辩日期 2007 年 6 月

培养单位 中国科学院国家天文台

学位授予单位 中国科学院研究生院

答辩委员会主席 邓李才 研究员

Design and Implementation of SkyMouse, an Integrated Astronomical Information Search Engine

Huaping Sun

Advisor:
Prof.Yongheng Zhao & Dr. Chenzhou Cui

National Astronomical Observatories, Chinese Academy of Sciences

June 2007

摘　　要

随着虚拟天文台的进一步发展，天文学家可以通过越来越多的服务来辅助完成研究工作。但随着不同种类的服务的进一步增多，也带来了很多的问题。突出表现在某个服务被开发出来天文学家并不知道或是有些服务比较复杂天文学家不易直接使用。SkyMouse系统就是为了解决以上问题而诞生的，它是基于互联网的新一代天文门户，在该门户中整合大多数的天文服务如SIMBAD、NED、ADS、Vizier等，而且其他新开发出来的天文服务也可以方便地整合到SkyMouse系统中。用户可以直接通过SkyMouse系统非常方便的访问世界范围内的天文资源，诸如星表，文献，光谱，图像等，而且通过这种整合的方式，用户还可以使用多个服务的协同调用来完成以往单个服务所不能完成的诸如智能分析检索等工作。

本论文介绍了 SkyMouse 项目的详细设计，论述了中国虚拟天文台进行 SkyMouse 项目开发的可行性和必要性。详细地讲述了中国虚拟天文台 SkyMouse 系统的系统设计与实现方法，并针对天文服务调用的问题上提出了自己的解决方案。在本文中将详细介绍 SkyMouse 程序架构、工作流程、数据库设计等诸多方面,结合中国虚拟天文台研发需要，本研究在天文服务访问、天文服务整合等几个方面进行了探索。

关键字：　中国虚拟天文台　SkyMouse　WebService　CGI　.NET

Abstract

With the progress of information technologies and astronomical observation technologies, as an example of cyber-infrastructure-based science, Virtual Observatory is initiated and spreading quickly. More and more on-line accessible database systems and different kinds of services are available. Although astronomers have been aware of the importance of interoperability, integrated access to the on-line information is still difficult.

SkyMouse is a smart system developed by Chinese Virtual Observatory project to let us access different online resource services easily then ever. Not like some other VO efforts on uniformed access systems, for example, NVO DataScope, SkyMouse tries to show a comprehensive overview for a specific object, but not to snatch as much data as possible. Stimulated by a simple Mouse Over on an interested object name, various VO-compliant and traditional databases, i.e. SIMBAD, NED, VizieR, DSS, ADS, are queried by the SkyMouse. An overview for the given object, including basic information, image, observation and references, is displayed in user's default web browser. In this article, we will introduce the framework of SkyMouse. During the development of SkyMouse, various of Web services will be called. In order to invoke these Web services, two problems must be solved, i.e. interoperability and performance. In the paper, a detailed description for these problems and our resolution are given.

Key Words: China –VO - SkyMouse - WebService - CGI - .NET – technique: miscellaneous-method: miscellaneous-astronomical data base: miscellaneous

目 录

图表目录

第1章 虚拟天文台发展概况

1.1 中国虚拟天文台概述

1.1.1 虚拟天文台概述

传统天文学经过了几十年的观测积累，加上近几年新型的天文仪器不断投入运行，产出的观测数据已成指数增长。天文学家面对如此巨大的数据量，欣喜的同时也在为如何有效地处理和分析这些数据而感到忧虑。如何从这包含数十亿天体的多波段海量数据中探求科学发现，已成为天文学家亟待解决的问题。与此同时信息科学领域也在经历着快速的变革，伴随高性能计算和互联网技术的发展，网格计算应运而生。网格计算[1]是指通过将多台计算机组成网格状网络，“模拟实现高性能计算机”的技术，实现互联网上所有资源的互联互通，包括计算资源、存储资源、通信资源、软件资源、知识资源等。

近些年来伴随着网格技术、XML、WEB 服务技术等新兴 IT 技术日趋成熟，虚拟天文台应运而生。虚拟天文台成为基于网格技术解决海量复杂天文数据访问和管理的一种有效手段。虚拟天文台通过先进的信息技术把全球范围内的研究资源无缝透明地联结起来，组成数据密集型在线天文研究环境[2]。它将 TB 甚至 PB 量级的数据、波长遍及从 γ 射线到射电波段的数十亿个天体的图像库、高度复杂的数据挖掘工具和统计分析工具、具有数千 PB 量级容量的存储设备和每秒运算次数达到万亿次的超级计算设备、以及各主要天文数据中心之间的高速网络连成一体；它能使世界各地的天文学家可以快速查询每个 PB 量级的数据库；使埋藏在庞大星表和图像数据库中的多变量模式可视化；增加了发现复杂规律和稀有天体的机会；可以进行大样本星表的统计研究及数据挖掘。

虚拟天文台的概念提出后各国也相继出台了自己的虚拟天文台计划。为了相互借鉴，相互促进以及国际上各项目之间的协调与合作，2002 年成立了国际虚拟天文台联盟（International Virtual Observation Alliance， IVOA）[3]，目前国际虚拟天文台联盟中成员已经增加到 16 个国家。如表 1.1 所示：

项目	地区	网址
中国虚拟天文台(China-VO)	中国	http://www.china-vo.org/
国家虚拟天文台(NVO)	美国	http://www.us-vo.org/
英国虚拟天文台(AstroGrid)	英国	http://www.astrogrid.org/
欧洲虚拟天文台(Euro-VO)	欧洲	http://www.euro-vo.org/
法国虚拟天文台(VO-France)	法国	http://www.france-vo.org/
加拿大虚拟天文台(CVO)	加拿大	http://services.cadc-ccda.hia-iha.nrc-cnrc.gc.ca/cvo/
印度虚拟天文台(VO-India)	印度	http://vo.iucaa.ernet.in/%7Evoi/
德国虚拟天文台(GAVO)	德国	http://www.g-vo.org/portal/
澳大利亚虚拟天文台(Aus-VO)	澳大利亚	http://www.aus-vo.org/
日本虚拟天文台(JVO)	日本	http://jvo.nao.ac.jp/
俄罗斯虚拟天文台(RVO)	俄罗斯	http://www.inasan.rssi.ru/eng/rvo/
意大利虚拟天文台(DRACO)	意大利	http://wwwas.oat.ts.astro.it/twiki/bin/view/Draco/WebHome
韩国虚拟天文台(KVO)	韩国	http://kvo.kao.re.kr/
匈牙利虚拟天文台(HVO)	匈牙利	http://hvo.elte.hu/en/
西班牙虚拟天文台(SVO)	西班牙	http://laeff.esa.es/svo/
亚美尼亚虚拟天文台(ArVO)	亚美尼亚	http://www.aras.am/arvo.htm

表 1.1：国际各国虚拟天文台项目

1.1.2 中国虚拟天文台概述

为了紧跟时代的发展，缩小与发达国家的信息天文学方面的差距，2002 年以中国科学院国家天文台为首在国内天文界首先提出建设“中国虚拟天文台[4]”项目。同年十月，中国虚拟天文台加入国际虚拟天文台联盟。

根据国内实际情况和需求，以及 China-VO 自身的条件和特点，China-VO 的研究开发

工作主要在如下五个方面进行：

1. 中国虚拟天文台系统平台的开发（China-VO Platform）：在先进的 IT 技

术基础上，按照 IVOA 互操作标准搭建 China-VO 系统平台。

2. 国内外天文研究资源的统一访问（Uniform Access to On-line Astronomical Resources and Services）：研究如何在 China-VO 系统平台上以与 IVOA 标准兼容的方式实现对分布式异构天文研究资源与服务的统一访问。

3. 支持 VO 的项目与观测设施（VO-ready Projects and Facilities）：协助国内外天文学家和工程技术人员，把手中的天文观测项目和观测设施建设为支持 VO 标准的项目和设施。

4. 基于 VO 的天文研究示范（VO-based Astronomical Research Activities）：利用 VO 环境开展天文研究活动，积累 VO 科研经验，带动天文学家使用 VO 资源。

5. 基于 VO 的天文科普教育（VO-based Public Education）：利用 VO 丰富的资源开展天文教育和普及活动。

中国虚拟天文台是一个相互之间能进行互操作的数据集和分析工具的集合，它们通过互联网给天文学家提供一个可在线进行天文科学研究的平台。它的科学目标与意义是：

1) 中国虚拟天文台将采用与我国现在正在建造的大天区面积多目标光纤光谱望远镜[5]（LAMOST）项目紧密结合的方式，充分发挥LAMOST 光学光谱数据中心的作用，建设“VO-oriented LAMOST”，使光谱数据及其相关处理技术成为中国虚拟天文台的核心与特色。

2) 中国虚拟天文台将利用先进的网格和WEB服务技术，方便、快捷、实时地把今后LAMOST产出的数据发布，以便能和国外的数据进行交互与融合。

3) 中国虚拟天文台将最大程度地集中国内各天文研究机构、IT 研究机构和数学等相关领域的人才资源，共同努力实现目标。

4) 中国虚拟天文台将采用开放的运作方式，与国际上各虚拟天文台计划开展充分的合作，在IVOA 中发挥积极作用。

5) 中国虚拟天文台将利用IVO丰富的资源加强教育和科学普及工作，提高我国公众的科学素质。

作为 e-Science 时代中国天文学研究的新平台和“网关”，中国虚拟天文台把国内的星表数据资源与国际共享，并把国际的资源输入到国内，逐步实现与

国际各天文数据中心数据的互操作和透明访问。最终，中国虚拟天文台的用户将可以通过使用互联网的方式方便地获取国际范围内的天文数据，可以对分布在各地的数据进行大规模多波段的分析和挖掘，并利用中国虚拟天文台的可视化系统将结果以 2D、3D 图像的形式显示出来。作为国际虚拟天文台的成员，中国虚拟天文台正在与国际上各个虚拟天文台进行紧密合作，在 IVOA 中发挥积极作用。

1.2 虚拟天文台的网络服务访问与互操作

1.2.1 虚拟天文台的网络服务访问与互操作的必要性

随着虚拟天文台的进一步发展，越来越多的天文服务被开发出来。随着各种服务的诞生也带来了诸多问题。例如某个服务被开发出来不为人知，或是服务比较复杂难以使用。由于各国自身条件不同和服务分散，这些天文服务相互孤立，这就导致了调用各国天文服务的异构性和复杂性。单个服务功能局限性很大，无法通过多个服务协同工作完成更复杂的功能。在这样一种背景下构建一个切实高效的天文信息综合检索平台，从而为天文学家和天文爱好者提供必要的信息查询支持和信息分析支持，是天文领域目前势在必行的急务之一。

1.2.2 虚拟天文台的网络服务访问与互操作的现状

目前国际上针对天文服务整合的网站只有美国国家虚拟天文台（US National Virtual Observatory , NVO）的 DataScope,它提供多个不同服务的整合服务，用户可以通过 DataScope 去查找世界范围内的多个服务，同时提供统一的显示页面。其查询门户如图 1.1 所示。

图 1.1：DataScope 程序截图

DataScope 符合 IVOA 的数据访问与互操作标准，前台为 Web 服务，用户可以很方便地访问。因此我们如果要想开展天文服务的整合与互操作的研究，使我们的服务平台可以与国外的天文服务进行互操作，获取国外天文服务的资源，必须满足 IVOA 所推出的规范与标准，借鉴他们的研发经验。

第2章　天文服务整合与互操作相关技术概述

2.1　网格技术

2.1.1　网格的定义

网格[1]概念最早于上世纪 90 年代中期提出，用于表述在高端科学和工程上分布式计算的一种基础构造形式。简单地讲，网格是把整个网络整合成一台巨大的超级计算机，实现计算资源、存储资源、数据资源、信息资源、知识资源、专家资源的全面共享。这样组织起来的“虚拟的超级计算机”有两大优势，一是数据处理能力超强；二是能充分利用网上的闲置处理能力。网格就是将收集网络上所有可用的分布式计算资源提供给最终用户或组建成一个巨大的计算系统。

潜藏在网格概念之中确切而特殊的问题就在于，实现对等的资源共享和解决动态的、分布式的虚拟组织所遇到的问题[2]。我们所关心的共享不仅仅是简单的文件交换，更强调直接对计算机、软件、数据以及其他资源的直接访问。共享联系能够开始于任何小组动态地处理新的节点，跨越不同的平台、语言和编程环境，显而易见，互操作性成为不可或缺的部分。在一个统一的网格环境下，互操作性意味着共同的协议。因此，在网格体系中最重要的是一个协议体系，该协议定义了基础机制，虚拟组织用户们通过这个机制来商议、建立、管理和开发分配关系。

2.1.2　网格体系结构

网格的体系结构[3]是开放的、可扩张的体系结构。体系结构为整个系统提供了一个结构、行为和属性的高级抽象，由对构成系统的元素的描述、元素间的相互作用、元素集成的模式以及这些模式的约束组成。体系结构不仅指定了系统的组织结构和拓扑结构，并且显示了系统需求和构成系统元素之间的相互关系，提供了一些设计决策的基本原理。

图 2.1 为网格的沙漏体系结构图，将功能分散在不同的层次上。从底层到上层依次是：构造层、连接层、资源层、汇集层、应用层。

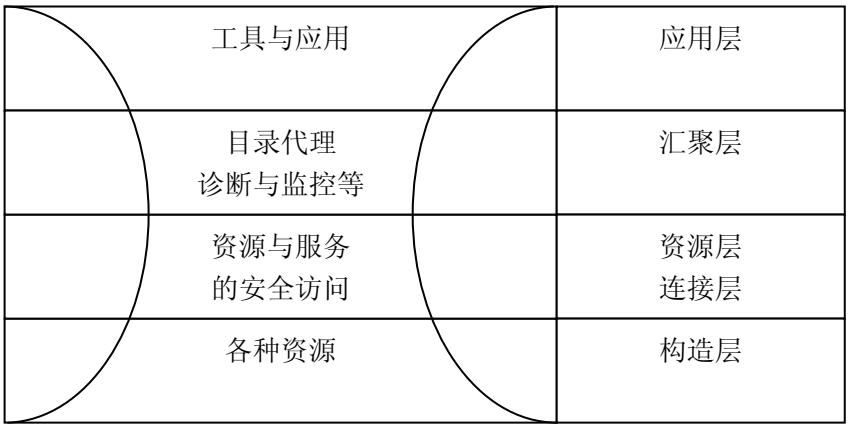

图 2.1：网格沙漏体系结构

1） 构造层：网格构造层是本地控制的接口，提供了共享的资源，执行本地与资源相关的操作。

2） 连接层：连接层定义了核心的网络事务处理所需要的通信和认证协议，以支持便利安全的通信。

3） 资源层：资源层建立在连接层的通信和认证协议之上，定义的协议包括安全的连接、初始化、监视与控制、计费等。

4） 汇集层：汇集层协调各种资源。如果说资源层着眼于和单个资源进行交互，那么汇集层就着眼于协调多种资源的共享。

5） 应用层：应用层是在虚拟组织环境中存在的，它是用户需求的具体体现。

2.1.3 OGSA 体系结构与 WEB 服务

开放网格服务架构(Open Gird Service Architecture，OGSA)[4]是在网络协议和 Web 服务技术融合的基础上提出的一套规范和标准。它最突出的思想就是以“服务”为中心。在 OGSA 框架中，将一切都抽象为服务，包括计算机、程序、数据资源、仪器设备等。这种观念有利于通过统一的标准接口来管理和使用网格。

OGSA 平台是旨在为各种网格系统所共同面临的基本问题定义标准的解决

方案和机制。这些基本问题包括网格服务间的通信、身份确立、授权对话、服务发现、错误通告、服务集管理等。

随着近些年来 WEB 服务技术在商业上应用的不断扩大、功能不断完善，网格技术也借鉴了其优点，将 WEB 服务技术融入网格体系架构之中，弥补网格技术在使用现有资源上的不足。WSRF 的定义意味着网格和 Web 服务团体在一个共同的基础上前进。网格技术与 WEB 技术融合图例如图 2.2 所示。

