天津大學

毕业设计论文

学　　院 电子信息工程
专　　业 计算机科学与技术
年　　级 2001 级
姓　　名 徐祯
指导教师 孙济洲 教授

2005 年　6 月 16 日

天津大學

毕业设计论文

题目：**天文巡天数据管理发布系统的设计与实现**

学生姓名　徐祯

学院名称　电子信息工程

专　　业　计算机科学与技术

学　　号　3001204105

指导教师　孙济洲

职　　称　教授

摘　　要

随着天文观测能力的增强，天文观测活动产生越来越庞大的数据量。大量数据以文件的形式记录和保存，对数据信息的统计和查询是非常耗费时间和精力的。针对该问题，我们开发了天文巡天数据管理发布系统。我们以北京国家天文台现有的巡天数据为实验数据，分三个步骤完成了整个系统的设计和实现。首先利用数据库搭建起数据平台，接着解决了数据平台如何从文件中自动导入数据的问题，最后实现了数据发布平台以及该平台上的数据查询、格式转换、文件下载等功能。

本文自上而下，介绍了天文巡天数据管理发布系统从设计到具体实现的主要过程。从系统开发的背景开始，先简单介绍了巡天数据的文件格式，然后在系统整体分析设计的基础上逐一介绍了数据平台搭建模块、数据导入模块以及数据发布模块的设计过程与具体实现。该系统中各个模块的开发都考虑到良好的兼容性、可移植性和可扩展性，力争做到灵活、高效。

关键词： 巡天数据；数据平台搭建；数据导入；数据发布

ABSTRACT

With the enhancement of astronomical observation capability, observation activities bring us more and more astronomical data. A lot of data are recorded and saved in the form of files, so it is a large waste of time and energy to execute statistic operation and query. Aiming at the problem we design the management and release system for astronomical Sky Survey data. Considering the Sky Survey data in Beijing National Astronomical Observatories as experimental data, we finally design and realize the system in three steps. We first utilize database to establish the data platform, then solve the problem how the data platform automatically imports data from files, and finally realize the data release platform as well as the functions such as data query, format transformation and file download on it.

The paper introduces the primary process of developing the system from design to realization. We begin at the background of the system development. After simply introducing the file format of the Sky Survey data, on the basis of holistic system analysis and design, we go into design process and detailed realization of data platform establishment module, data import module and data release module. Good compatibility, migratability and extensibility is considered among all the modules aiming for agility and efficiency.

Key words： Sky Survey data ;data platform establishment ;data import ;data release

目　　录

第一章　　绪论

1.1 项目背景

随着天文望远镜及终端设备的设计与制造技术不断提高，天文观测能力大大增强，天文学已从古老的光学观测变为全波段的天文学，并正在进入一个“数据雪崩”时代。

国家天文台立足于国内现有的条件，对兴隆观测基地一台基本闲置的60/90厘米施密特望远镜进行了全面的技术改造，配备了2048×2048像元的大面积CCD, 15个中等带宽滤光片和多种窄带滤光片，组成了在国际上极富创新特色的大视场、高精度、低分辨的大样本天体光谱巡天系统。这套系统引起了国际天文界的重视，并由此建立了以我为主的BATC（Beijing-Arizona-Taiwan-Connectitut）合作计划。该计划自1995年正式观测运作以来，开辟了从小行星、星团、星系、星系团、类星体到宇宙大尺度结构等一系列有特色的前沿研究领域，并承担了中国科学院重大项目和国家基金会重点研究项目，其中部分研究工作在国际上已取得令人瞩目的成果。

自95年至今，该系统已经观测并记录了大量巡天数据，这些数据有着丰富的科研价值，如能及时发布，对于广大的天文研究者将是一件无比振奋的事情。

1.2 项目概述

BATC巡天活动产生的数据记录在后缀名为fit的文件中，这种文件是遵循FITS标准的。FITS (Flexible Image Transport System)是国际天文学会1982年确定的世界各天文台之间用于数据传输、交换的统一标准格式[1]，具体规则会在后面章节中介绍。目前这些数据全部存储在服务器的archive目录下，该目录还有两级子目录，所有的数据文件都在最底层的子目录中。随着观测数据的不断增加，archive目录中会随时添加进新的子目录。这种存储方式既不利于内部管理，也不利于对外发布。所以我们要设计一个肩负数据的管理和发布功能的完整的系统，改善目前的状况。

总的来说，我们的任务包括三个方面：第一，要搭建一个数据平台，这个数据平台要能存储所有的BATC巡天数据的信息，也就是说，是目前所有数据文件的一个完整映射。此外，它还要方便管理者管理数据和发布数据。不管怎样，都要求我们先详细了解FITS文件的格式和数据含义，才有可能搭建合适的数据平台。第二，我们要建立一个数据导入系统，负责自动将数据文件信息存入已搭建好的数据平台中。第三，搭建数据发布平台，这一部分至少应包括提供查询页面、处理查询请求、

对查询结果进行数据转换、提供文件下载等功能的实现。

在该项目开发过程中要注意下面几个问题：首先，对FITS文件的详细了解是做好该项目的前提和重要保障。其次，天文数据属于科研数据，对数据的规范性要求很高。所以数据平台中对数据的记录一定要保证数据原始性，发布平台中的数据格式转化也要严格遵守相关规范。最后，考虑到该服务要在LINUX平台下实现，服务的使用者也多是使用LINUX等开放平台的科研工作者，因此在程序开发过程中选用的技术最好是能被这些平台和平台上层软件所支持的成熟技术，实现对各平台的兼容性。因此选用JAVA作为主要编程语言。