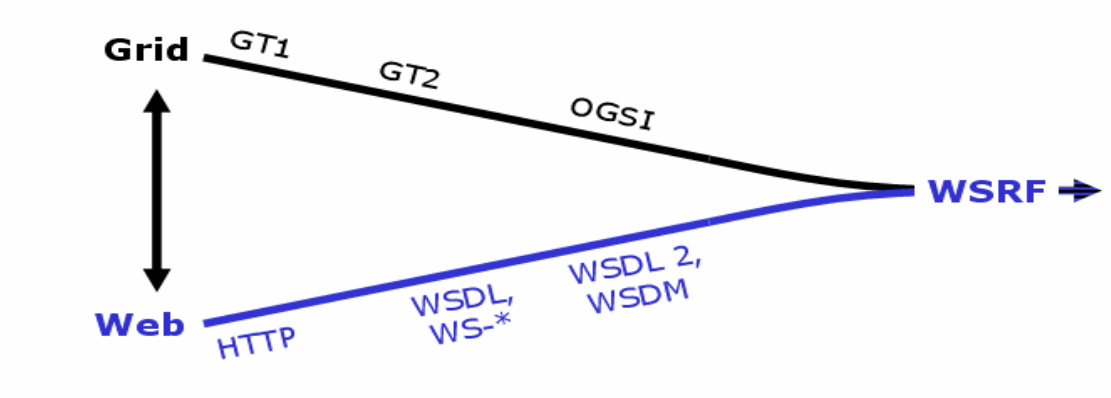

图 2.2 网格技术与 WEB 技术融合

OGSA 平台包括三个基本元素：OGSA 平台接口、OGSA 平台模型和 WEB 服务资源框架[5]（Web Service Resource Framework，WSRF）。OGSA 体系结构如图 2.3 所示。

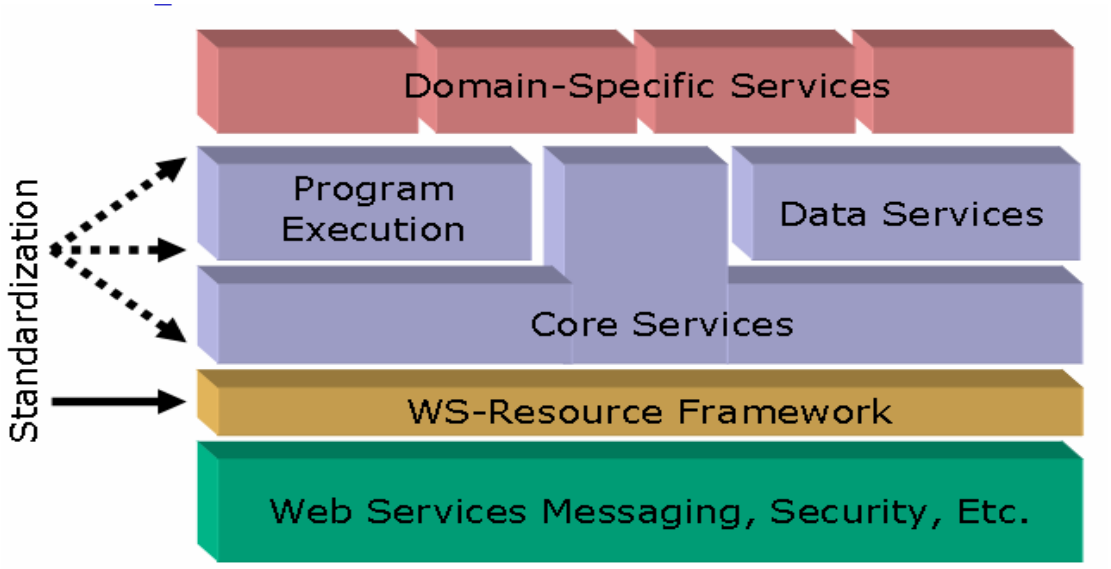

图 2.3 OGSA 体系结构

2.1.4 Globus Toolkit 4 平台和 WSRF

Globus 项目[6]是目前国际上最有影响的网格计算项目之一。Globus 对信息安全、资源管理、信息服务、数据管理以及应用开发环境等网格计算的关键理论和技术进行了广泛的研究，开发出能在多种平台上运行的网格计算工具包软件(Globus Toolkit)，能够用来帮助规划和组建大型的网格试验和应用平台，开发适合大型网格系统运行的大型应用程序。Globus 项目组发布了 Globus Toolkit 4(GT4)最新版本。GT4 体系结构做了较大的变动，由开放网格服务基础架构(OGSI)迁移到了 Web 服务资源框架(WSRF)，WSRF 是一套规范的集合，重新定义了网格服务的组成，提高了与 WEB 服务的兼容性。

WS-Resource 被定义为由 Web 服务和有状态的资源构成的实体。有状态的资源可以在 Web 服务消息交换中使用。可以创建和销毁 WS-Resources，而且可以通过消息交换查询或更改其状态。WS-Resource 有四个很重要的称为 ACID 特性的软件工程特性。在 Web Services Atomic Transaction 规范 [WS-AtomicTransaction] 中，对这些特性进行了介绍，如下所示：

1) 原子性（Atomicity）： 在事务单元中，要么对有状态资源全部进行更新，要么全都不进行更新。
2) 一致性（Consistency）： 有状态资源应该总能维护一致状态，即使失效之后也是如此。
3) 隔离性（Isolation）： 应该在给定的事务单元中隔离对有状态资源的更新。
4) 持久性（Durability）： 在事务单元中对有状态资源的更新是永久性的。

WSRF 的提出是为了避免将来两者不兼容，同时可以利用大量的现成 Web 服务的开发工具、已有成果；同时减少了太多的面向对象机制的引入，提高了系统性能。下表 2.1 总结了 Globus Toolkit 4 主要协议的一些基本特性和兼容性问题。

服务	协议	特性	向后兼容性
数据传输	可靠文件传	1. 使用 GridFTP 控制和监视第三方文件传输。 2. 指数补偿（Exponential back-off）。	与 OGSI (GT3.2) 不存在向后兼容

	输(RFT)	3. 全部传输或者全部不传输。 4. 并行传输。 5. TCP 缓冲区大小。 6. 递归目录传输。	
资源管理	WS-GRAM	1.改进任务性能：并行性、吞吐量和等待时间等。 2.改进可靠性/可恢复性。 3.支持 mpich-g2 任务，包括： 多任务提交。 在一个任务中协调处理。 在一个多任务的子任务之间进行协调。	该协议已经被修改成 WSRF 兼容的。该版本与以前的版本之间不存在向后兼容
信息服务	MDS4	**索引服务** 1. 基于 WSRF 而不是 OGSI。 2. 已经删除 Xindice 支持。 3. 已经重构聚合的永久性配置。 **全新的服务** 1. 触发器服务。 2. 聚合器（Aggregator）。 3. 归档服务。	与 GT3.2 的索引服务不兼容，因为该服务已经使用 WSRF 代替 OGSI，重新进行了建模

表 2.1 GT4 中的新协议

GT4 核心服务提供了程序开发支持的功能，包括开放式开发模式和访问网格服务的实现，像网格资源管理(Grid Resource Allocation Management，GRAM)。应用 GT4 强有力的理由就是因为它是建立在现有的 WEB 服务标准和技术的基础上，像 SOAP 和 WSDL。网格服务提供的接口都是通过 WSDL 来描述的。GT4 提供了一个软件仓库，像安全支持、软件的探索、软件的资源管理、软件的调用、软件之间的通信，异常处理和数据管理等。图 2.4 主要描述了在服务器端的 GT4 中的主要组件体系结构。

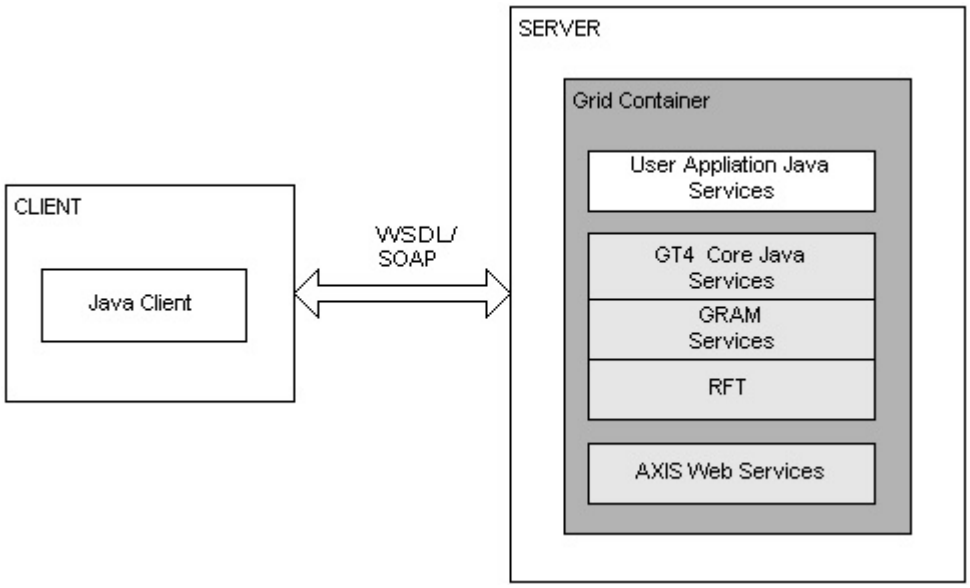

图 2.4 GT4 主要组件体系结构

GT4 结构由一个网格容器组成，网格容器主要用来管理所有部署的 WEB 服务，贯穿于每个 WEB 服务的运行周期。GT4 使用 apache 的 axis 作为它的 WEB 服务的引擎来处理所有的 SOAP 消息，JAX-RPC (Java API for XML-Based RPC) 处理和 WEB 服务的配置。

2.2 开发语言

2.2.1 .NET Framework

2000 年 6 月，微软公司推出了“Microsoft.NET 下一代互联网软件和服务战略”，引起 IT 行业的广泛关注。在.NET 体系结构中，XML 是各应用之间无缝接合的关键.Microsoft.NET 代表了一个集合、一个环境、一个可以作为平台支持下一代 Internet 的可编程结构。

.NET 首先是一个环境。是一个理想化的未来互联网环境，微软的构想是一个“不再关注单个网站、单个设备与因特网相连的互联网环境，而是要让所有的计算机群、相关设备和服务商协同工作”的网络计算环境。简而言之，互联网提供的服务，要能够完成更高程度的自动化处理。未来的互联网，应该以一个整体服务的形式展现在最终用户面前，用户只需要知道自己想要什么，而不需要一步步地在网上搜索、操作来达到自己的目的。这是一种理想，但的的确

确是互联网的发展趋势所在。

.NET 谋求的是一种理想的互联网环境。而要搭建这样一种互联网环境，首先需要解决的问题是针对现有因特网的缺陷，来设计和创造一种下一代 Internet 结构。这种结构不是物理网络层次上的拓扑结构，而是面向软件和应用层次的一种有别于浏览器只能静态浏览的可编程 Internet 软件结构。因此.NET 把自己定位为可以作为平台支持下一代 Internet 的可编程结构。

.NET 的最终目的就是让用户在任何地方、任何时间，以及利用任何设备都能访问他们所需要的信息、文件和程序。而用户不需要知道这些东西存在什么地方，甚至连如何获得等具体细节都不知道。他们只需发出请求，然后只管接收就是了，而所有后台的复杂性是完全屏蔽起来的。所以对于企业的 IT 人员来说，他们也不需要管理复杂的平台以及各种分布应用之间的工作是如何协调的。.NET Framework 包括通用语言运行环境、Framework 类库和 Active Server Pages.NET

有了理想目标和相应可编程 Internet 软件结构，.NET 这样的一个协同计算环境的具体实现还必然需要一系列的软件产品支撑，因此微软的.NET 还包括一个产品的集合。这个集合包含以下组成部分：

.NET 平台

这一平台建立在 XML 和因特网标准协议的基础上，包含了.NET 的基础结构和基础工具，为开发新型的互动协作软件提供了一个先进的体系结构模型。

.NET 的技术特征

.NET 包括 4 个重要特点，一是软件变服务；二是基于 XML 的共同语言；三是融合多种设备和平台；四是新一代的人机界面。这四个特点基本上覆盖了.NET 的技术特征。

伴随着 ASP 产业的兴起，软件正逐渐从产品形式向服务形式转化，这是整个 IT 行业的大势所趋。在.NET 中，最终的软件应用是以 Web 服务的形式出现并在 Internet 发布的。Web 服务是一种包装后的可以在 Web 上发布的组件，.NET 通过 WSDL 协议来描述和发布这种 Web 服务信息，通过 DISCO 协议来查找相关的服务，通过 SOAP 协议进行相关的简单对象传递和调用。

基于 XML 的共同语言

XML 是从 SGML 语言演化而来的一种标记语言。作为元语言，它可以定义不同种类应用的数据交换语言。在.NET 体系结构中，XML 作为一种应用间无缝接合的手段，用于多种应用之间的数据采集与合并，用于不同应用之间的互操作和协同工作。具体而言，.NET 通过 XML 语言定义了简单对象访问协议（SOAP）、Web 服务描述语言（WSDL）、Web 服务发现协议（DISCO）。SOAP 协议提供了在无中心分布环境中使用 XML 交换结构化有类型数据的简单轻量的机制。WSDL 协议定义了服务描述文档的结构，如类型、消息、端口类型、端口和服务本身。DISCO 协议定义了如何从资源或者资源集合中提取服务描述文档、相关服务发现算法等。

融合多种设备和平台

随着 Internet 逐渐成为一个信息和数据的中心，各种设备和服务已经或正在接入和融入 Internet，成为其中的一部分。.NET 谋求与各种 Internet 接入设备和平台的一体化，主要关注在无线设备和家庭网络设备及相关软件、平台方面。

新一代的人机界面

新一代人机界面主要体现在“智能与互动”两个方面。.NET 包括通过自然语音、视觉、手写等多种模式的输入和表现方法；基于 XML 的可编辑复合信息架构——通用画布；个性化的信息代理服务；使机器能够更好地进行自动处理的智能标记等技术。

.NET 框架的目的是更容易建立网络应用程序和网络服务。建立在操作系统最底层的服务，是管理运行时代码需求的 common language runtime，这些代码可以用任何现代编程语言所写。Runtime 提供了许多服务，这些服务有助于简化代码开发和应用程序的开发同时也将提高应用程序的可靠性。.NET Framework 包括一套可被开发者用于任何编程语言的类库。在此之上是许多应用程序模板，这些模板特定地为开发网络站点和网络服务提供高级组件和服务。.NET Framework 提供了应用程序模型及关键技术，让开发人员容易以原有的技术来产生、布署，并可以继续发展具有高安全、高稳定，并具高延展的 Web Services。对于.NET Framework 而言，所有的组件都可以成为 Web Services，Web Services 只不过是另一种形态的组件罢了。微软将 COM 的优点整合进来，它可以不用像 COM 那么严谨的来栓锁两个对象，.NET Framework 以松散的方

式来栓锁 Web Services 这种形态的组件。这样的结果让开发人员非常容易地发展出强而有力的 Web 服务组件，提高了整体的安全及可靠性，并且大大的增加系统的延展性。

基于以上对.NET Framework 的介绍和分析，我们可以看出.NET Framework 创建就是基于 Web Service 理念的，而 SkyMouse 这套基于 Web Service 的整合系统使用.NET Framework 将是最好的方式。

2.2.2 Ajax

Ajax 是 Asynchronous JavaScript and XML 的缩写。它并不是一门新的语言或技术,它实际上是几项技术按一定的方式组合在协作中发挥各自的作用,它包括

使用 XHTML 和 CSS 标准化呈现;

使用 DOM 实现动态显示和交互;

使用 XML 和 XSLT 进行数据交换与处理;

使用 XMLHttpRequest 进行异步数据读取;

最后用 JavaScript 绑定和处理所有数据;

Ajax 的工作原理相当于在用户和服务器之间加了—个中间层,使用户操作与服务器响应异步化。并不是所有的用户请求都提交给服务器,像—些数据验证和数据处理等都交给 Ajax 引擎自己来做,只有确定需要从服务器读取新数据时再由 Ajax 引擎代为向服务器提交请求。

所谓的 Ajax 其核心只有 JavaScript、XMLHTTPRequest 和 DOM，如果所用数据格式为 XML 的话，还可以再加上 XML 这一项(Ajax 从服务器端返回的数据可以是 XML 格式，也可以是文本等其他格式)。