1.3 本文的组织结构

本文从项目的背景和我们的任务讲起，逐步讨论整个系统的设计和实现细节。

第二章开始对系统整体框架的设计，将系统划分为若干模块并归纳了各模块的任务，确定了系统功能，并简单介绍所用的开发工具。

第三章在了解BATC巡天数据的文件格式的基础上，重点描述数据平台的设计结果和解决的主要问题。

第四章介绍了数据导入，主要是类的设计和实现。

第五章探讨如何搭建数据发布平台，给出了查询界面、请求处理和数据转换部分的具体实现，以及相关的配置信息。

第六章是对已完成工作的总结和对未来的展望。

第二章　　系统分析设计

2.1 系统结构和模块划分

根据需求分析的结果，我们得到了如图 2-1 所示的系统结构图。从该图中不难看出，整个系统大致分为三个模块：数据平台搭建模块、数据导入模块、数据发布模块。

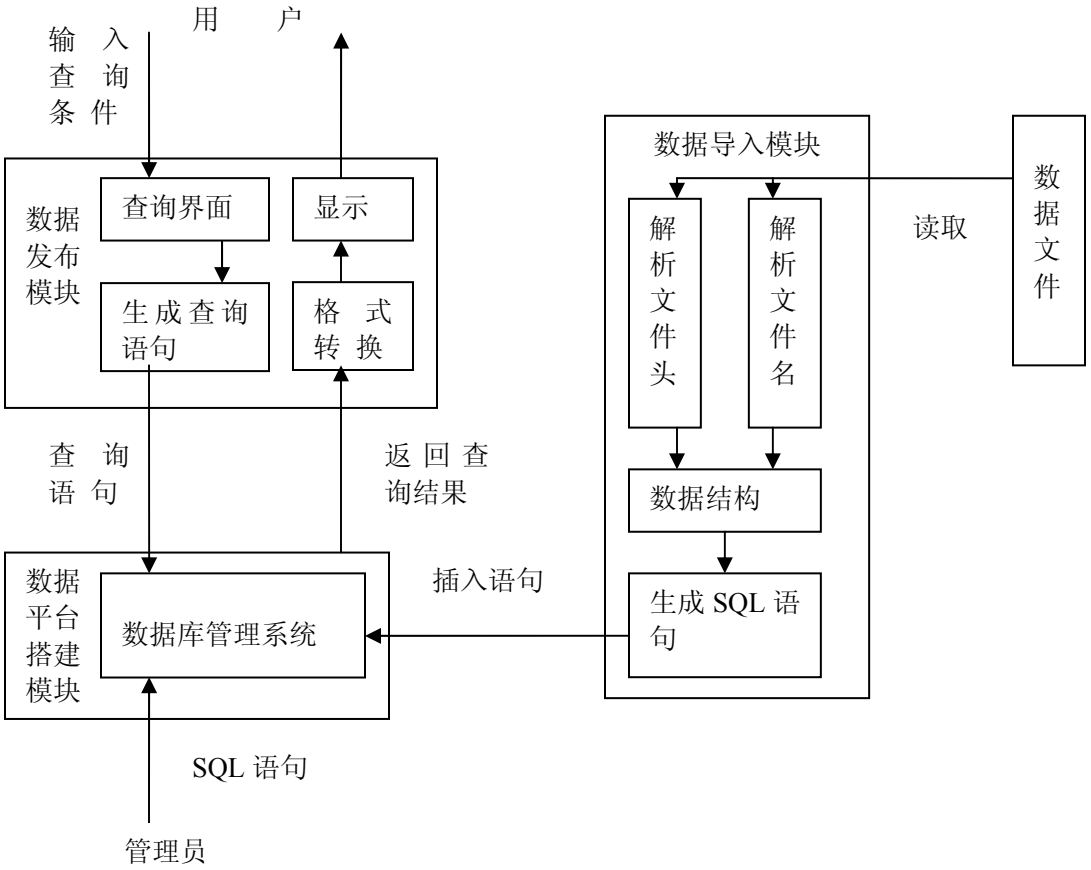

图 2-1　　系统结构图

2.2 各模块的工作

数据平台搭建模块负责搭建数据平台，这是进行后两步工作的基础。数据平台的搭建要借助于关系型数据库，主要的工作是创建数据库，设计和创建表，设置权限，建立索引等。

数据文件信息如何录入到搭建好的数据平台中，这一过程是靠数据导入模块来

实现的。该系统是使用 JAVA 语言编写的若干个封装性很好的类，主要的工作是解析 FITS 文件，建立合适的数据结构描述解析得到的信息，以及将数据信息插入到数据库表中。

数据发布模块负责数据信息的查询和发布，是跟用户打交道的部分。这部分的实现采用客户端/服务器模式，由客户端向服务器发出请求，然后等待服务器返回结果。服务器端的实现对于客户是透明的。完成的工作主要有：提供给用户一个可以方便查询的界面；负责处理查询请求，到数据库表中进行查询，取得查询结果；根据用户选择的返回格式对查询结果进行格式转换，然后显示出来（不支持显示的应该可以保存为相应格式的文件）；提供需要的 FITS 文件的下载。

2.3 系统功能说明

该系统的前两个模块主要是面向数据管理者的，最后一个模块面向用户。下面从不同的使用者的角度阐述该系统所能提供的功能。

数据管理者可以做到的事：⑴根据我们在数据平台搭建模块提供的设计方案，完成数据平台的搭建；⑵在指定的配置文件中输入数据库的用户名和密码、数据文件的主目录的绝对路径等信息，运行数据导入模块提供的程序，将所有数据文件信息存入已搭建好的数据平台。程序会递归分析主目录下所有文件，如果找到了符合条件的 FITS 文件，并且该文件的信息还未入库，程序就能解析该文件的文件名和文件头部的标题记录，将相应信息以合适的数据类型的存入到数据库对应表中，否则就接着处理别的文件，直到把所有文件都扫描一遍为止。

用户可以做到的事：⑴在查询界面中输入若干个查询条件，选择返回格式和返回内容，提交页面，实现对所有条件的联合查询；⑵如果返回的是 HTML 格式，自动分页显示，所有信息以表格形式罗列出来，提供对查询到的文件的链接，同时提供三种方式的打包下载：下载选中的文件、下载本页所有文件和下载所有查询到的文件；⑶如果返回的是别的格式，进行格式转换后显示出来（不支持显示的应该可以保存为相应格式的文件）。

2.4 开发环境及工具

首先数据平台的搭建需要有数据库管理系统的支持。在整个开发过程中，我们对数据库表只涉及一些诸如查询、插入等的简单操作，但是由于数据量比较大，对性能和速度的要求会比较高。所以“灵巧”是我们选择数据库管理系统的主要标准。MySQL是一个多用户、多线程的强壮的SQL数据库管理系统。因为没有线程创建开

销、一个较小的语法分析器、较少功能和简单的安全性，所以它的执行速度是比较快的[2]，很好的满足了我们的需求。我们采用的是MySQL的 4.0 版本。

其次，我们选用JAVA作为主要编程语言。JAVA是目前比较流行的编程语言，我们选择它则不仅仅是因为它流行，而是它在下述几个方面都有很好的支持，非常符合我们的需求。第一，JAVA是面向对象的编程语言，有良好的封装性和可扩展性，还有丰富的API库可以使用[3]；同时它又与平台无关，在不同软硬件平台上移植非常方便。第二，JAVA程序很注重安全性，这对于网络应用程序的开发是极其重要的[4]，我们的数据发布平台就属于网络应用；JAVA支持多线程，不必开辟多个进程，可以减轻系统负担。JDK(Java Development Kit)提供了JAVA应用程序的运行环境和基本API，我们使用的是JDK1.4 版本。

最后要介绍的是一个Java Web应用的容器——Jakarta Tomcat，它是一种Servlet/JSP容器[5]。Servlet是一种运行在支持JAVA语言的服务器上的组件，它在服务器端运行，不依赖于浏览器[6]。在数据发布模块中会依赖Servlet处理查询请求和实现程序逻辑，所以Tomcat容器是必不可少的工具。我们使用的是Tomcat 5.0 版本。

第三章　数据平台搭建模块的设计及实现

3.1 巡天数据的文件格式

数据平台搭建模块的主要设计目标就是要搭建适合 BATC 数据文件的数据平台，因此对 BATC 数据文件格式的了解是我们做好设计工作的重要前提。我们已经知道，BATC 巡天观测产生的数据是以 FITS 格式的文件进行保存的。FITS (Flexible Image Transport System)是国际天文学会 1982 年确定的世界各天文台之间用于数据传输、交换的统一标准格式。它提供了图像的单值转换，对一维、二维、三维甚至多维的数据类型都提供了合适的转换，精度包括符号在内可以达到 32 位。多年的使用表明，以 FITS 格式记录天文数据对以后的数据再处理、科学研究以及不同图像处理系统间的数据交换都是非常有利的。

FITS 标准规定，一个标准的 FITS 文件应包含一系列逻辑单元，每个逻辑单元由两部分构成：标题记录和数据组。每个逻辑单元的长度是固定的，总是 2880 字节。每个单元的开头是一组标题记录，作用是描述随后的数据组的结构和坐标系统以及传送任一附加的参数和伴随的正文。这样的记录的定义在建立一个交换格式中是非常重要的。它使用了 ASCⅡ编码的卡片映像，基本语法格式是：关键词= 参数值/注释说明。关键词是一个左边对齐的 8 字符 ASCⅡ码字符集，放在 1-8 列里，字母必须大些。“=”号出现在第 9 列，而第 10 列是空格。参数值按 FORTRAN-77 表控规定书写，直接可读格式。逻辑值（T 或 F）和数值在数值区中紧靠右排列，字符串用单引号括出，字母大些。符号“/”表示数值区结束。注释说明紧接符号“/”，用大小写 ASCⅡ码字符书写。用关键词 END 表示标题记录结束。数据组紧接在标题记录后面出现，以象元记录图像信息。从存储角度来讲，也就是用二进制流来记录图像详细信息。

我们所要处理的 FITS 文件只包含一个逻辑单元，即前面是标题记录，紧接着是数据组，中间以关键词 ENDS 作为分隔。图 3-1 就是一个 FITS 文件的标题记录部分，该文件是从我们要处理的大量数据文件中选取的一个典型例子，其他的文件在格式上都是类似的。但是由于需要描述的信息不同，不同的文件在关键字的选择上有着少许不同，现有的数据文件中就存在着三种不同的类型，以后还可能会出现更多种，因此在设计数据平台时一定要考虑到这一点，否则将无法记录所有文件信息。

```
SIMPLE  =                    T / NORMAL FITS IMAGE
BITPIX  =                  -32 / DATA PRECISION
NAXIS   =                    2 / NUMBER OF IMAGE DIMENSIONS
NAXIS1  =                 2048 / NUMBER OF COLUMNS
NAXIS2  =                 2048 / NUMBER OF ROWS
BLOCKED =                    T / CHECK FOR POSSIBLE BLOCKING
CRVAL1  =                    0 / COLUMN ORIGIN
CRVAL2  =                    0 / ROW ORIGIN
CDELT1  =             1.000000 / COLUMN BINNING
CDELT2  =             1.000000 / ROW BINNING
IMNAME  = '              TA05' / OBSERVATION SEQUENCE NAME
DATE-OBS= '          26/09/95' / DATE OF START OF OBSERVATION (dd/mm/yy)
TIME    = '        15:10:56.0' / UT TIME OF START OF OBS.
EXPOSURE=                  300 / EXPOSURE TIME (SEC)
RA      = '       22:25:52.68' / RIGHT ASCENSION (1995.7)
DEC     = '        17:19:51.9' / DECLINATION     (1995.7)
HA      = '       00:57:40.73' / HOUR ANGLE OF MIDDLE OF OBS.
EPOCH   = '            1995.7' / EPOCH OF RA AND DEC
VOLT1   =             -92.1771 / TEMPERATURE
VOLT2   =               62.876 / SKY ADU/PIXEL
VOLT3   =                 3.59 / SEEING  (arcsec)
VOLT4   =               4.9525 / 5 VOLT SUPPLY
VOLT5   =               0.0000 / DRAIN VOLTAGE
VOLT6   =               0.0000 / SUBSTRATE VOLTAGE
VOLT7   =               0.0357 / UNUSED
OBJECT  = 'AB_32443_A05 6075A 300s                                              '
INSTRUME= 'China CCD #01                                                        '
CCDID   = 'FORD_1             ' / DETECTOR ID
STATUS  = 'SUPER STAR         ' / IMAGE PROCESSING STATE
SHUTTER = 'OPEN               ' / SHUTTER STATE
RECORDS =                    0 / IMAGE SIZE IN FITS RECORDS
OBSERVER= 'X. Zhou & Y.J. Chen                     (OVERSCAN CORRECTED)     '
PZERO   =             0.000000 / ZERO POINT BIAS
PSCALE  =             1.000000 / IMAGE SCALE FACTOR
LICK    = 'FITS2              ' / SPECIAL LICK FLAG
CRPIX1  =                    1 / PIXEL CORRESPONDING TO CRVAL1
CRPIX2  =                    1 / PIXEL CORRESPONDING TO CRVAL2
COMMENT = '* BAO Schmidt CCD Data-Taking system                                 '
COMMENT = '                                                                     '
COMMENT = '                                                                     '
COMMENT = '                                                                     '
COMMENT = '                                                                     '
TELESCOP= 'BAO 60-90 SCHIMDT ' / TELESCOPE USED
PROGRAM = 'CHINA              ' / DATA ACQUISITION PROGRAM
TTIME   =                  300 / ELAPSED TIME (EXPOS. START TO READ START)
A81     =  -0.828430670279E-05 / 6 plate coef.
A82     =  -0.230594013854E-06 / matched star:   164
A83     =   0.229760131038E-06 / RMS (arcsec):  0.38
A84     =  -0.828679359768E-05 /                          l_sun = 345:15:53
A85     =   0.825201544582E-02 /                          b_sun =  25:08:45
A86     =   0.872605573237E-02 /
A87     =             5.8734102 / p_c (2000.0)
A88     =             0.3028678 / in rad.
RA2000  = '       22:26:05.19' / plate center
DEC2000 = '        17:21:11.0' /
GALLONG =                80.38 / galactic coordinates (1950.0)
GALLATI =               -33.25 / both units in degrees
EXTINCT =                0.213 / BH Bmag, EXTINCTION = 4*E(B-V)
AIRMASS =                1.114 /
AZIMUTH =                32.56 / Azimuth is measured from south through west
ALTITUDE=                63.79 / both units in degrees
MPHASE  =                  2.0 / Define the length of Moon_Month 29.53d
MAZIMUTH=               124.38 / Moon Azimuth is measured from south thr. west
MALITIUD=               -47.29 / both units in degrees
MANGLE  =               131.97 / Position angle of Moon to CCD_Field center
MDIRECT =                 2.79 / Moon direction in the X_Y plane of CCD frame




END                            / END OF HEADER
```

图 3-1　　FITS 文件的标题记录部分

3.2 数据平台搭建模块的设计

在整个系统开发过程中，数据平台的设计是一个非常重要的环节。它既是整个系统开发的起点，同时又是设计和实现数据导入模块和数据发布模块的基础。因此我们将足够的注意力放在数据平台的设计方案上，为整个系统的构建打下坚固的地基。

我们很明确数据平台的设计目标就是要完整地记录信息以及方便的管理和发布信息，因此我们借助关系型数据库管理系统来构建数据平台。我们使用它的原因有两个：首先，关系型数据库技术发展到现在，已经是比较成熟的技术了，拥有自己的独立规范和接口，有利于复用。而且几乎各大厂商都推出了自己的数据库管理系统，各有特色，我们可以有丰富的选择。其次，关系型数据库是以表的形式存放数据，利用表的各种属性信息可以很快掌握所有数据的情况，无论是查询还是添加新数据都是非常方便的。此外，我们还可以在表上建立索引加快查询速度，还能给使用者设立不同权限，比起直接管理大量数据文件来，要方便和快捷的多。

这样一来，我们就将数据平台的设计转为对数据库表的相关设计。关于表的设计，应该注意几个问题。第一，创建表时注意选择合适的数据类型，尽量保证信息的原始性，同时兼顾处理的简单性。第二，标题记录是以字符形式来记录的，读起来直观易懂，数据量很小。数据组是二进制流，数据量很大，所以从性能角度考虑，可以只存储从标题记录和文件名中解析得到的信息，然后给出文件路径即可。第三，表的设计要体现各种子类型的并集，才能保证不丢失信息。

3.3 数据库相关设计

3.3.1 表的设计

首先根据需求建立实体-关系模型。该系统中只涉及到一个实体集，那就是 FITS 文件。我们要确定该用哪些属性来描述该实体集。要描述一个文件，首先可以确定的一个属性就是文件名，它是唯一的。文件名是 16 位字符串，从左向右除了第 1 位都是字母 P 外,其余各位都有着特定含义。第 2 位到第 4 位表示流水号，第 5 位到第 8 位表示儒略日，第 9 位到第 12 位表示天区名，第 13 位表示滤光片，第 14 位到第 16 位表示当天序列号。按照各部分的含义，从文件名中可以解析出五个属性。其次，文件的相对路径信息也要作为一个很重要的属性。最后，在 FITS 文件的标题记录部分，每一个关键词都可以作为该实体集的一个属性。综合上述分析，可以得到该实体集的所有属性。