在旧的交互方式中,由用户触发一个 HTTP 请求到服务器,服务器对其进行处理后再返回一个新的 HTML 页到客户端,每当服务器处理客户端提交的请求时,客户都只能空闲等待,并且哪怕只是一次很小的交互、只需从服务器端得到很简单的一个数据,都要返回一个完整的 HTML 页,而用户每次都要浪费时间和带宽去重新读取整个页面。

而使用 Ajax 后，用户从感觉上几乎所有的操作都会很快响应，没有页面重

载（白屏）的等待。

1、XMLHTTPRequest

Ajax 的一个最大的特点是无需刷新页面便可向服务器传输或读写数据(又称无刷新更新页面),这一特点主要得益于 XMLHTTP 组件 XMLHTTPRequest 对象。这样就可以向再发桌面应用程序只同服务器进行数据层面的交换,而不用每次都刷新界面，也不用每次将数据处理的工作提交给服务器来做,这样即减轻了服务器的负担又加快了响应速度、缩短了用户等候时间。

最早应用 XMLHTTP 的是微软,IE（IE5 以上）通过允许开发人员在 Web 页面内部使用 XMLHTTP ActiveX 组件扩展自身的功能,开发人员可以不用从当前的 Web 页面导航而直接传输数据到服务器上或者从服务器取数据。这个功能是很重要的,因为它帮助减少了无状态连接的痛苦,它还可以排除下载冗余 HTML 的需要,从而提高进程的速度。Mozilla（Mozilla1.0 以上及 NetScape7 以上）做出的回应是创建它自己的继承 XML 代理类：XMLHttpRequest 类。Opera 也将在其 v7.6x+以后的版本中支持 XMLHttpRequest 对象。对于大多数情况，XMLHttpRequest 对象和 XMLHTTP 组件很相似,方法和属性也类似,只是有一小部分属性不支持。

2、JavaScript

JavaScript 是一在浏览器中大量使用的编程语言，它以前一直被贬低为一门糟糕的语言（它确实在使用上比较枯燥），以在常被用来作一些用来炫耀的小玩意和恶作剧或是单调琐碎的表单验证。但事实是，它是一门真正的编程语言，有着自己的标准并在各种浏览器中被广泛支持。

3、DOM

DOM 是给 HTML 和 XML 文件使用的一组 API。它提供了文件的结构表述，让你可以改变其中的内容及可见物。其本质是建立网页与 Script 或程序语言沟通的桥梁。

4、XML

可扩展的标记语言（Extensible Markup Language , XML）具有一种开放的、可扩展的、可自描述的语言结构，它已经成为网上数据和文档传输的标准。它是用来描述数据结构的一种语言，就正如它的名字一样。它使对某些结构化数

据的定义更加容易，并且可以通过它和其他应用程序交换数据。

5、综合

JavaScript 的 Ajax 引擎读取信息，并且互动地重写 DOM，这使网页能无缝化重构，也就是在页面已经下载完毕后改变页面内容，这是我们一直在通过 JavaScript 和 DOM 在广泛使用的方法，但要使网页真正动态起来，不仅要内部的互动，还需要从外部获取数据。在以前，我们是让用户来输入数据并通过 DOM 来改变网页内容的，但现在，XMLHTTPRequest 可以让我们在不重载页面的情况下读写服务器上的数据，使用户的输入达到最少。

基于 XML 的网络通讯也并不是新事物，实际上 FLASH 和 JAVA Applet 都有不错的表现，现在这种富交互在网页上也可用了，基于标准化的并被广泛支持和技术，并且不需要插件或下载小程序。

Ajax 是传统 WEB 应用程序的一个转变。以前是服务器每次生成 HTML 页面并返回给客户端（浏览器）。在大多数网站中，很多页面中至少 90%都是一样的，比如：结构、格式、页头、页尾、广告等，所不同的只是一小部分的内容，但每次服务器都会生成所有的页面再返回给客户端，这无形之中是一种浪费，不管是对于用户的时间、带宽、CPU 耗用，还是对于 ISP 的高价租用的带宽和空间来说。如果按一页来算，只能几 K 或是几十 K 可能并不起眼，但像 SINA 每天要生成几百万个页面的大 ISP 来说，可以说是损失巨大的。而 AJAX 可以所为客户端和服务器的中间层，来处理客户端的请求，并根据需要向服务器端发送请求，用什么就取什么、用多少就取多少，就不会有数据的冗余和浪费，减少了数据下载总量，而且更新页面时不用重载全部内容，只更新需要更新的那部分即可，相对于纯后台处理并重载的方式缩短了用户等待时间，也把对资源的浪费降到最低，基于标准化的并被广泛支持和技术，并且不需要插件或下载小程序，所以 Ajax 对于用户和 ISP 来说是双盈的。

Ajax 使 WEB 中的界面与应用分离（也可以说是数据与呈现分离），而在以前两者是没有清晰的界限的，数据与呈现分离的分离，有利于分工合作、减少非技术人员对页面的修改造成的 WEB 应用程序错误、提高效率、也更加适用于现在的发布系统。也可以把以前的一些服务器负担的工作转嫁到客户端，利于客户端闲置的处理能力来处理。

Ajax 理念的出现，揭开了无刷新更新页面时代的序幕，并有代替传统 web 开发中采用 form(表单)递交方式更新 web 页面的趋势，可以算是一个里程碑。但 Ajax 不是适用于所有地方的，它的适用范围是由它的特性所决定的。

举个应用的例子，是关于级联菜单方面的 Ajax 应用。

我们以前的对级联菜单的处理是这样的：

为了避免每次对菜单的操作引起的重载页面，不采用每次调用后台的方式，而是一次性将级联菜单的所有数据全部读取出来并写入数组，然后根据用户的操作用 JavaScript 来控制它的子集项目的呈现，这样虽然解决了操作响应速度、不重载页面以及避免向服务器频繁发送请求的问题，但是如果用户不对菜单进行操作或只对菜单中的一部分进行操作的话，那读取的数据中的一部分就会成为冗余数据而浪费用户的资源，特别是在菜单结构复杂、数据量大的情况下（比如菜单有很多级、每一级菜又有上百个项目），这种弊端就更为突出。

如果在此案中应用 Ajax 后，结果就会有所改观：

在初始化页面时我们只读出它的第一级的所有数据并显示，在用户操作一级菜单其中一项时，会通过 Ajax 向后台请求当前一级项目所属的二级子菜单的所有数据，如果再继续请求已经呈现的二级菜单中的一项时，再向后面请求所操作二级菜单项对应的所有三级菜单的所有数据，以此类推……这样，用什么就取什么、用多少就取多少，就不会有数据的冗余和浪费，减少了数据下载总量，而且更新页面时不用重载全部内容，只更新需要更新的那部分即可，相对于后台处理并重载的方式缩短了用户等待时间，也把对资源的浪费降到最低。此外，Ajax 由于可以调用外部数据，也可以实现数据聚合的功能（当然要有相应授权），比如微软刚刚在 3 月 15 日发布的在线 RSS 阅读器 BETA 版；还可以利于一些开放的数据，开发自己的一些应用程序，比如用 Amazon 的数据作的一些新颖的图书搜索应用。

总之，Ajax 适用于交互较多，频繁读数据，数据分类良好的 WEB 应用。

Ajax 的优势

1、减轻服务器的负担。因为 Ajax 的根本理念是“按需取数据”，所以最大可能在减少了冗余请求和响应对服务器造成的负担；

2、无刷新更新页面，减少用户实际和心理等待时间；

首先，“按需取数据”的模式减少了数据的实际读取量，打个很形象的比方，如果说重载的方式是从一个终点回到原点再到另一个终点的话，那么 Ajax 就是以一个终点为基点到达另一个终点；

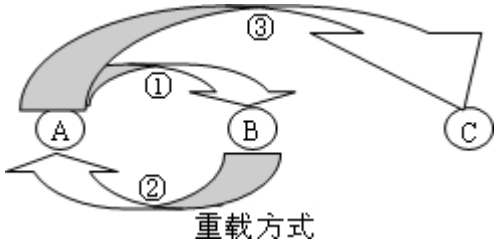

图 2.5:传统页面重载方式

图 2.6:Ajax 重载方式

其次，即使要读取比较大的数据，也不用像 RELOAD 一样出现白屏的情况，由于 Ajax 是用 XMLHTTP 发送请求得到服务端应答数据，在不重新载入整个页面的情况下用 Javascript 操作 DOM 最终更新页面的，所以在读取数据的过程中，用户所面对的也不是白屏，而是原来的页面状态（或者可以加一个 LOADING 的提示框让用户了解数据读取的状态），只有当接收到全部数据后才更新相应部分的内容，而这种更新也是瞬间的，用户几乎感觉不到。总之用户是很敏感的，他们能感觉到你对他们的体贴，虽然不太可能立竿见影的效果，但会在用户的心中一点一滴的积累他们对网站的依赖。

3、更好的用户体验；

4、也可以把以前的一些服务器负担的工作转嫁到客户端，利于客户端闲置的处理能力来处理，减轻服务器和带宽的负担，节约空间和带宽的租用成本；

5、Ajax 由于可以调用外部数据；

6、基于标准化的并被广泛支持和技术，并且不需要插件或下载小程序；

7、Ajax 使 WEB 中的界面与应用分离（也可以说是数据与呈现分离）；

基于以上对 Ajax 技术的介绍和分析，我们可以看到 SkyMouse 系统的需求是需要在全世界范围内调用 Web Service，而不同的 Web Service 的相应速度也是不同的，我们可以通过 Ajax 把先返回的 Web 服务展现给用户，无刷新的应

用可以提供更好的用户体验。

2.3 IVOA 的数据访问与互操作标准

2.3.1 VOTable 标准

VOTable[9]是 XML 格式的，用来存储和传输天文星表数据及其元数据的标准。在 VOTable 标准中，提供了对星表元数据的描述。元数据包括对星表自身的描述和对星表每一列的描述，如表名、使用的坐标框架、数据类型、单位等等。同时它还引入了统一内容描述（Unified Content Descriptors，UCD），使程序可以理解每一列的具体天文含义。

2.3.2 SkyNodeInterface 标准

SkyNodeInterface[10]标准以 ADQL 标准和 VOTable 标准为基础，从概念上定义了天文数据结点需要实现的一些必要的接口。这些接口依据其功能大致分两类，一类接口用来获取元数据。这些元数据是描述一个查询中列的信息和表的信息，同时为了增强互操作和实现更高层面的数据处理，也获得了每一列的 UCD 信息以及每一列的单位。另一类接口则是用来接收 ADQL 查询语句并且返回 VOTable 格式的结果。

第3章 SkyMouse 概要描述与设计

3.1 SkyMouse 概述

SkyMouse 天文服务搜索系统是一套智能的天文服务整合系统。SkyMouse 对天文领域常用的几个主要的网络服务进行整合，构建健壮的服务整合模型。同时完善屏幕取词客户端，最大限度的方便天文学家使用。SkyMouse 着眼于世界范围内的天文服务，为天文学家和天文爱好者构建基于 Web 的星表、文献、光谱、图像数据和信息检索平台，为天文信息检索提供强有力的支持。SkyMouse 系统包括 SkyMouse 门户网站和 SkyMouse 屏幕取词客户端两部分组成。

3.1.1 SkyMouse 系统建设原则

3.1.1.1 整合性原则

SkyMouse 系统是一套整合天文相关服务的系统。初期将整合大多数天文服务如 Simbad,NED,ADS,Vizier 等等，用户可以直接通过 SkyMouse 系统访问星表，文献，光谱，图像等数据，而不需要再进入相应网站查询。整合性将是 SkyMouse 的发展方向，随着 IVOA 的标准进一步制定和规范，将会有越来越多的系统可以自动无缝地整合到 SkyMouse 系统中。

3.1.1.2 实用性原则

SkyMouse 系统构建的首要目的是借助互联网服务于天文学家和天文爱好者，因此，SkyMouse 系统的建设必须遵循实用性原则，以天文学家，天文爱好者的常用操作为基本实现目标。

3.1.1.3 定制性原则

SkyMouse 系统并不是对现有网络服务的简单整合。定制性对于 SkyMouse 系统来说有两个层面的意义。第一层面是开发人员可以使用 SkyMouse 系统中已有的服务，通过几个服务的协同调用，让 SkyMouse 系统去完成一些复杂的

智能分析检索工作。第二个层面是对于不同类型的用户，SkyMouse 系统会预制一些该类用户常用的服务类型。用户也可以选择自己感兴趣的服务进行查询。

3.1.1.4 分步实施原则

SkyMouse 系统的定位着眼于虚拟天文台领域的大局，必须根据虚拟天文台领域目前和未来的发展进程，分期分段逐步实现系统的近期目标和远期目标。

3.1.2 SkyMouse 系统设计原则

3.1.2.1 先进性原则

SkyMouse 系统的设计遵循先进性原则，采用目前先进的数据建模工具和软件组件技术来构建系统的概念模型、逻辑模型和物理模型。

3.1.2.2 可靠性原则

SkyMouse 系统遵循可靠性设计原则，采用的软件技术、设计方法和实施策略均为业界所广泛验证和成熟应用；SkyMouse 系统的数据采集和加工处理保证数据的完整性、一致性和可靠性。

3.1.2.3 灵活性原则

SkyMouse 系统采用灵活的设计原则，应用功能模型保持最大限度的共性，同时允许各类用户根据实际情况订制个性活动。

3.1.2.4 适应性原则

SkyMouse 系统的设计充分考虑到天文领域网络服务的各种情况，遵循 SkyMouse 系统应用功能的适应性设计原则,对某些特定的或不规范的服务进行特殊处理，以适应日后应用的不断完善。

3.2 SkyMouse 前期调研分析

3.2.1 需求调研

SkyMouse 系统的需求调研分析是刻画 SkyMouse 系统模型的首要也是最重要的一个步骤。我们在研究了国际虚拟天文台进展和天文领域常见需求的基础上，通过中国虚拟天文台人员的仔细讨论和对天文学家与天文爱好者的需求调查，完成目前的需求调研。在 2005 年中国虚拟天文台年会上，我们演示了 SkyMouse 的系统原型，与会的各位天文学家和计算机人员也提出了很多很好的建议。

3.2.2 需求分析

3.2.2.1 功能需求

功能需求包括系统管理，Web Service 调用，CGI 服务调用，工作流引擎，门户网站，屏幕取词等六项功能。

A、 系统管理

功能： 完成 SkyMouse 系统门户网站管理。

描述： 系统管理包括用户类型管理，日志管理，Web Service 管理，CGI 服务管理，工作流引擎管理等

B、Web Service 调用

功能： 完成 Web Service 调用的通用模块；

描述： 对已有 Web Service 的调用模块。通过该模块可以直接调用互联网上已有的 Web Service。该模块的特点就是通用性。初期对不同的 Web Service 可以使用.Net 常用的引用模式的调用，在后期该模块将和 UDDI 注册结合，实现对 Web Service 自动化的调用。

C、 CGI 服务调用

功能： 完成互联网上常用的 CGI 程序调用的通用模块

描述： 由于目前 Web Service 尚未完全普及，互联网上还存在着大量基于传统 CGI 程序的服务，该模块负责调用这些服务。在调用时传递 CGI 程序所需的参数，取得返回结果后根据正则表达式或是字符串匹配的方式把最终的结果展现给用户。

D、 工作流引擎

功能：对 Web Service 和 CGI 程序进行协同调用；

描述：简单的 Web Service 或是 CGI 程序的整合意义事实不大。SkyMouse 系统通过该工作流引擎对整合进来的 Web Service 或 CGI 程序进行协同调用，以完成更复杂的功能。同时该引擎也承担相应 CGI 或 Web Service 参数的自动传递功能。目前典型的需求就是调用 Simbad 的名称解析服务得到查询关键字所在的赤经赤纬再将所得的赤经赤纬传递给下一个所需的服务。

E、 门户网站

功能： 以网页的形式接收用户提交查询信息将所得信息展现给用户；

描述： SkyMouse 系统服务器。基于 B/S 系统能很好的解决跨平台，维护和更新等问题。

F、 屏幕取词

功能： 提供基于屏幕取词的查询功能

描述： 类似金山词霸的屏幕取词功能。用户可以直接在屏幕取词并把取得的关键字传递给 SkyMouse 系统。初期需要提供基于 Windows 操作系统的屏幕取词客户端。后期将提供基于 Linux 平台的版本，并且可能会在客户端上整合进诸如天文词汇翻译等功能。