下一步要将实体-关系模型转化成表。一个实体集对应数据库中的一张表，该实体集的各个属性对应于表中的各列。此外，添加一列作为文件的 ID 号（整型），也是这张表的主键，ID 号每插入一条记录自动增 1。文件名虽然不是主键，却是唯一键，且不允许为空，否则就没有意义了。当数据文件的某数据字段可能出现缺值时，将本字段设置允许为空。

为了增加可读性，表中数据字段的命名也要遵循一定的规则。一般来说，文件标题中的关键词直接作为表中对应列的列名。如果遇到关键词中有数据库不支持的符号时，转换成下划线即可。如果由上述规则得到的列名与数据库的保留字冲突，则要进行相应调整[7]。例如，表示赤纬的关键词DEC就是MySQL数据库中的保留字，在字段名中不允许出现，所以改用DECC表示。

为了保证文件标题数据的原始性，将所有关键词对应的列都使用字符串类型存储。

表 3-1 是对表的设计结果的一个详细描述。

表 3-1　　表的设计结果

字段名	存储类型	索引	主键	说明
CCDID	varchar(50)			
MPP	varchar(50)			
RAWMESS	varchar(150)			
RECORDS	varchar(50)			
TEMPCON	varchar(50)			
VOLT4	varchar(50)			
VOLT5	varchar(50)			
READ_SPD	varchar(50)			对应 fit 中 READ-SPD
DATE_OBS	varchar(50)			对应 fit 文件中 DATE-OBS
DECC2000	varchar(50)	有		对应 fit 文件中 DEC2000
DECC	varchar(50)	有		对应 fit 文件中 DEC
A81	varchar(50)			
A82	varchar(50)			
A83	varchar(50)			
A84	varchar(50)			
A85	varchar(50)			
A86	varchar(50)			
A87	varchar(50)			
A88	varchar(50)			
AIRMASS	varchar(50)			
ALTITUDE	varchar(50)			

AZIMUTH	varchar(50)			
b_sun	varchar(50)			
BITPIX	varchar(50)			
BLOCKED	enum('T','F')			
CDELT1	varchar(50)			
CDELT2	varchar(50)			
COMMENT	varchar(250)			
CRPIX1	varchar(50)			
CRPIX2	varchar(50)			
CRVAL1	varchar(50)			
CRVAL2	varchar(50)			
EPOCH	varchar(50)			
EXPOSURE	varchar(50)			
EXTINCT	varchar(50)			
filter	varchar(10)			滤光片
GALLATI	varchar(50)			
GALLONG	varchar(50)			
HA	varchar(50)			
id	bigint	有	√	文件 ID 号，表的主键，自增
IMNAME	varchar(50)			
INSTRUME	varchar(50)			
jd	int(10)			儒略日
l_sun	varchar(50)			
LICK	varchar(50)			
MALITIUD	varchar(50)			
MANGLE	varchar(50)			
MAZIMUTH	varchar(50)			
MDIRECT	varchar(50)			
MPHASE	varchar(50)			
name	varchar(50)			文件名，非空，unique key
NAXIS	varchar(50)			
NAXIS1	varchar(50)			
NAXIS2	varchar(50)			
numberOrder	int(10)			流水号
OBJECT	varchar(50)	有		观察物体的名字
OBSERVER	varchar(150)			
path	varchar(150)			文件相对路径
PROGRAM	varchar(50)			
PSCALE	varchar(50)			
PZERO	varchar(50)			

RA	varchar(50)	有		
RA2000	varchar(50)	有		
sequence	int(10)			当天序号
SHUTTER	varchar(50)			
SIMPLE	enum('T','F')			
skyarea	varchar(10)			天区名
STATUS	varchar(50)			
TELESCOP	varchar(50)			
TIME	varchar(50)			
TTIME	varchar(50)			
VOLT1	varchar(50)			
VOLT2	varchar(50)			
VOLT3	varchar(50)			
VOLT6	varchar(50)			
VOLT7	varchar(50)			

3.3.2 其他设计

要为表内表示天体名称和 ID 号的列添加单列索引，同时要为星表中表示位置信息的列添加单列索引，比如表示赤道天体坐标位置的赤经、赤纬列等，如果表内有多个历元的赤经、赤纬，也必须为他们都添加单列索引。

最后为数据库设置权限，将普通用户对表的权限设定为 Select，不允许对表进行 delete，insert 和 update 操作，管理员拥有所有权限。

3.4 遇到和解决的主要问题

在设计数据库的表结构时遇到了一个棘手的问题，那就是无从得知我们要处理的 FITS 文件有几种可能的格式。在第三章讲搭建数据平台需要注意的问题时曾经提到，那几种不同的格式在标题关键词的选取上有着少许差异，如果不了解这些差异，我们就无法确定数据库表的数据字段。现有的 FITS 文件有 2 万多个，我们不可能去查看所有文件。权宜之计就是先抽查若干有代表性的文件，根据抽查的结果确定数据字段，先创建一个测试用的数据库。后面要介绍的数据导入模块是运行在数据平台之上的，我们可以在它的程序中添加一些测试代码，如果遇到新的格式的文件就记录它的路径。根据记录的信息我们就能找到尚未发现的格式，然后完善数据库。表 3-1 就是这样不断完善的结果，它包含了 FITS 文件中出现的所有关键词。

第四章　数据导入模块的设计及实现

4.1 数据导入模块的设计目标

数据文件信息如何录入到搭建好的数据平台中，这一过程是靠数据导入模块来实现的。该模块是建立在已经搭建完毕的数据平台的基础上，解决如何将现有大量数据自动存入数据平台的问题。在确立数据导入模块的设计目标时，我们首先考虑该模块能够对外提供的基本功能，然后讨论怎样在实现基本功能的基础上增强该模块的可移植性和复用性。

数据导入模块处理的对象是 archive 主目录下满足某一命名规则的所有 FITS 文件，得到的数据信息最终应该要被存入数据库对应表中。这张表的各列中，有的是在描述文件标题部分的信息，有的记录了文件名信息，还有一项是要记录文件路径信息的，这些信息都是从数据文件中解析得到的。因此数据导入模块首先要实现的功能就是将数据文件的标题部分和文件名解析成许多属性值，这些属性值应该是和数据库表的列一一对应的。此外，为了方便对数据的操纵，我们还应该构造专门的数据结构存储解析得到的属性值，对属性值的读取要通过函数实现。最后，要有专门的函数负责与数据库的操作。

要提高该模块的可移植性和复用性，可以考虑在以下两个方面进行加强。首先，FITS 文件子类型间的差异要求我们在设计数据结构时不能只依赖于现有文件本身的结构，要足够灵活才能有更好的扩展性。其次，该模块涉及了与本地数据库和本地文件系统的操作，比如在连接数据库时要指出数据库名、表名、用户名和密码，在操作文件时要给出文件路径，这些信息如果写在程序中势必会影响程序的可移植性。可以考虑单独创建一个配置文件，在配置文件中记录所有跟本地系统相关的内容，程序只要去读取配置文件即可获得想要的信息。

以上述目标作为设计的出发点，我们最终需要设计若干个封装性很好的类去实现不同部分的功能，同时在主程序中创建各个类的对象去实现程序逻辑。

4.2 数据导入模块的设计和实现

4.2.1 类的设计

在数据导入模块中共设计了七个独立的类，下面是对这些类的功能描述：

类 DbControl：进行所有有关数据库的操作。包括与数据库建立连接；判断给出的信息是否已经存在于数据库中；对数据库进行数据插入；关闭对数据库的连接。

类 HeadAnalysis：对 FITS 文件标题部分进行分析，读出所有关键词及其对应的值。将这些关键词与值构造成哈希表。该类不依赖于具体 FITS 文件的格式，如关键词的个数、种类，只要文件头部存在一个结束标记 END，程序就可正确运行，该类可以复用。

类 FileHead：构造该类的目的是为了保存文件的标题信息。它可以应用于现有的所有种类的 FITS 文件。它将哈希表中的关键字和值封装到该类中。它的工作是建立在已经存在的哈希表的基础之上，类中方法的输入是哈希表。

类 NameAnalysis：对 FITS 文件名进行分析，返回一个 FITS 文件的基本属性，包括文件名、文件所在路径、天区名、滤光片、流水号、儒略日和当天序号。类中方法的输入是一个 FITS 文件名。

类 FitsFile：构造该类的目的是为了组织 FITS 文件名信息。它把由 NameAnalysis 分析的属性都封装在该类中，通过这个类，操作 FITS 文件名中的属性。主要包含对文件名各属性的 set 和 get 方法。

类 Config：构造该类的目的是为了方便获取所有涉及平台相关性的配置信息。这部分信息存在一个名为 MyVO.properties 的属性文件里，在不同平台间移植时只要修改这个属性文件即可。

类 SaveFile：包含 main 方法，程序的入口点。主要实现对特定路径中下的所有 FITS 文件（包含子目录中的）进行遍历，提取出 FITS 文件的文件名信息、文件标题信息，将这些信息写入数据库。主要是一个递归的遍历过程。

4.2.2 类的描述

下面给出了各个类的成员变量和成员函数的描述。

```
public class DbControl{
    private static Connection theConnection = null; //成员变量，代表与数据库的连接
    public static void connectDb(Config conf){
    } //成员函数，负责连接数据库
    public static boolean isexist(FitsFile f){
    }//成员函数，判断数据库中是否已经存在该文件的信息
    public static void insertDb(FitsFile f,FileHead fh){
    } //成员函数，将解析得到的文件名和标题部分的信息存入数据库中
    public static void closeConn(){
    } //成员函数，关闭数据库连接，释放资源
}
public class HeadAnalysis {
    public static Hashtable getHead(File f){
} //成员函数，分析文件标题，将得到的信息存入哈希表中
}
public class FileHead implements Serializable{
    private Hashtable hhtable; //哈希表类型的成员变量
    public FileHead(Hashtable htable){
    } //构造函数
    public String getSIMPLE(){
    } //成员函数，得到哈希表中 SIMPLE 关键字的值
    public String getBITPIX(){
    } //成员函数，得到哈希表中 BITPIX 关键字的值
    ……
```

```
}
public class NameAnalysis{
      public static FitsFile getBaseFits(File f){
} //成员函数，解析文件名，将结果保存在 FitsFile 对象中并返回
}
public class FitsFile implements Serializable{
      private String name;              //fits 文件名
      private int       numberOrder; //流水号
      private String path;      //fits 文件所在目录
      private String skyArea;       //对应的天区
      private String filter;          //滤光片
      private int jd;               //拍摄日期，儒略日
      private int sequence; //当天序号
     public FitsFile(){
      } //不带参数的构造函数
      public FitsFile(String name){
      }//带参数的构造函数

      public void setName(String name){
      } //给成员变量 name 赋值
      public String getName(){
      } //返回成员变量 name 的值
     ……
}
public class Config {
     Properties prop; // Properties 类型的成员变量
     public Config() {
```

```
    } //不带参数的构造函数
    public Config(String propFileName){
    } //带参数的构造函数
    public String getAttribute(String name){
    } //成员函数，返回配置文件中参数 name 的值
    public void setAttribute(String name, String value){
    } //成员函数，将配置文件中参数 name 的值置为 value
}
public class SaveFile{
    public static void main(String[] args) {
    } //main 函数
    private static Config conf = null; //Config 类型的成员变量
    private static String path = null; //String 类型的成员变量，表示主目录路径
    private static void init(){
    } //初始化函数
    private static void intoDir(File f){
    } //成员函数，递归处理主目录路径下的所有文件
}
```

4.2.3 主程序流程

图 4-1 描述了主程序的执行过程以及对各个类的使用。

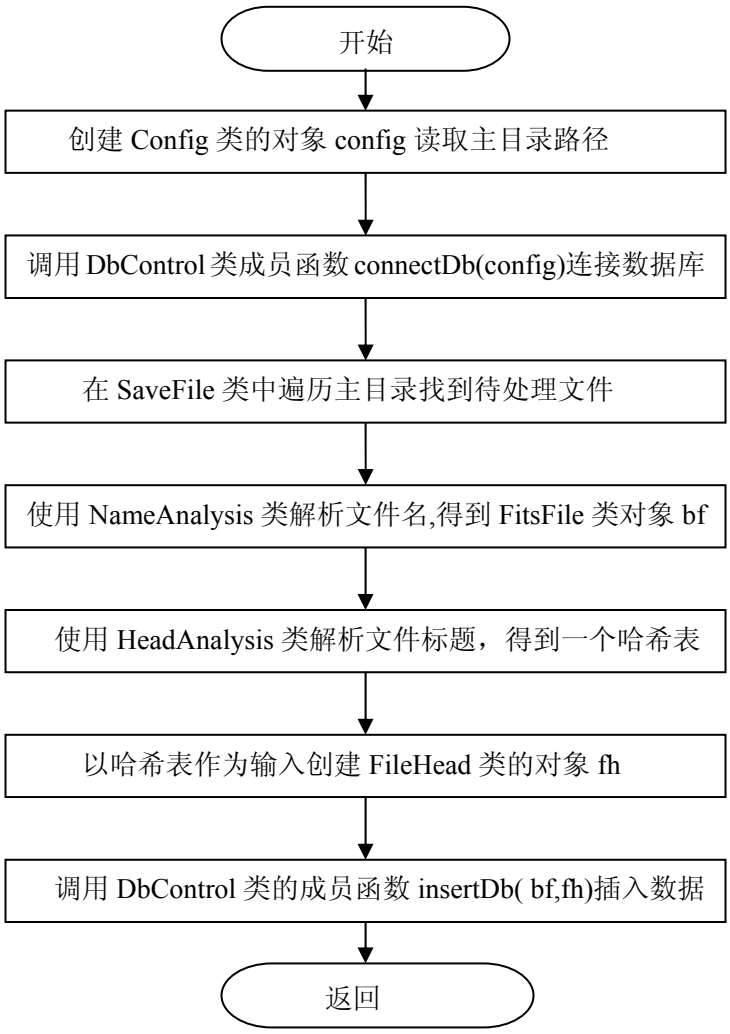
开始
创建 Config 类的对象 config 读取主目录路径
调用 DbControl 类成员函数 connectDb(config)连接数据库
在 SaveFile 类中遍历主目录找到待处理文件
使用 NameAnalysis 类解析文件名,得到 FitsFile 类对象 bf
使用 HeadAnalysis 类解析文件标题，得到一个哈希表
以哈希表作为输入创建 FileHead 类的对象 fh
调用 DbControl 类的成员函数 insertDb(bf,fh)插入数据
返回

图 4-1　　主程序流程图

第五章　数据发布模块的设计及实现

5.1 数据发布模块的设计目标

在数据平台搭建模块和数据导入模块都实现之后，我们就要开始着手数据发布模块的设计和实现了。数据发布模块的实现要建立在已存有数据的数据平台的基础之上，通过 Web 技术提供数据发布服务，因此需要有 Tomcat 服务器的支持。该模块涉及的工作包括两部分，一部分是设计程序来实现发布界面和发布功能，另外一部分是要在 Tomcat 服务器中配置资源和访问信息。下面就这两部分谈一下我们的设计目标。

首先，数据库中存储着大量的数据信息，我们要提供给用户一个查询界面帮助他们找到自己需要的信息。界面的设计应当是友好的、灵活的，对浏览器的兼容性要强。用户通过查询界面与服务器进行交互，由服务器端的后台程序处理用户提交的表单，再将结果返回给用户。查询界面和后台程序都是作为 Web 应用中的一部分部署在 Tomcat 应用目录下的。Web 应用由一组静态文档（如 HTML 页、图片等）、Servlet、JSP、实用类、客户端类和描述 Web 应用的 web.xml 文件组成，并且它们在 Web 应用中都有各自固定的存放目录。可以说，我们主要的目标就是要设计、实现和部署一个 Java Web 应用，这个应用要能实现我们最初设定的提供查询页面、处理查询请求、对查询结果进行数据转换、提供文件下载等功能。