3.2.2.2 系统需求

A、 系统要有与外部数据的交互接口

SkyMouse 系统与外部系统可根据约定的数据格式进行数据的导入、导出；需要提供一个 Web Service 的接口。

B、 系统功能的共性和个性要兼备

不同的用户需求不同，SkyMouse 系统应该充分考虑个性化需求；

C、 系统需提供灵活多样的查询方式

SkyMouse 系统应满足各种信息查询模式。

3.2.2.3 其它需求

A、 SkyMouse 系统的性能和实时性需要充分考虑；

B、 SkyMouse 系统作为统一的信息共享平台，网络规划需保障信息的快速交换和远程传递；

3.3 SkyMouse 系统设计

3.3.1 系统功能模型

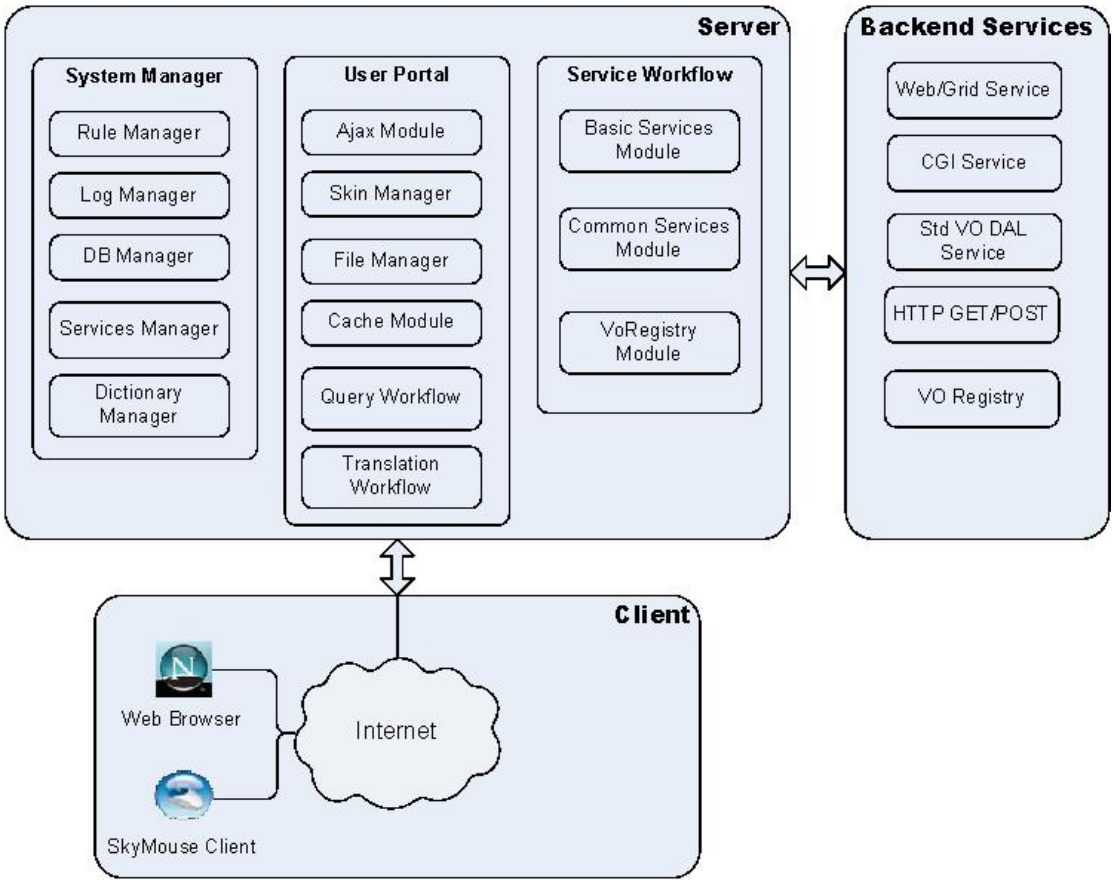

图 3.1:SkyMouse 系统架构图

系统功能模型说明：

1.服务器端组成：

A:系统管理模块：完成 SkyMouse 系统管理功能.包括规则管理子模块,日志管理子模块，数据库管理子模块，服务信息管理子模块，字典管理子模块五部分。

a. 规则管理子模块负责 SkyMouse 调用服务的规则管理。

b. 日志管理子模块负责记录用户访问日志并生成相应的统计数据。

c. 数据库管理子模块负责所有与数据库的相关操作。

d. 服务信息管理子模块负责所有在 SkyMouse 中登记的服务基本信息，调用信息等信息的维护。

e. 字典管理子模块负责 SkyMouse 字典的操作和维护

B:用户门户： SkyMouse 系统和用户交互的界面.包括 Ajax 通用调用子模块，服务显示样式子模块，文件管理子模块，缓存管理子模块和查询接口，翻译工作流模块。

a. Ajax 负责通过 Ajax 无刷新的调用 Page 管理子模块所获得调用 Web 服务结束后的展示页面。

b. 服务显示样式子模块负责调用相应的显示样式来显示服务返回的数据。

c. 文件管理子模块负责对图片，视频等文件的远程获取功能。

d. 缓存管理子模块负责系统缓存的功能，对常用的关键字调用页面和结果进行缓存以加快下一次的调用速度。

e. 查询接口为用户最终看到的显示接口，在服务器端已网页的形式体现。

f. 翻译工作六模块服务调用系统词典或互联网词典进行关键词翻译的工作。

C:服务工作流模块： 该模块负责 SkyMouse 系统和互联网上的 Web Service 和 CGI 程序的交互，是 Skymouse 最核心的模块。包括基本服务调用模块，通用服务调用模块， VoRegistry 调用模块。

a. 基本服务调用模块负责调用所有通过引用的方式添加到 SkyMouse 中的所有服务。这种方式能提供更高的调用效率，缺点是对服务的定制性太强，每次添加服务都需要修改服务器的源代码并重新生成。对于调用频繁的服务都通过该模块调用。

b. 通用服务调用模块服务调用所有管理员通过后台管理直接添加的服务，该模块通过反射生成代理类的方式调用服务。优点是可以很方便的添加服务，并且通过该模块直接调用通过 Voregistry 获得的服务，缺点是效率比第一种要差。对于调用不频繁并且需要实时添加的服务都通过该模块调用。

c. VoRegistry 调用模块负责所有和 VoRegistry 交互的工作，包括从 VoRegistry 中获取所需服务等功能。目前 SkyMouse 使用的 VoRegistry 是 NVO 的 VoRegistry。

2.客户端组成：

SkyMouse 系统不需要客户端也可以访问，用户可以直接使用浏览器访问，支持的浏览器有 IE，FireFox,Netscape 等常用的浏览器。同时 SkyMouse 系统还提供一个基于屏幕取词的客户端。

3.3.2 SkyMouse 网络括扑图

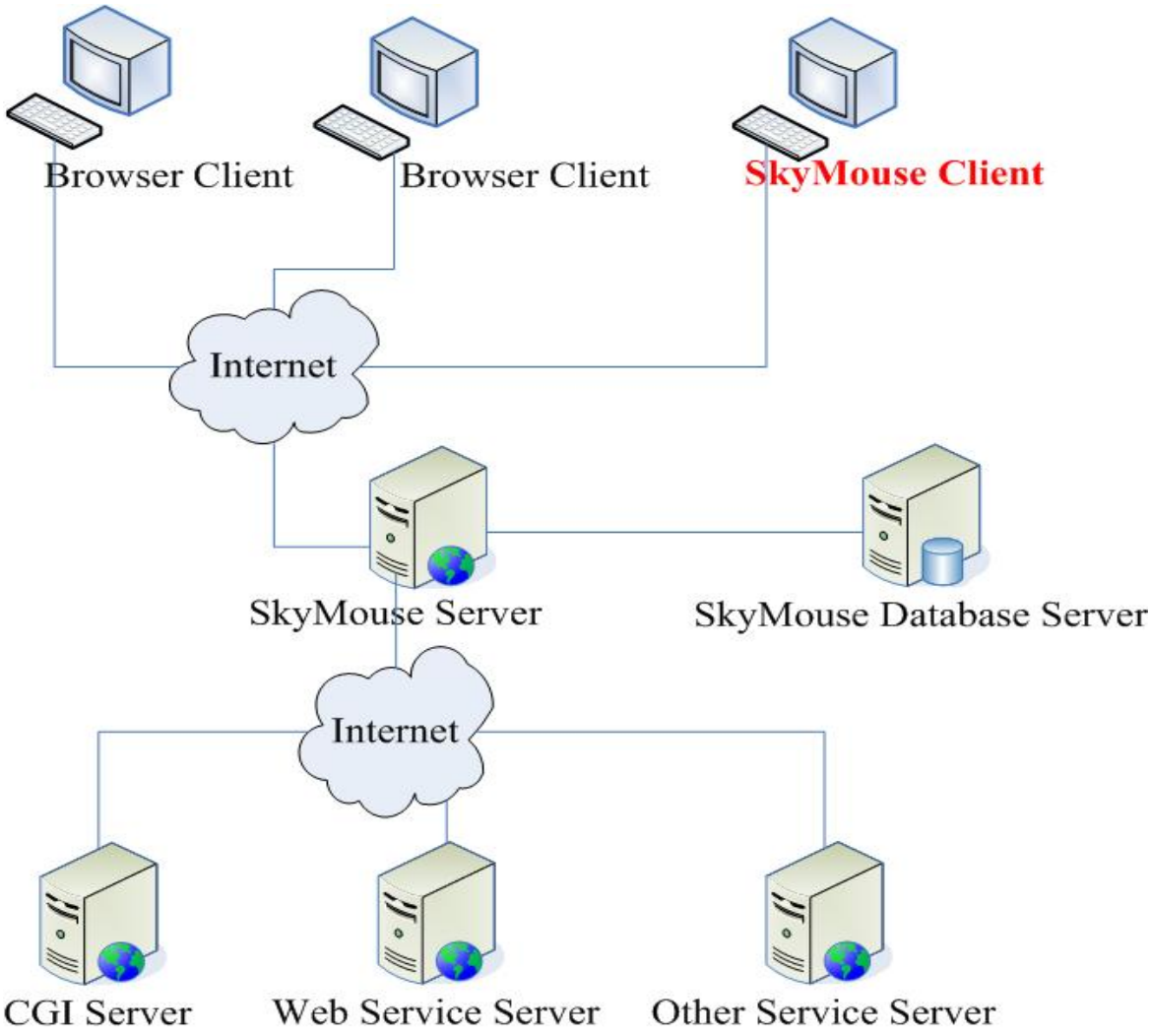

图 3.2：SkyMouse 系统网络括扑图

SkyMouse 系统的网络图由用户接入层、应用服务层组成，各层完成的功能如下：

用户接入层

用户接入 SkyMouse 系统的直接方式，如笔记本电脑，网络终端等；SkyMouse 系统采用 B/S 方式，因此任何一台可访问服务器的终端经授权都可以应用系统；远程用户接入可通过国际互联网和远程拨号两种方式。同时 SkyMouse 还提供基于屏幕取词的客户端，用户可以通过客户端来访问 SkyMouse 系统。

应用服务层

应用服务层为用户提供最终的应用服务，由应用服务器和数据库服务器组成。应用服务器提供 SkyMouse 系统的服务应用运行环境，并响应和处理客户

端的服务请求,同时负责调用互联网上的其他 Web 服务；数据库服务器主要运行 SkyMouse 系统的关系型数据库管理系统，为系统的结构化信息数据提供统一高效的存取。

应用服务器和数据库服务器独立专用，以保证各种接入方式的独立应用和系统应用的松散耦合以及安全性。

3.4 SkyMouse 软件开发

SkyMouse 系统采用的软件开发技术与工具：

✧ 软件架构模式采用 B/S 模式(浏览器/服务器模式)；

B/S 模式便于客户端的零维护管理；程序发布不受影响；需要的配置资源紧紧是浏览器，不需要其它而外资源。

✧ 开发语言采用.NET；

基于微软的.Net 架构，利于构建基于 WEBSERVICE 的应用服务。

✧ 数据库采用目前采用 SQLSERVER2000

目前 SkyMouse 系统的数据库是基于微软的 SQLSERVER，SQLSERVER 与其开发语言拥有天然的结合力，便于开发。但是 SkyMouse 系统本身对于数据库并无特定要求，使用 Oracle，MySQL 甚至是 XML 作数据库都可以。

第4章 SkyMouse 详细设计与实现

SkyMouse 系统由客户端和服务器端组成。客户端基于 VC 语言开发，实现屏幕取词，天体信息显示等功能。服务器端基于 C#语言开发，后台数据库基于微软的 SQLSERVER。在本章中将逐一对各个部分进行详细说明。

4.1 客户端详细设计

4.1.1 客户端界面

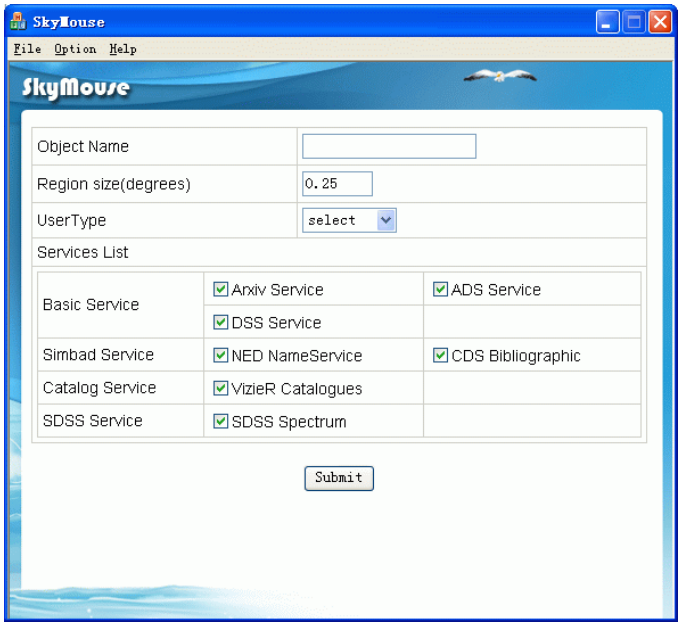

图 4.1:SkyMouse 客户端界面

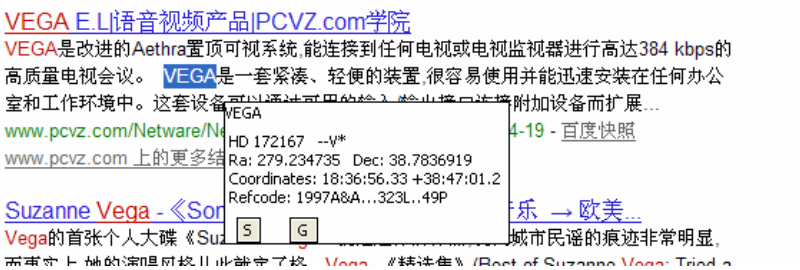

图 4.2:SkyMouse 客户端运行界面

客户端显示界面如图 4.1 所示。用户可以直接在该页面输入需要查询的天文关键字并选择所选服务提交给服务器端进行处理。同时用户也可以如图 4.2 选

择通过屏幕取词的方式，在屏幕的任意位置选择相应的天文单词查询该天体的天体基本信息。同时，针对中国的用户，客户端还包含了中英文翻译解释的功能。

4.1.2 客户端处理流程

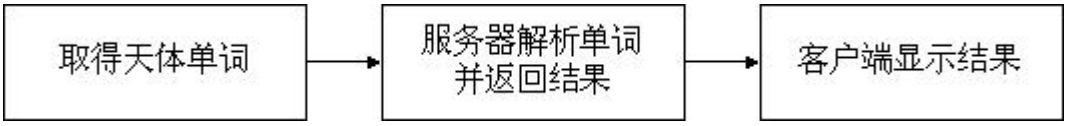

图 4.3:SkyMouse 客户端处理流程

流程描述如下：

1. 取得天文单词。并把单词提交到 SkyMouse 服务器上。
2. SkyMouse 服务器处理提交的单词，并将返回结果返回给客户端。
3. 客户端取得服务器返回的内容，并通过浮动小窗口的形式展现。