其次，Web 应用的正常发布依赖 Tomcat 容器的某些配置。这些配置有些是必需的，有些是可选的，而且配置的方法也不止一种。我们应该选择一种最优的方式，按部就班的完成配置工作，为 Web 应用的正常发布和稳定运行提供良好的保障。Web 应用的 web.xml 也是一个配置文件，在配置数据源或发布某些组件（如 Servlet）时，必须在 web.xml 文件中添加相应的配置信息。

根据我们设定的目标，先讨论一下 Tomcat 环境配置的问题，然后在配置好的环境中展开 Web 应用的设计、开发和部署工作。

5.2Tomcat 环境配置

5.2.1 配置虚拟主机

如果我们以<CATALINA_HOME>代表 Tomcat 的安装目录，那虚拟主机的配置就要在<CATALINA_HOME>/conf/server.xml 中进行。server.xml 是 Tomcat 中主要的配置文件，Tomcat 的组件都可以在该文件中进行配置。我们在这里配置虚拟主机，

目的是将不同域名与服务器中特定文件夹关联起来。用户使用某一个域名访问服务器，只能访问到对应文件夹中的内容。配置的方法很简单，在 server.xml 中找到一个名为 localhost 的<Host>元素，将名字改为想要的域名即可。

5.2.2 创建 Context

从Tomcat5.0起，推出了“上下文描述符”的概念。[8]一个上下文描述符是一段XML数据的片断，这个片段包含了一个有效的Context元素，它与主服务器配置文件*server.xml*里的<Context>元素作用是一样的，可以配置跟对应Web应用有关的所有方面。对每一个已经部署的web应用，Tomcat会自动创建一个匹配的Context XML描述符，直到该应用被撤销。自动创建的Context片段中只包含该Web应用的访问路径信息。对于对一个给定的主机，它的上下文描述符位于<CATALINA_HOME>/conf/Catalina/[hostname]中，文件名默认与Web应用名相同。

5.2.3 配置数据源

数据源的配置主要是为了使用 Tomcat 中的连接池。由于建立数据库连接是一种比较耗时的操作，所以对于支持 JDBC2.0 的数据库，Tomcat 服务器提供了连接池的支持。有了连接池的实现，在客户端调用 close（）方法的时候实际上并不关闭连接，而是把连接返回到一个可重用连接的连接池中给其它客户端使用。这样就避免了任何重复打开和关闭数据库连接造成的开销，并且允许大量的客户端分享相对较少的数据库连接，从而提高数据库操作的性能。数据源配置完毕之后，我们就能在应用程序中引用 DataSource（数据源）对象，从连接池中直接获取数据库连接。

DataSource对象是由Tomcat提供的，因此不能在程序中采用创建一个实例的方式来生成DataSource对象，而需要采用Java的另一个技术JNDI，来获得DataSource对象的引用。[8]可以简单地把JNDI理解为一种将对象和名字绑定的技术，对象工厂负责生产出对象，这些对象都和惟一的名字绑定。外部程序可以通过名字来获得某个对象的引用。数据源的配置涉及修改Context片段和web.xml文件，在Context片段中加入定义数据源的元素< Resource >，在web.xml文件中加入<resource-ref>元素，声明该Web应用所引用的数据源。

在该 Web 应用的 Context 片段中，我们要添加如下一段描述：

```
<Resource name="jdbc/TestDB" type="javax.sql.DataSource"/>
  <ResourceParams name="jdbc/TestDB">
    <parameter>
```

```
        <name>url</name>
        <value>jdbc:mysql://localhost:3306/batc?autoReconnect=true</value>
      </parameter>
      <parameter>
        <name>username</name>
        <value>root</value>
      </parameter>
       <parameter>
        <name>password</name>
        <value></value>
      </parameter>
      <parameter>
        <name>maxActive</name>
        <value>10000</value>
      </parameter>
      <parameter>
        <name>maxWait</name>
        <value>5000</value>
      </parameter>
      <parameter>
        <name>driverClassName</name>
        <value>org.gjt.mm.mysql.Driver</value>
      </parameter>
      <parameter>
        <name>maxIdle</name>
        <value>20</value>
      </parameter>
    </ResourceParams>
</Context>
```

< Resource >元素用来定义 JNDI Resource 。Tomcat 把 DataSource 作为一种可以配置的 JNDI 资源来处理。其中，"jdbc/TestDB"是我们给该 JNDI 资源起的名字，"javax.sql.DataSource"表明它是数据源类型。它有 7 个参数需要配置，分别是：url(JDBC 的连接 url)，username(连接数据库的用户名)，password（username 连接数据库的密码），MaxActive（连接池中最大可用连接数），MaxIdle（连接池中最大闲置连接数），MaxWait（连接可用的最大等待时间，单位 ms，超过该时间就返回异常），driverClassName（JDBC 驱动类名）。

同样的，在该 Web 应用的 web.xml 中，我们要添加如下描述：

```
<web-app ....>
    .....
   <resource-ref>
     <res-ref-name>jdbc/TestDB</res-ref-name>
     <res-type>javax.sql.DataSource</res-type>
     <res-auth>Container</res-auth>
   </resource-ref>
</web-app>
```

5.3 Web 应用的设计和实现

如图 5-1 所示， Web 应用中包括 5 个主要的模块：一个模块负责查询界面的生成，一个模块控制查询请求的处理和转发，另外三个模块是针对不同的返回格式所做的不同处理。

所有模块中除了查询界面是由HTML实现外，其余的都使用Servlet实现。Servlet是一种独立于平台和协议的服务器端的Java应用程序，可以生成动态的Web页面。我们通常使用的Servlet类都是扩展HttpServlet抽象类，再覆盖它的部分方法所实现的。[9]常覆盖的方法有doGet( )和doPost( )，这两个方法分别是对HTTP请求中的Get请求和Post请求进行处理的。尽管如此，这两个方法的参数却是完全相同的，一个参数是HttpServletRequest类型的对象，另一个参数是HttpServletResponse类型的对象。HttpServletRequest类型的对象封装了HTTP请求的各种信息，我们可以通过它来检索HTML表单所提交的数据或URL上的查询字符串，这些信息以参数名/参数值的形式存放，可以通过调用该对象的方法获取参数信息。通过HttpServletResponse对象可以生成HTTP响应结果，也有众多方法可供调用。因此，我们的工作主要集中在设计并实现在获得请求和生成响应之间的程序逻辑。