4.1.3 屏幕取词实现方式

在 Windows 系统实现屏幕取词中一般有两种方式：

1：截获部分 Window API 调用,截获 textout、TextOutA 函数，通过拦截这两个函数获得屏幕上鼠标范围的文字 。这种方式的优点是实现起来比较容易，并且兼容性比较好。缺点是对某些程序无法获得文字。

2：对每个设备上下文做一分拷贝,并跟踪所有修改上下文的操作。只要上下文 DC 有变化就可以获得。这种方式的优点是可以获得的文字更多，但是缺点也很明显，程序的兼容性比较差，容易引起其他程序的冲突并导致可能系统死机。

基于以上两种方式的比较，在 SkyMouse 系统中，我们采用的是第一种方式，通过截获 Window API 的调用，获得屏幕当前的输出文字。

截获 Window API 的调用，一般来说也可以采取两种方法：

第一种方法是通过直接改写 Window API 在内存中的映像，嵌入汇编代码，使之被调用时跳转到指定的地址运行来截获；第二种方法则是通过改写 IAT（import address table 输入地址表），重定向 Window API 函数的调用来实现对

Window API 的截获。

第一种方法的实现起来比较繁琐。因为不同的 Window 系统使用的 API 的位数是不同的，Window 9x 系统基本上使用的都是同名的 16 位 API，而 Window NT 和 Window XP 使用的基本上就是 32 位的 API 了，我们很难为不同的系统写不同位数的汇编程序，所以基于这点考虑，在 SkyMouse 系统中，我们采用的是第二种拦截方法，这种方法在 Window 9x、Window NT、Window XP 下面运行都比较稳定，兼容性也很好。这种方式设计的知识有 Windows 虚拟内存的管理、打破进程边界墙、向应用程序的进程空间中注入代码、PE（portable executable）文件格式和 IAT 等较底层的知识，在文章中我们对涉及到的知识简单介绍，最后会给出程序的关键代码。

Window 9x 给每一个进程分配了 4GB 的地址空间，Window NT 会分配 2GB，系统保留了 2GB 到 4GB 之间的地址空间禁止进程访问。在 Win9x 中，2GB 到 4GB 这部分虚拟地址空间是由所有的 Win32 进程所共享的，这部分地址空间加载了共享 Win32 dll、内存映射文件和 vxd、内存管理器和文件系统码，Win9x 中这部分对于每一个进程都是可见的，这也是 Win9x 操作系统不够健壮的原因。Win9x 中为 16 位操作系统保留了 0 到 4MB 的地址空间，而在 4MB 到 2GB 之间也就是 Win32 进程私有的地址空间，由于 每个进程的地址空间都是相对独立的，也就是说，如果程序想截获其它进程中的 API 调用，就必须打破进程边界墙，向其它的进程中注入截获 API 调用的代码，这项工作我们交给钩子函数来完成。所有系统钩子的函数必须要在动态库里，这样的话，当进程隐式或显式调用一个动态库里的函数时，系统会把这个动态库映射到这个进程的虚拟地址空间里，这使得 dll 成为进程的一部分，以这个进程的身份执行，使用这个进程的堆栈，也就是说动态链接库中的代码被钩子函数注入了其它 GUI 进程的地址空间，（当然对于非 GUI 进程，钩子函数就无能为力了，这也就是我们这种方式的局限性，但是事实上这样的程序是非常少的） 。当包含钩子的 dll 注入其它进程后，就可以取得映射到这个进程虚拟内存里的各个模块（exe 和 dll）的基地址。

exe 和 dll 被映射到虚拟内存空间的什么地方是由它们的基地址决定的。它们的基地址是在链接时由链接器决定的。系统将 exe 和 dll 原封不动映射到虚拟

内存空间中，它们在内存中的结构与磁盘上的静态文件结构是一样的。即 PE (portable executable) 文件格式。我们得到了进程模块的基地址以后，就可以根据 PE 文件的格式穷举这个模块的 image_import_descriptor 数组，看看进程空间中是否引入了我们需要截获的函数所在的动态链接库，比如需要截获“TextOutA”，就必须检查“gdi32.dll”是否被引入了。在 PE 文件的“.idata”段中包含了所有的引入函数信息，还有 iat（import address table）的 rva（relative virtual address）地址。 所有进程对给定的 API 函数的调用总是通过 PE 文件的一个地方来转移的，这就是一个该模块(可以是 exe 或 dll)的“.idata”段中的 iat 输入地址表（import address table）。在那里有所有本模块调用的其它 dll 的函数名及地址。对其它 dll 的函数调用实际上只是跳转到输入地址表，由输入地址表再跳转到 dll 真正的函数入口。

我们可以通过 image_import_descriptor 数组来访问“.idata”段中引入的 dll 的信息，然后通过 image_thunk_data 数组来针对一个被引入的 dll 访问该 dll 中被引入的每个函数的信息，找到我们需要截获的函数的跳转地址，然后改成我们自己的函数的地址。具体的做法在后面的关键代码中会有详细的讲解。

除了 API 函数的截获，要实现“鼠标屏幕取词”，还需要做一些其它的工作，简单的说来，可以把一个完整的取词过程归纳成以下几个步骤：

1. 安装鼠标钩子，通过钩子函数获得鼠标消息。 使用 Window API 函数 setWindowshookex

2. 得到鼠标的当前位置，向鼠标下的窗口发重画消息，让它调用系统函数重画窗口. 使用 Window API 函数： Windowfrompoint， screentoclient，invalidaterect .

截获对系统函数的调用，取得我们要取的词。

核心代码将在附录中展示。

4.2 服务器端详细设计

4.2.1 服务器端概述

SkyMouse 系统包括客户端和服务器端，但是最重要的应用还是在服务器端上实现的。服务器端实现服务调用、服务整合、天文翻译、系统维护等多个功能。

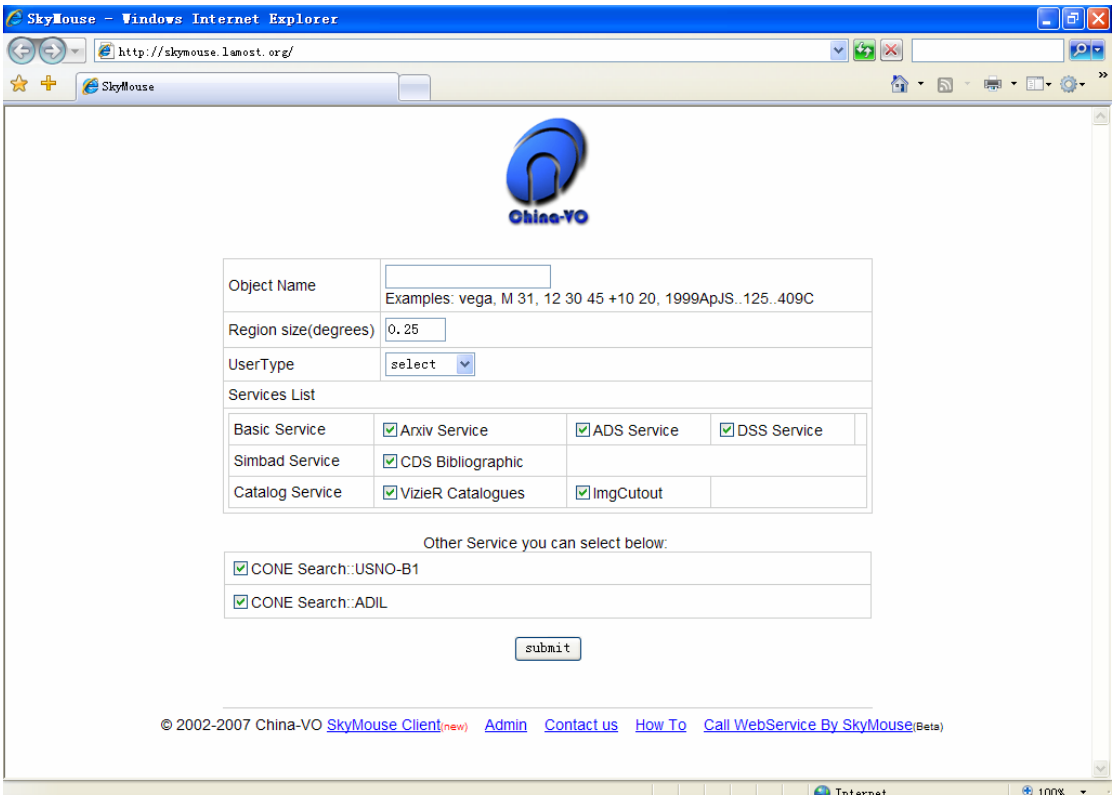

图 4.4：服务器端初始界面

图 4.5：服务器端查询显示结果页面

4.2.2 服务器端基本工作流程

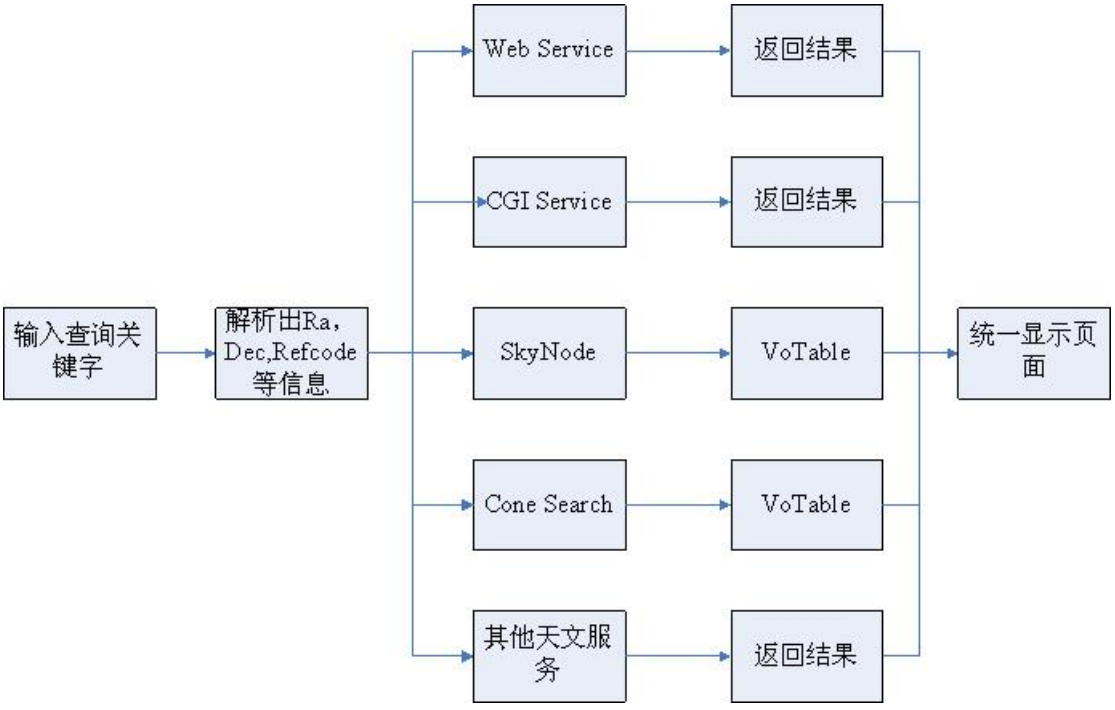

图 4.6：天文服务调用基本流程

流程描述：

1. 用户输入查询关键字或者从客户端获取查询关键字。

2. 查询系统天体信息缓存，如果缓存中没有数据，则查询 SIMBAD 和 NED 获得该天体的基本信息。

3. 将获得的基本信息传递给用户所选择调用的各个服务。包括各种 CGI 服务、WebService 服务、SkyNode 服务、ConeSearch 服务及其它天文服务。

4. 对返回的各个结果进行显示处理，如 Votable 转换为 Html，图像数据流转换为图片等。最后显示统一的查询返回结果页面，对每个服务返回的结果异步显示给用户。

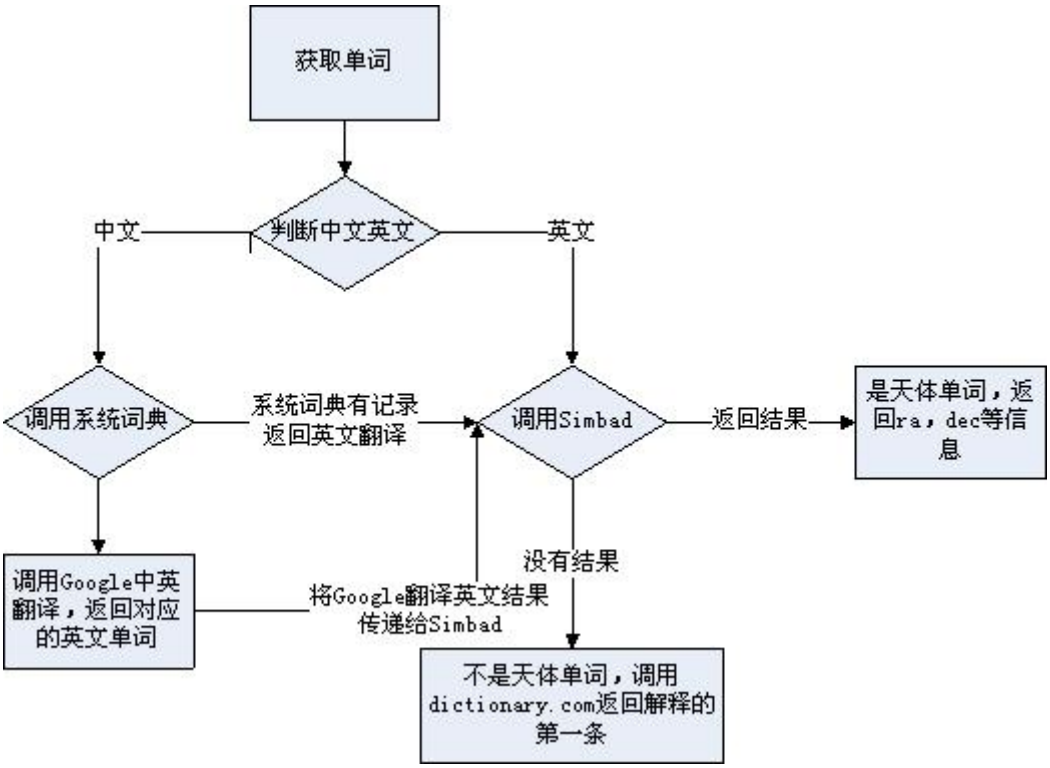

图 4.7：天体基本信息查询翻译流程

1. 用户输入查询关键字或者从客户端获取查询关键字传递到服务器端。

2. 服务器端判断查询关键字是否为英文。

3. 如果是中文，调用系统缓存中的天文词典翻译，如果有记录，直接将翻译结果传递给天体信息查询模块，进入第 4 步。如果没有记录，查询 Goolge 中英词典将 Goolge 返回的英文翻译结果传递给天体信息查询模块，进入第 4 步。

4. 如果是英文，则调用系统缓存获得天体信息，如果系统缓存中没有，则调用 SIMBAD 和 NED 获得天体信息。如果在上述服务中都没有找到结果，则设定查询关键字不是天体，调用 Dictionary.com 返回英文单词的解释结果。

5. 对返回的各个结果进行显示处理，如 Votable 转换为 Html，图像数据流转换为图片等。最后显示统一的查询返回结果页面，对每个服务返回的结果异步显示给用户。

4.2.3 服务器端各个组件说明

服务器端组件包括：

1. GotDotNet.ApplicationBlocks.Data：数据访问层组件。
2. SkyMouse.Web.Services.DynamicProxy：WebService 动态调用组件。
3. SkyMouse.Web.Services.Message：WSDL 文件解析组件。
4. SkyMouse：Web 服务器端程序。

下面对各个组件逐个进行分析解释。

4.2.3.1 GotDotNet.ApplicationBlocks.Data 数据访问组件

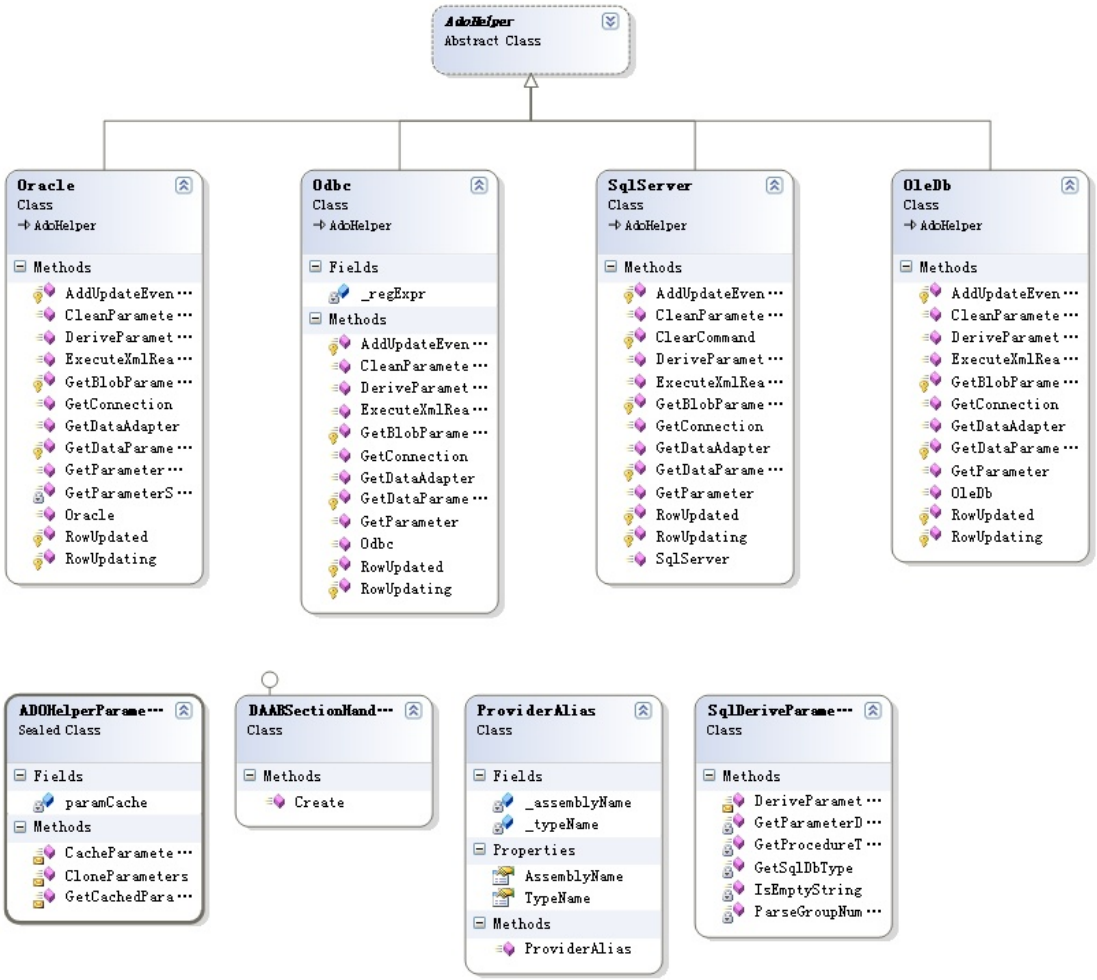

图 4.8: GotDotNet.ApplicationBlocks.Data 组件类图

该组件使用的是由微软公司提供的免费开源组件。提供数据访问层的封装。服务器端程序通过来组件同数据库进行通讯，而不是直接操作原始数据库，这样的好处是显而易见的，我们通过这种方式实现对数据库的无关性，也就是说，如果我们想要实现数据库的迁移，我们只需简单的修改一下服务器端配置文件就可以了。

4.2.3.2 SkyMouse.Web.Services.Message

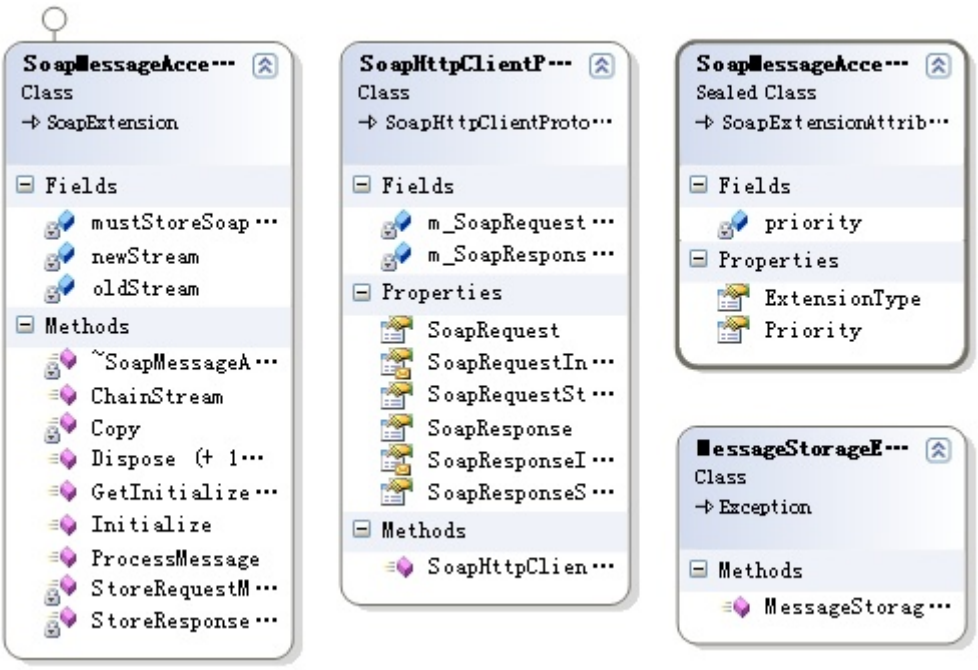

图 4.9: SkyMouse.Web.Services.Message 组件类图

该组件负责获取网络上的 WSDL 文件，并通过解析 WSDL 文件获得 WebService 中包括的函数和每个函数所需的参数及返回的结果类型。

4.2.3.3 SkyMouse.Web.Services.DynamicProxy

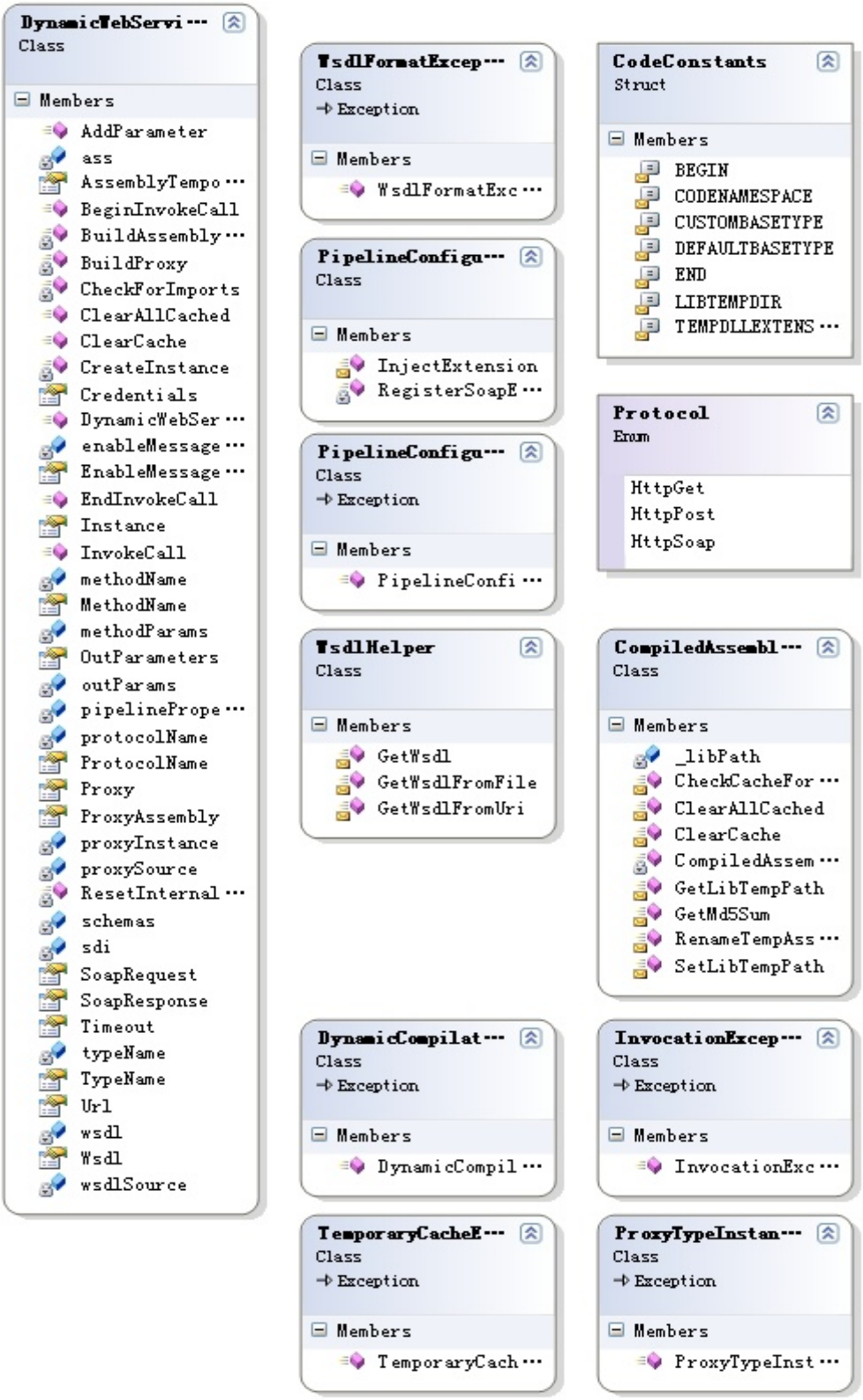

图 4.10: SkyMouse.Web.Services.DynamicProxy 组件类图

该组件负责将 SkyMouse.Web.Services.Message 组件解析过的 WSDL 通过.NET Framework 的反射功能动态生成 WebService 代理类，SkyMouse 通过该代理类生成相应的 SOAP 请求消息调用相应的 WebService，同时解析 SOAP 的返回消息获得返回的结果。

4.2.3.4 SkyMouse 服务器端

SkyMouse 服务器端程序即为 Web 网站程序。服务器端程序基于 ASP.NET 2.0 开发，通过内置的 WebService 工作流引擎实现 WebService 的异步调用，使用 Ajax 技术实现 WebService 调用结果的异步展现。

A、服务器端层次结构

在过去应用系统开发过程中，C/S 或者 B/S 体系结构得到了广泛的应用。其特点是：应用程序逻辑通常分布在客户和服务器两端，客户端发出数据资源访问请求，服务器端将结果返回客户端。但这样的结构存在着很多体系结构上的问题，比如：当客户端数目激增时，服务器端的性能会因为负载过重而大大衰减；一旦应用的需求发生变化，客户端和服务器端的应用程序都需要进行修改，给应用维护和升级带来了极大的不便；大量的数据传输增加了网络的负载等等。

在 SkyMouse 的服务器端程序中，我们使用了三层结构。所谓三层体系结构，是在客户端与数据库之间加入了一个"中间层"，也叫组件层。这里所说的三层体系，不是指物理上的三层，不是简单地放置三台机器就是三层体系结构，也不仅仅有 B/S 应用才是三层体系结构，三层是指逻辑上的三层，即使这三个层放置到一台机器上。

在 SkyMouse 中，服务器段分为了显示层，逻辑层，数据层三层。同时，根据 SkyMouse 天文服务整合的特殊需要，对每个层又进行了一定的扩展。如下图：

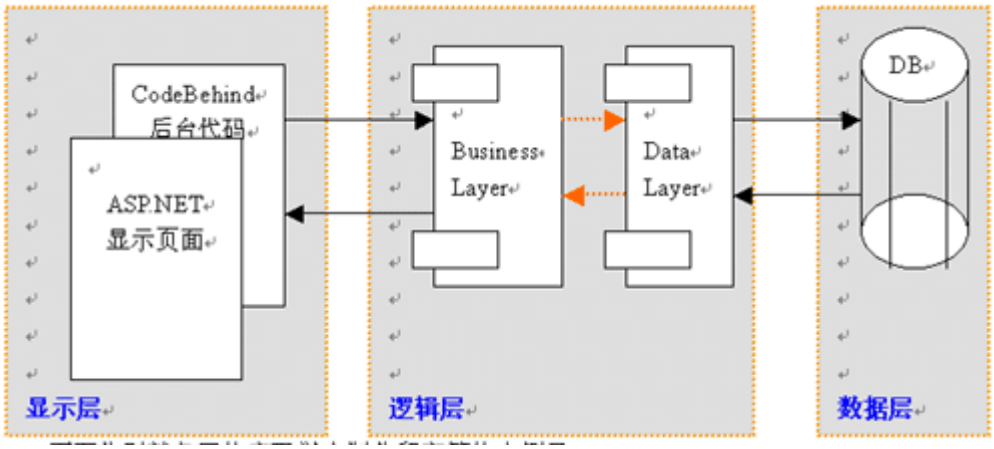

图 4.11：三层结构介绍

1. 显示层：该层不单用于所有页面的显示，同时也承载了 WebService 返回结果的异步显示。
2. 逻辑层：该层用于放置所有的业务逻辑，包括 WebService 工作流调用引擎，WebService 添加逻辑等。
3. 数据层：和以往的数据层不同，该层并不是简单的数据库抽象，而是将所有的数据库的操作以及数据库放为一层。例如所有的 SQL 语句以及 OLEDB 数据库操作都在这一层中实现。

B、服务器端配置

SkyMouse 服务器端的配置是高度灵活方便的。对于网站的基本设置，我们都可以通过 web.config 文件进行修改。文件说明如下：

```
<?xml version="1.0"?>
<configuration>
    <!--    设置Daab数据库类型-->
<configSections>
        <section name="daabProviders"
type="GotDotNet.ApplicationBlocks.Data.DAABSectionHandler,
GotDotNet.ApplicationBlocks.Data"></section>
</configSections>
    <daabProviders>
    <daabProvider alias="mainDB" assembly="GotDotNet.ApplicationBlocks.Data"
type="GotDotNet.ApplicationBlocks.Data.SqlServer"/>
    </daabProviders>
    <system.web>
<pages validateRequest="false" />
  <!--  动态调试编译    -->
<compilation defaultLanguage="c#" debug="true">
    <assemblies>
        <add assembly="System.Data.OracleClient, Version=2.0.0.0,
Culture=neutral, PublicKeyToken=B77A5C561934E089"/>
        <add assembly="System.Design, Version=2.0.0.0, Culture=neutral,
PublicKeyToken=B03F5F7F11D50A3A"/>
        <add assembly="System.Windows.Forms, Version=2.0.0.0,
Culture=neutral, PublicKeyToken=B77A5C561934E089"/>
    </assemblies>
</compilation>
<!--  自定义错误信息    -->
<customErrors mode="RemoteOnly"/>
<!--  身份验证    -->
<authentication mode="Windows"/>
<!--  授权    -->
<authorization>
    <allow users="*"/>
        <!-- 允许所有用户 -->
</authorization>
<!--  应用程序级别跟踪记录    -->
<trace enabled="false" requestLimit="10" pageOutput="false"
traceMode="SortByTime" localOnly="true"/>
<!--  会话状态设置    -->
<sessionState mode="InProc" stateConnectionString="tcpip=127.0.0.1:42424"
sqlConnectionString="data source=127.0.0.1;Trusted_Connection=yes"
cookieless="false" timeout="60"/>
<!--  全球化    -->
<globalization requestEncoding="gb2312" responseEncoding="gb2312"/>
```

```
<xhtmlConformance mode="Legacy"/>
</system.web>
<appSettings>
<!--    连接字符串-->
    <add key="ConnectionString" value="Data
Source=skymouse.lamost.org;Initial catalog=SUN;user id=sun;password=sun"/>
    <add key="ConnectionString2" value="Provider=SQLOLEDB;Data
Source=skymouse.lamost.org;Initial catalog=SUN;user id=sun;password=sun"/>
    <add key="source" value=""/>
    <add key="RegistryWS.registry"
value="http://nvo.stsci.edu/voregistry/registry.asmx"/>
    <add key="SDSSConeSearch.sdssConeSearch"
value="http://casjobs.sdss.org/vo/dr4cone/sdssConeSearch.asmx"/>
    <add key="SkyMouse.NEDWS.NED"
value="http://voservices.net/NED/ws_v1_0/NED.asmx"/>
    <add key="SkyMouse.SimbadName.Reference"
value="http://vizier.hia.nrc.ca:8080/axis/services/Sesame"/>
    <add key="SkyMouse.Spectrum.search"
value="http://voservices.net/spectrum/ws_v2_1/search.asmx"/>
    <add key="SkyMouse.VizieCatalogWS.Reference"
value="http://cdsws.u-strasbg.fr/axis/services/VizieR"/>
    <add key="HtmlDoc" value="C:\Program Files\HTMLDOC\ghtmldoc.exe"/>
    <add key="SiteUrl" value="http://localhost/skymouse/"/>
    <add key="NVOVoregistry.registry"
value="http://nvo.stsci.edu/voregistry/registry.asmx"/>
</appSettings>
</configuration>
```