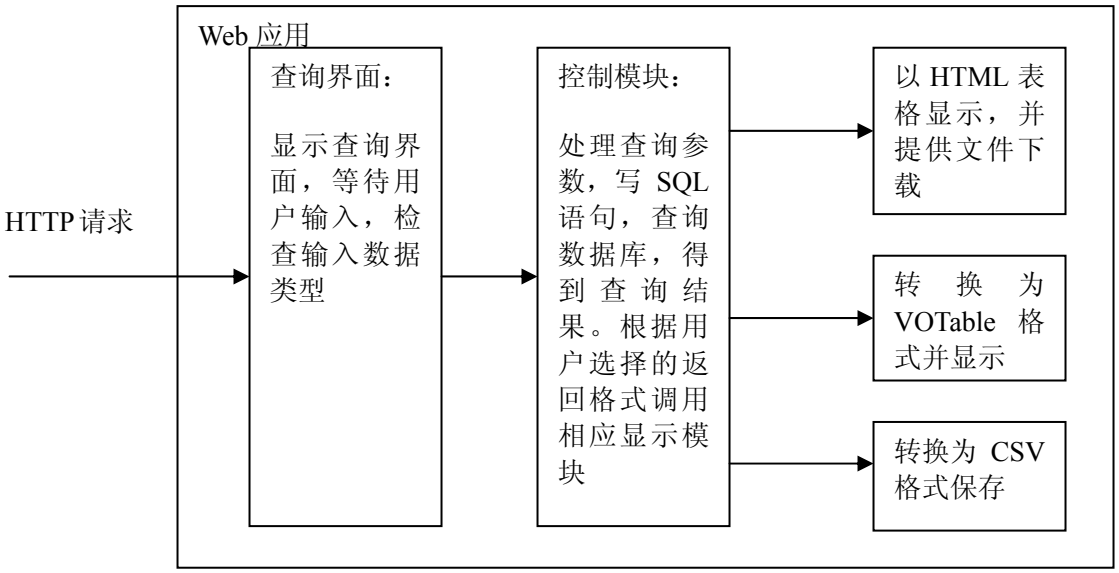

图 5-1　　Web 应用的功能模块示意图

5.3.1 查询界面的设计和实现

查询界面是一个含有表单的HTML页面index.htm，引用了CSS样式表文件skyserver.css控制界面样式[10]。表单中实现的查询条件有：文件名查询，观测日期查询，观测物体查询，赤经赤纬查询，天区查询，滤光片查询。该页面实现的是所有条件的联合查询，不需要的条件可以不填。此外，用户还能灵活选择查询结果的返回格式和返回内容。可选的返回格式有三种，分别是HTML、VOTable和CSV，用户可以在这三种格式中任选一种。可选的返回内容有 73 项，每一项对应数据库中的一列。在提交表单前，调用JavaScript脚本文件check.js对用户的输入进行数据类型检查，如果输入类型有误，将不会提交表单[11]。查询界面如图 5-2 和图 5-3 所示。

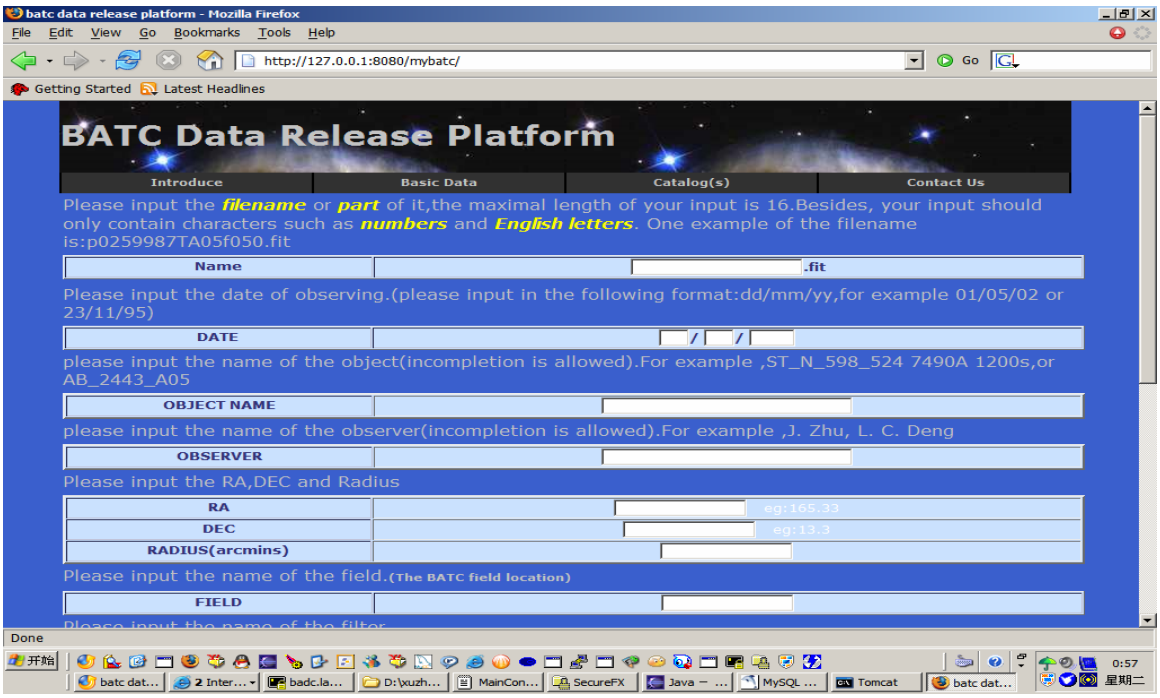

图 5-2　　查询界面上半部分截图

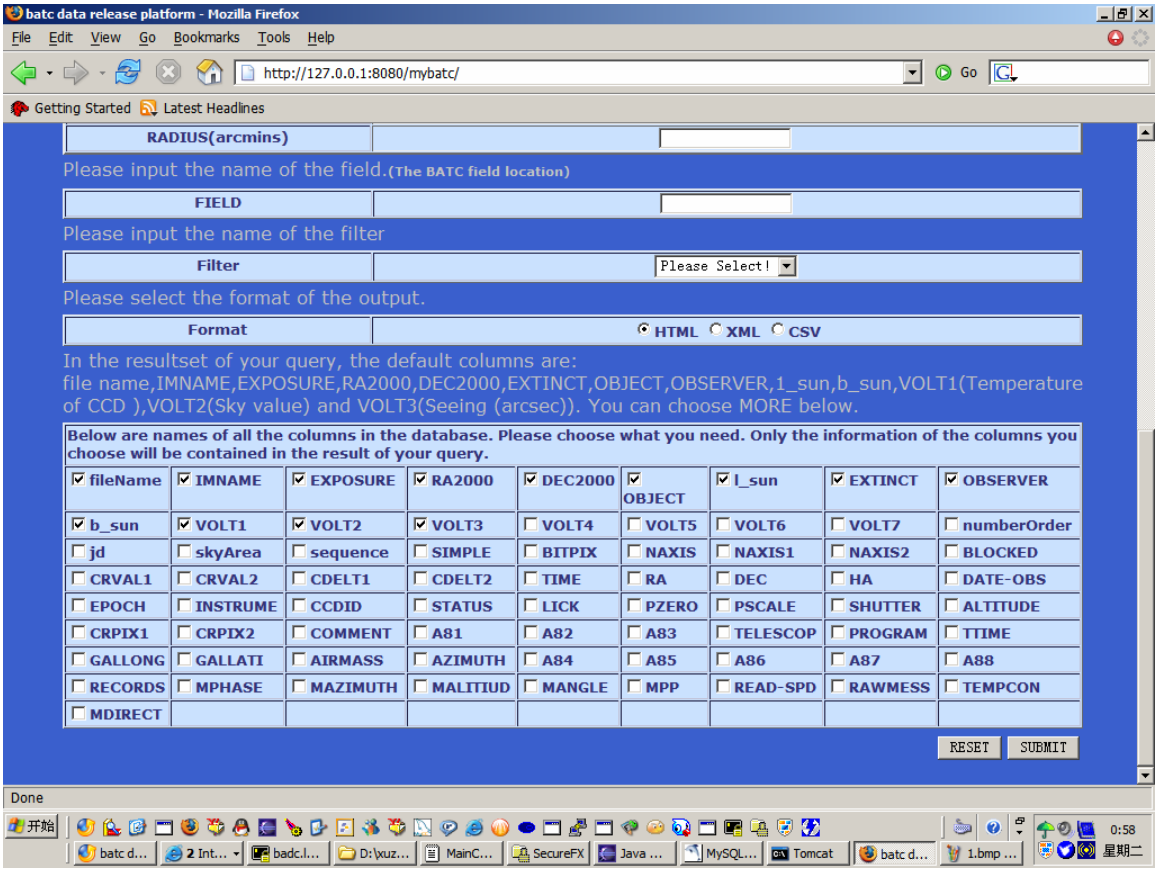

图 5-3　　查询界面下半部分截图

5.3.2 控制模块的设计和实现

控制模块的工作流程如下[12]：处理用户提交的表单数据，取得所有的参数值，根据这些参数值写出SQL查询语句。连接数据库，进行查询，将得到的结果集和结果集元数据保存在Session中。最后根据用户选择的返回格式转发给相应的Servlet处理。