C、 调用天文服务常见问题的解决

在 SkyMouse 项目的开发中，服务器端天文服务的调用遇到了很多问题。总体来说，天文服务的调用存在以下两个主要问题：不同平台的天文 Web Service 的调用问题和天文 Web Service 调用时的性能问题。

下面分别针对以上两个问题进行分析。

不同平台的天文 Web Service 互操作问题

目前天文 Web Service 主要还是在两个平台下开发的，一种是以 Java 为开发语言，典型的应用是 Apache 的 SOAP 服务 Axis，法国的 SIMBAD 的所有 Web Service 都是在这个平台下开发的。另一种是在.NET[6]框架下开发的，美国

NVO[7]下开发的一些 Web Service 应用的是.NET 技术。Web Service 开发的初衷是所有的服务都通过 SOAP 协议调用，通过这种方式实现平台的无关性，然而在实际应用时，我们却经常碰到两种平台的 Web Service 不能相互调用的问题。这个问题的根源就在于两个平台的某些数据类型不能匹配，详细地说.NET 平台的 WSDL 文件是自我描述的，任何语言都可以通过这个描述选择自己平台的数据类型，而由于 Apache 的 SOAP 服务需要每个参数都要有一个详尽的描述，Java 编写的 Web Service 经常会加入它自己特有的数据类型，例如 Datahandler 类型，在.NET 平台里就没有，而几乎所有的 Apache Web Service 的附件下载都是通过这个类型实现的，当调用的 Web Service 的参数或结果出现这种类型时，跨平台的操作是无法进行的。

天文 Web Service 调用时的性能考虑

众所周知，Web Service 的调用是比较慢的，这种情况在同一台服务器同时调用多个 Web Service 时尤其明显，我们可以从以下三个方面进行优化：

A HTTP 双连接限制

HTTP 规范表明，一个 HTTP 客户端与任一服务器最多可以同时建立两个 TCP 连接。这可以防止单个浏览器在浏览某个页面时，由于连接请求过多而使服务器负载过重。此时，浏览器将仅创建 2 个连接，然后通过这两个管道开始发送 120 个 HTTP 请求，而不是创建 120 个 TCP 连接并通过每个连接来发送 HTTP 请求。因为 HTTP 双连接限制，如果同时有 50 个用户访问一个服务，我们不得不为其中的 2 个用户进行一次 SIMBAD 的 Web Service 调用，这将会导致会有 48 个用户等待两个管道中的一个空闲下来。

B 线程池限制

.NET 处理传入的请求的方式是通过一个称为进程线程池的一组线程为其提供服务。正常情况下，请求传入后，池中某个空闲的线程将为其提供服务。这里的问题在于，进程线程池不会创建无数个线程来处理大量的请求。具有最大线程数限制是一件好事，因为如果我们无限地创建线程，计算机上的全部资源将只能用来管理这些线程了。通过限制所能创建的线程数，我们可以把线程管理的系统开销保持在一个可控的水平。如果某个请求传入时线程池中的所有线程都被占用，则该请求将排队等候，在忙线程完成任务后，空闲出来的线程

才能处理新请求。此方法实际上比切换到某个新线程更有效，因为不需要在请求之间进行线程切换。但存在的问题是，如果线程的使用效率不高（尤其是在非常忙的 Web 服务器上），则等候的请求队列会变得很大。

在 SkyMouse 的开发过程中，我们针对上面两个问题提出了下述解决方案。

1　不同平台的天文 Web Service 互操作

这个问题的解决办法是我们在 Java 下的 Web Service 时，必须要提供一个服务中参数类型说明，例如：

```
  <isd:service xmlns:isd="http:"//xml.apache.org/xml-soap/deployment
  id="urn:MyService "><isd:mappings>
    <isd:map encodingStyle="http:"//schemas.xmlsoap.org/soap/encoding/

xmlns:x=""qname="x:num1"xml2JavaClassName="org.apache.soap.encoding.soapenc.D
oubleDeserializer"/><isd:map
encodingStyle="http:"//schemas.xmlsoap.org/soap/encoding/"
xmlns:x="" qname="x:num2"
xml2JavaClassName="org.apache.soap.encoding.soapenc.DoubleDeserializer"/>

    </isd:mappings>
    </isd:service>
```

但是这种方式在实际应用时还是会有一些问题，对于简单的数据类型我们可以通过这种方式解决，可是复杂的数据类型我们还是无能为力。这个问题的根本解决办法是我们在开发天文 Web Service 的时候要考虑到跨平台操作的问题，尽可能使用两个平台都能接受的参数和结果的数据类型（事实上几乎所有的天文服务都可以只用基本数据类型来完成），这样的话，我们就可以很方便的跨平台调用了。遗憾的是，很多服务已经开发完成，我们不可能要求服务的提供者再开发一遍。不过令人庆幸的是，计算机界也认识到了这个问题，由美国 Accenture、美国 SUN、美国 BEA Systems、富士通、美国惠普、美国 IBM、美国英特尔、美国微软、美国甲骨文以及德国 SAP AG 等 10 家公司已经成立了旨在促进 Web Service 普及的团体网络服务互操作性组织(WS-I) [8]。该组织致力于在各种平台、应用软件以及编程语言之间开发 / 推广通用的 Web Service。WS-I 的宪章提升了跨越平台，应用程序和编程语言的 Web services 的一致性和可靠的互操作性。WS-I 的关键成分包括 Profiles，示例应用程序和测试工具。

Web Services 互操作组织(WS-I)已经制定出了两个工作宪章，一个是 Basic Profile ；另外一个是 Reliable Secure Profile。进一步的标准和规范正在制定中。随着规范的逐步制定，天文服务的跨平台操作将会很快得到圆满的解决。

2 天文 Web Service 调用时的性能优化

A HTTP 双连接限制

对于我们的系统，这部分由于规范而导致的限制是没有办法解决的。为了尽快地提高性能，我们采取了服务缓存的技术，即对某个服务返回的结果事先保存到数据库，用户调用的时候直接先访问缓存如果没有缓存再调用服务。

B 线程池限制

在 SkyMouse 中，这种情况我们通过修改服务器的配置文件加以解决。我们在 Machine.config 中进行配置，当空闲的线程数低于所设置的限制时，将禁止使用线程池中的线程来处理传入的 HTTP 请求。

C 使用异步的 Web Service 调用和异步 PreRequestHandler 执行

Web Service 有同步异步的方式。传统的调用是同步的方式，这种方式发出调用请求后，线程就一直等待服务返回结果。同步的调用是阻塞的，性能也是最差。我们可以想象如果某个天文的 Web Service 出现了问题或者返回的数据量非常大，那我们宝贵的系统线程就会出现假死的状态，这种状况是不能忍受的。所以，对于天文的 Web Service，我们尽量都选择采用异步调用的方式。异步调用的原理是这样的，当发出一个 Web Service 的调用请求后，线程不等待而是继续执行，直到服务返回了结果，线程才开始对结果进行处理。异步调用是不阻塞的，它能够使线程不用等待 Web 服务调用完成即释放线程以便处理更多的请求，性能比同步好很多。

对于 SkyMouse 系统，对性能的要求是至关重要的。我们不但采取了异步调用，还针对.NET 平台把所有的异步调用修改为异步 PreRequestHandler 调用。普通的异步调用是通过 Begininvoke 和与之配对的 EndInvoke 来实现的，这是最基本的异步调用。而 SkyMouse 平台采用了更高级的异步调用--异步 PreRequestHandler 调用来实现。PreRequestHandler 是通过绑定 PreRequestHandlerExecute 事件，在某个特定请求的 HttpHandler 被调用之前就调用。这种调用对服务器的性能优化是非常大的，在我们的测试之中，在 60 秒

钟内从 100 个虚拟客户端连续发送请求，同步调用只能完成大概 200 个左右，异步调用可以完成 300 个左右，而通过 PreRequestHandler 方式可以完成 1800 个左右，这对性能的提升无疑是巨大的。

4.3 数据库设计

SkyMouse 系统数据库基于 SQL Server2000,服务器端程序通过 OLEDB 方式连接数据库。所有表列表如下, 具体每个表的功能说明参考附录 2:

表名	说明
AdminUser	管理员信息表
Catalog	服务类型信息表
CatalogType	大类信息表
Contact	建议信息表
Dict	翻译字典表
KeyInfo	查询关键字历史记录表
ObjectInfo	天体基本信息缓存表
Status	服务状态字典表
SYS_Table_Base	系统基本信息表
SYS_Table_ColTypeCode	系统维护字典表
SYS_Table_DrpBase	系统维护下拉表
SYS_Table_DrpChild	系统子表字典表
UserType	用户类型表
WS_user	自定义服务表
WSAddInfo	天文服务附加信息表
WSBasicInfo	天文服务基本信息表
WSCache	WS 信息缓存表

表 4.1:SkyMouse 所有表列表

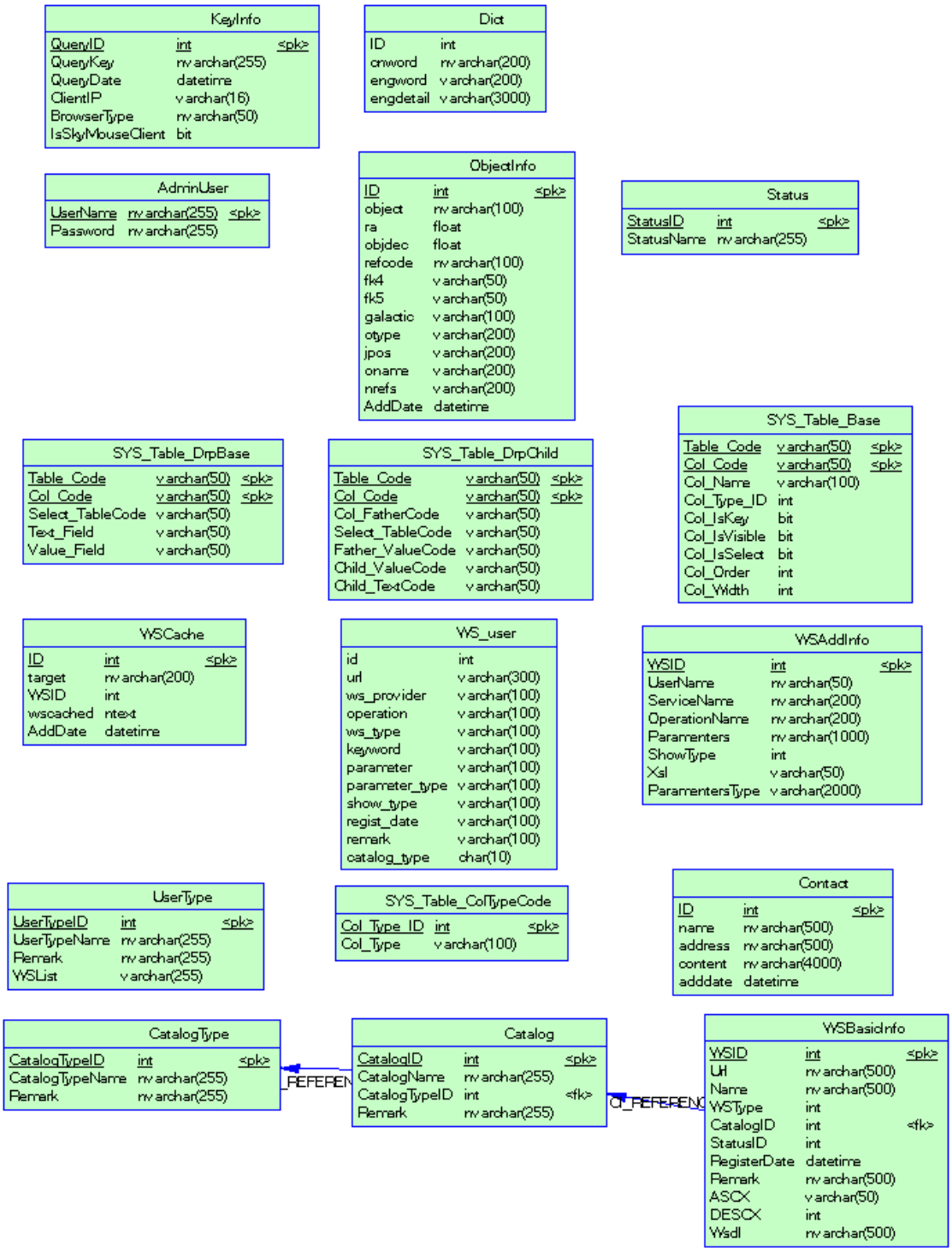

图 4.12：SkyMouse 数据库设计

第5章 总结与展望

本论文从中国虚拟天文台的产生说起，分析了国际天文界数据访问与互操作方向的现状，介绍了 IVOA 标准和网格等相关技术，着重讲述了 SkyMouse 系统的设计和实现方案。论文的主要内容如下：

第一章介绍了虚拟天文台的产生背景，中国虚拟天文台目前主要的研究方向，并分析了服务访问与互操作在中国虚拟天文台建设中的重要性以及必要性和可行性。同时也综述了各国在数据访问与互操作研究上的成果。

第二章概述了天文服务整合与互操作方面的相关技术。其中包括网格技术、中国虚拟天文台的体系结构和 IVOA 的数据访问与互操作标准。

第三章着重的讲述了 SkyMouse 系统的概要描述与概要设计。

第四章讲述了 SkyMouse 系统的详细设计与实现。

目前世界各国虚拟天文台都在从事天文数据结点和天文数据访问门户的建设，越来越多的天文服务被开发出来。如何能使它们的效率更高、更稳定的整合到一起，实现 Google 似的天文访问统一门户将成为今后几年这个研究方向关心的重点。

参考文献

[1] 都志辉,陈渝,刘鹏.网格计算.北京：清华大学出版社,2002

[2] 崔辰州. 中国虚拟天文台系统设计：[学位论文].北京：中国科学院研究生院

[3] [IVOA] http://www.ivoa.net/.

[4] [China-VO] http://www.china-vo.org/.

[5] [LAMOST] http://www.lamost.org

[6] [SkyQuery] http://www.skyquery.net/

[7] [OpenSkyQuery] http://www.openskyquery.net/

[8] [SkyCom] http://www.skycom.jp/

[9] Miguel L. Bote-Lorenzo, Yannis A. Dimitriadis.

Grid Characteristics and Uses: a Grid Definition.

http://www.chinagrid.net/dvnews/upload/2005_03/05031923425425.pdf

[10] Ian Foster Carl Kesselman Steven Tuecke. The Anatomy of the Grid.

http://www.globus.org/research/papers/anatomy.pdf

[11] 华宇. 浅析基于 WSRF 的网格体系结构.

http://www.chinagrid.net/grid/paperppt/GridArchi.ppt

[12] 肖侬. 网格体系结构之 OGSA.

http://www.chinagrid.net/grid/paperppt/bigfile/paperppt/XiaoN/Arch2.ppt

[13] 肖侬 网格体系结构之 WSRF

http://www.chinagrid.net/grid/paperppt/bigfile/paperppt/XiaoN/Arch3.ppt

[14] 陈渝. 网格平台 Globus 的核心技术.

http://www.chinagrid.net/dvnews/upload/2005_02/05022813117467.doc

[15] ChaoLiu, DanWang, BoLiu. et.al. An Astronomical Data Mining Application Framework in China-VO SPIE accepted.

[16] [ADQL] http://www.ivoa.net/internal/IVOA/IvoaVOQL/ADQL-0.91.pdf

[17] [VOTABLE] http://www.us-vo.org/VOTable/

[18] [SKYNODE]

http://www.ivoa.net/internal/IVOA/IvoaVOQL/SkyNodeInterface-0.9.pdf

[19] [2MASS] http://www.ipac.caltech.edu/2mass/

[20] Cardelli, J. et. al. The relationship between infrared, optical, and ultraviolet extinction。 ApJ, 345, p245-256, 1989

[21] Koornneef, J. Near-infrared photometry. II - Intrinsic colours and the absolute calibration from one to five micron. A&A, 128, p84-93, 1983

[22] Blaauw, Adriaan. The O Associations in the Solar Neighborhood. A&A, 2, p213-246, 1964

[23] Morgan, W. et. al. Studies in Galactic Structure. I. a Preliminary Determination of the Space Distribution of the Blue Giants. ApJ, 118, p318-322, 1953

[24] Morgan, W. et. al. Some features of galactic structure in the neighborhood of the Sun. AJ, 57, p3, 1952

[25] Vallée, Jacques P. The Spiral Arms and Interarm Separation of the Milky Way: An Updated Statistical Study. AJ 130,p569-575, 2005

附录 1：屏幕取词核心代码

对于大多数的 Windows 应用程序来说，如果要取词，我们需要截获的是“gdi32.dll”中的“TextOutA”函数。 我们写一个自己的 myTextOutA 函数，如：

```
bool WinAPI myTextOutA(hdc hdc, int nxstart, int nystart, lpcstr lpszstring,int cbstring)
{
// lpszstring 就是我们取得的文字，这里会发到 SkyMouse 客户端进行处理
// 然后调用正版的 TextOutA 函数
}
```

把这个函数放在安装了钩子的动态连接库中，然后调用我们最后给出的 hookimportfunction 函数来截获进程对 TextOutA 函数的调用，跳转到我们的 myTextOutA 函数，完成对输出字符串的捕捉。

核心函数 hookimportfunction 的源代码:

```
#include <crtdbg.h>
#define makeptr(cast, ptr, addvalue) (cast)((dword)(ptr)+(dword)(addvalue))
tyPEdef struct tag_hookfuncdesc
{
lpcstr szfunc; // the name of the function to hook.
proc pproc; // the procedure to blast in.
} hookfuncdesc , * lphookfuncdesc;
pimage_import_descriptor getnamedimportdescriptor(hmodule hmodule, lpcstr szimportmodule);
// 主函数
bool hookimportfunction(hmodule hmodule, lpcstr szimportmodule,
lphookfuncdesc pahookfunc, proc* paorigfuncs)
{
_assert(szimportmodule);
_assert(!isbadreadptr(pahookfunc, sizeof(hookfuncdesc)));
#ifdef _debug
if (paorigfuncs) _assert(!isbadwriteptr(paorigfuncs, sizeof(proc)));
_assert(pahookfunc.szfunc);
_assert(*pahookfunc.szfunc != '\0');
_assert(!isbadcodeptr(pahookfunc.pproc));
#endif
if ((szimportmodule == null) || (isbadreadptr(pahookfunc, sizeof(hookfuncdesc))))
{
_assert(false);
setlasterrorex(error_invalid_parameter, sle_error);
return false;
}
// 监测当前模块是否是在 2GB 虚拟内存空间之上
if (!isnt() && ((dword)hmodule >= 0x80000000))
{
_assert(false);
```

```
setlasterrorex(error_invalid_handle, sle_error);
return false;
}
// 清零
if (paorigfuncs) memset(paorigfuncs, null, sizeof(proc));
// 调用 getnamedimportdescriptor()函数,得到 hmodule
// 截获的函数所在的 dll 模块的引入描述符(import descriptor)
pimage_import_descriptor pimportdesc = getnamedimportdescriptor(hmodule,
szimportmodule);
if (pimportdesc == null)
return false;
// 从 dll 模块中得到原始的 thunk 信息
pimportdesc->originalfirstthunk
// 指针来访问引入函数名等信息
pimage_thunk_data porigthunk = makeptr(pimage_thunk_data, hmodule,
pimportdesc->originalfirstthunk);
// 所有的引入信息,所以真正的截获实际上正是在这里进行的
pimage_thunk_data prealthunk = makeptr(pimage_thunk_data, hmodule,
pimportdesc->firstthunk);
// 穷举 image_thunk_data 数组,寻找我们需要截获的函数
while (porigthunk->u1.function)
{
// 只寻找那些按函数名函数
if (image_ordinal_flag != (porigthunk->u1.ordinal & image_ordinal_flag))
{
// 得到引入函数的函数名
pimage_import_by_name pbyname = makeptr(pimage_import_by_name, hmodule,
porigthunk->u1.addressofdata);
// 如果函数名以 null 开始,跳过,继续下一个函数
if ('\0' == pbyname->name[0])
continue;
// bdohook 用来检查是否截获成功
bool bdohook = false;
// 检查是否当前函数是我们需要截获的函数
if ((pahookfunc.szfunc[0] == pbyname->name[0]) &&
(strcmpi(pahookfunc.szfunc, (char*)pbyname->name) == 0))
{
if (pahookfunc.pproc)
bdohook = true;
}
if (bdohook)
{
//改变这一块虚拟内存的内存保护状态
memory_basic_information mbi_thunk;
```

```
virtualquery(prealthunk, &mbi_thunk, sizeof(memory_basic_information));
_assert(virtualprotect(mbi_thunk.baseaddress, mbi_thunk.regionsize,
page_readwrite, &mbi_thunk.protect));
// 保存截获的函数的正确跳转地址
if (paorigfuncs)
paorigfuncs = (proc)prealthunk->u1.function;
// 将 image_thunk_data 数组中的函数跳转地址改写为 SkyMouse 的函数地址!
// 以后所有进程对这个系统函数的所有调用都将成为对我们自己编写的函数的调用
prealthunk->u1.function = (pdword)pahookfunc.pproc;
//将虚拟内存改回原来的保护状态
dword dwoldprotect;
_assert(virtualprotect(mbi_thunk.baseaddress, mbi_thunk.regionsize,
mbi_thunk.protect, &dwoldprotect));
setlasterror(error_success);
return true;
}
}
porigthunk++;
prealthunk++;
}
return true;
}
// getnamedimportdescriptor 函数的实现
pimage_import_descriptor getnamedimportdescriptor(hmodule hmodule, lpcstr
szimportmodule)
{
_assert(szimportmodule);
_assert(hmodule);
if ((szimportmodule == null) || (hmodule == null))
{
_assert(false);
setlasterrorex(error_invalid_parameter, sle_error);
return null;
}
pimage_dos_header pdosheader = (pimage_dos_header) hmodule;
if (isbadreadptr(pdosheader, sizeof(image_dos_header)) ||
(pdosheader->e_magic != image_dos_signature))
{
_assert(false);
setlasterrorex(error_invalid_parameter, sle_error);
return null;
}
// 取得 PE 文件头
pimage_nt_headers pntheader = makeptr(pimage_nt_headers, pdosheader,
```

```
pdosheader->e_lfanew);
// 检测是否 PE 映像文件
if (isbadreadptr(pntheader, sizeof(image_nt_headers)) ||
(pntheader->signature != image_nt_signature))
{
_assert(false);
setlasterrorex(error_invalid_parameter, sle_error);
return null;
}
// 检查 PE 文件的引入段
if (pntheader->optionalheader.datadirectory[image_directory_entry_import].virtualaddress ==
0)
return null;
pimage_import_descriptor pimportdesc = makeptr(pimage_import_descriptor, pdosheader,
pntheader->optionalheader.datadirectory[image_directory_entry_import].virtualaddress);
// 穷举 pimage_import_descriptor 数组寻找我们需要截获的函数所在的模块
while (pimportdesc->name)
{
pstr szcurrmod = makeptr(pstr, pdosheader, pimportdesc->name);
if (stricmp(szcurrmod, szimportmodule) == 0)
break; // 找到!中断循环
// 下一个元素
pimportdesc++;
}
if (pimportdesc->name == null)
return null;
return pimportdesc;
}
// isnt()函数的实现
bool isnt()
{
osversioninfo stosvi;
memset(&stosvi, null, sizeof(osversioninfo));
stosvi.dwosversioninfosize = sizeof(osversioninfo);
bool bret = getversionex(&stosvi);
_assert(true == bret);
if (false == bret) return false;
return (ver_platform_Win32_nt == stosvi.dwplatformid);
}
```

附录 2：数据库表结构及说明

AdminUser：管理员信息表

列名	说明	类型
UserName	管理员账号	nvarchar(255)
Password	管理员密码	nvarchar(255)

Catalog：服务类型表

列名	说明	类型
CatalogID	ID	int
CatalogName	类型名	nvarchar(255)
CatalogTypeID	大类	int
Remark	注释	nvarchar(255)

CatalogType：服务小类表

列名	说明	类型
CatalogTypeID	ID	int
CatalogTypeName	大类名	nvarchar(255)
Remark	注释	nvarchar(255)

Contact：用户建议表

列名	说明	类型
ID	ID	int
name	姓名	nvarchar(500)
address	地址	nvarchar(500)
content	建议内容	nvarchar(4000)
adddate	时间	datetime

Dict：系统字典表

列名	说明	类型
ID	ID	int
cnword	中文	nvarchar(200)
engword	英文	varchar(200)
engdetail	英文解释	varchar(3000)

KeyInfo：关键字查询记录表

列名	说明	类型
QueryID	ID	int
QueryKey	查询关键字	nvarchar(255)
QueryDate	查询日期	datetime
ClientIP	IP	varchar(16)
BrowserType	浏览器类型	nvarchar(50)
IsSkyMouseClient	是否使用 Client 端	bit

ObjectInfo：天体基本信息缓存表

列名	说明	类型
ID	ID	int
object	天体名	nvarchar(100)
ra	ra	float
objdec	dec	float
refcode	refcode	nvarchar(100)
fk4	fk4	varchar(50)
fk5	fk5	varchar(50)
galactic	galactic	varchar(100)
otype	otype	varchar(200)
jpos	jpos	varchar(200)
oname	oname	varchar(200)
nrefs	nrefs	varchar(200)
AddDate	添加时间	datetime

SYS_Table_Base：系统表信息记录表

列名	说明	类型
Table_Code	表名	varchar(50)
Col_Code	列名	varchar(50)
Col_Name	列显示名	varchar(100)
Col_Type_ID	类型	int
Col_IsKey	是否为主键	bit
Col_IsVisible	可见	bit
Col_IsSelect	下拉	bit
Col_Order	排序	int
Col_Width	列宽	int

SYS_Table_ColTypeCode：下拉框记录表

列名	说明	类型
Col_Type_ID	类型 ID	int
Col_Type	类型	varchar(100)

SYS_Table_DrpBase：下拉框父表

列名	说明	类型
Table_Code	表名	varchar(50)
Col_Code	列名	varchar(50)
Select_TableCode	下拉列名	varchar(50)
Text_Field	文本	varchar(50)
Value_Field	值	varchar(50)

SYS_Table_DrpChild：下拉框子表

列名	说明	类型

Table_Code	表名	varchar(50)
Col_Code	列名	varchar(50)
Col_FatherCode	父列名	varchar(50)
Select_TableCode	选取表名	varchar(50)
Father_ValueCode	父值名	varchar(50)
Child_ValueCode	子值名	varchar(50)
Child_TextCode	子文本	varchar(50)

Status：服务状态字典表

列名	说明	类型
StatusID	ID	int
StatusName	状态描述	nvarchar(255)

UserType:用户预制类型表

列名	说明	类型
UserTypeID	ID	int
UserTypeName	用户类型	nvarchar(255)
Remark	标注	nvarchar(255)
WSList	默认的天文服务	varchar(255)

WSAddInfo：动态调用服务信息记录表

列名	说明	类型
WSID	ID	int
UserName	创建人	nvarchar(50)
ServiceName	服务名	nvarchar(200)
OperationName	方法名	nvarchar(200)
Paramenters	参数列表	nvarchar(1000)
ShowType	显示类型	int
Xsl	XSL 文件地址	varchar(50)
ParamentersType	参数类型列表	varchar(2000)

WSBasicInfo：服务基本信息表

列名	说明	类型
WSID	ID	int
Url	服务地址	nvarchar(500)
Name	名称	nvarchar(500)
WSType	服务类型	int
CatalogID	所属类型	int
StatusID	状态	int
RegisterDate	添加时间	datetime
Remark	注释	nvarchar(500)
ASCX	显示控件	varchar(50)
DESCX	排序	int

Wsdl	wsdl 文件地址	nvarchar(500)

WSCache：服务缓存表

列名	说明	类型
ID	ID	int
target	查询关键字	nvarchar(200)
WSID	WSID	int
wscached	缓存值	ntext
AddDate	加入时间	datetime

WS_user：标准服务基本信息表

列名	说明	类型
id	ID	int
url	WSDL 文件地址	varchar(300)
ws_provider	VoRegistry	varchar(100)
operation	方法名	varchar(100)
ws_type	WSDL/CGI	varchar(100)
keyword	服务类型	varchar(100)
parameter	参数列表	varchar(100)
parameter_type	参数类型列表	varchar(100)
show_type	显示类型	varchar(100)
regist_date	注册时间	varchar(100)
remark	注释	varchar(100)
catalog_type	类别	char(10)

发表论文目录

1．孙华平,崔辰州,赵永恒. 天文服务的统一调用及其在 SkyMouse 系统中的实现. 天文研究与技术, 2007 已接受。

2．崔辰州,孙华平,罗宇 屏幕取词技术及其在 SkyMouse 中的应用 天文研究与技术 2007 审稿中。

致　谢

三年时光转眼即逝，在此我要我衷心地感谢我的导师赵永恒和指导我工作的崔辰州老师，感谢他们近年来对我无微不至的关怀和孜孜不倦的教诲。

首先，最为感谢的是我的指导老师赵永恒研究员。感谢他两年多来对我无微不至的关怀和孜孜不倦的教诲！两年来，无论在理论学习阶段，还是科研项目过程中，赵老师在学术上给予我启迪，拓宽我研究思路，引导我学术思维、教给我科学研究的方法。虽然赵老师日常工作非常繁忙，但他仍然在百忙之中对我不解的问题进行细致地指导和帮助，提供了优良的软硬件环境和学习环境，十分感谢赵老师对我研究能力的培养。导师的言传身教是我两年来最大的收获，其为人之谦和大度，其为学之严谨让我长久铭记。我将继续刻苦钻研、不断进取，只有这样才能不负老师之期望。

同样在这里我要深深地感谢我的指导老师崔辰州博士，我的每一步工作都是在他的指引下完成的。我从开始选题、资料收集、开题、研究和撰写文章的每一个环节，无不得到崔辰州博士的悉心指导和帮助，全文字里行间无不浸透了崔老师之心血。最让我难以忘记还是崔老师严谨的治学态度、对事业执著的追求以及平易近人的品质都是我学习的榜样。而且他还是一位十分尊重学生想法的老师，这一点更让我敬佩不已。

同时也感谢张彦霞老师，张老师为人心胸宽阔、治学严谨、待人热忱。虽然我跟张老师相处的时间不多，但是从相处的一段时间我已经可以深深地感受到张老师的为人。非常感谢她在工作上给予的帮助与支持。

感谢杜红荣老师。从我走进天文台开始，研究生期间的生活、学习到毕业分配，她都给我给予了很大的帮助。

我还要感谢 LAMOST 项目组的袁辉老师和陈英老师在生活上的帮助。

感谢刘超博士在理论研究、算法实现和程序设计方面给我提供了很大的帮助。感谢经常一起讨论的王丹、吴潮、尹红星、李会贤、高丹、田海俊、郑征、张旭、路勇、罗宇同学。感谢天津大学的亓大志对项目的辛勤付出。

最后，我要感谢我的父母和爱人。正是他们默默无闻的奉献和一贯的支持和

鼓励，才使我有信心去克服一切困难，有充足的时间和精力去完成学业。他们全心的付出和无言的关爱始终是我前进的巨大动力。

感谢所有关心过我的人。