控制模块靠Controller这个起着指挥作用的Servlet来实现的。该Servlet重写了doPost( )方法，并构造了一个函数doSelect( )专门负责生成SQL查询语句。连接数据库时使用了查找数据源的方法[13]，具体代码如下：

```
DataSource ds = null; //定义数据源
Context ctx = new InitialContext();
ds = (DataSource) ctx.lookup("java:comp/env/jdbc/TestDB"); //查找名为
                                                    jdbc/TestDB 数据源
Connection conn = ds.getConnection(); //  从连接池中取得数据库连接
```

5.3.3 显示模块的设计和实现

显示模块主要负责以 HTML 表格、VOTable 和 CSV 三种不同方式显示查询结果，因为浏览器无法显示 CSV 格式的结果，所以会要求用户将查询结果保存到一个 CSV 文件里。

该模块是由 HtmlDisplay、Download 、TransVOTable 、VOTableDisplay 、CsvDisplay 五个 Servlet 共同实现的。下面会分别介绍它们各自的功能，至于主要的程序代码会在附录中给出。

HtmlDisplay：如果用户选择将查询结果以 HTML 表格的格式显示出来，Controller 会把 request 请求转发给该 Servlet。它主要的工作是：捕获 Session，从 Session 中得到结果集和结果集元数据。采用分页的方式显示结果，显示哪一页由 request 请求中的参数 curPge 决定。每页显示 20 个条目，每个条目中的文件名是一个可以直接点击下载的链接，每个条目前有一个复选框，可以选择多个文件下载。此外还有两个链接，一个可以打包下载本页所有文件，另一个可以打包下载本次查询的所有文件。上面出现的三个链接都是将页面导向 Download 进行处理的，只不过每个链接传递的参数不同。显示新的一页时将 curPge 置为相应的页码回调自己即可。

Download：接收从 HtmlDisplay 传递的 request 请求，取得所带参数值。根据不同的参数值得到用户选择的下载方式信息，转入不同的程序分支。如果是单个文件就直接提供下载，如果是多个文件就先将文件打成 Zip 包再下载。

TransVOTable：如果用户选择将查询结果以 VOTable 的格式显示出来，Controller 会把 request 请求转发给该 Servlet。它主要的工作是：捕获 Session，从 Session 中得到结果集和结果集元数据。使用一个专门负责生成 VOTable 格式流的 JAR 包 voi.vowrite.jar，得到 VOTable 格式的字节流，存入字节数组。在 Session 中保存字节数组，并借助 JavaScript 的 replace()窗口函数将当前页面替换为 VOTableDisplay。

VOTableDisplay：捕获 Session，从 Session 中取得字节数组，写入输出流。这样就能在网页上显示出 VOTable 格式的文件了。

CsvDisplay：如果用户选择将查询结果转换为 CSV 格式， Controller 把 request 请求转发给该 Servlet。由于 CSV 格式浏览器不支持显示，所以会要求用户将查询结果保存到一个 CSV 文件里，在本地用 Excel 或别的工具显示。

5.4 Web 应用的组织和发布

5.4.1 Web 应用的组件和目录结构

我们创建了一个名为mybatc的Web应用，该Web应用包括如下组件[14]：

HTML 组件：index.htm

JSP 组件：error.jsp

Servlet 组件：Controller，HtmlDisplay，Download，TransVOTable，VOTableDisplay，CsvDisplay

其他组件：check.js(javascript)，skyserver.css(样式表)，voi.vowrite.jar(引入的进行格式转换的 jar 包)，web.xml(配置文件)

它的目录结构如图 5-4 所示。

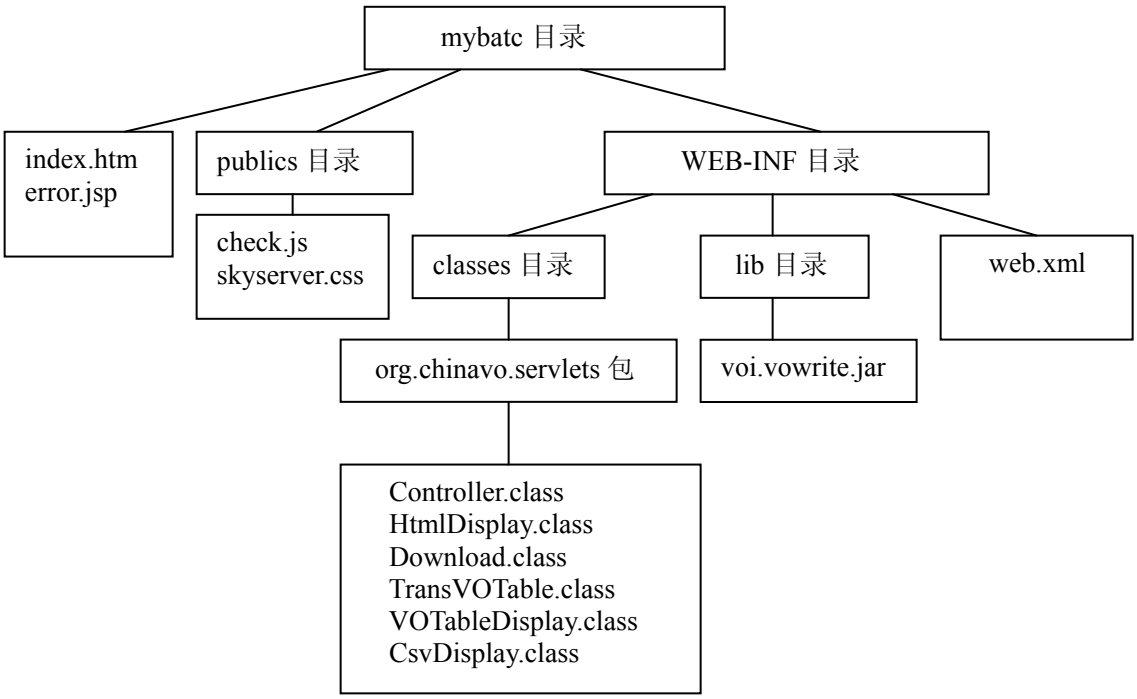

图 5-4　　Web 应用的目录结构

5.4.2 Web 应用的发布

Java Web 应用通过一个基于 XML 的发布描述符文件来配置其发布信息，这个文件名为 web.xml，它存放于 WEB-INF 子目录下。在 web.xml 文件中可包含如下配置信息：

- Servlet 的定义
- Servlet 的初始化参数
- Servlet 以及 JSP 的映射
- 安全域配置参数
- welcome 文件清单
- 资源引用
- 环境变量的定义

在我们创建的 Web 应用中，需要配置的有 Servlet 的信息和数据源信息。数据源的配置在前面已经讲过了，Servlet 的配置方法如下：在文件中加入一个<servlet>标签，表示这个 servlet，它有两个子元素，分别是<servlet-name>标签，表示这个 servlet 的名字；<servlet-class>表示这个 servlet 对应的 Java 类的全称（必须包含所在的包）。第二个子元素是<servlet-mapping>标签，用来表达从 URL 到 servlet 的映射，应当在

里面加入两个子元素，<servlet-name>表示 servlet 的名字，<url-pattern>表示要映射的 URL。

配置完毕后，将整个 mybatc 目录打成 WAR 包，放在<CATALINA_HOME>/webapps 下即可。

第六章　　总结及展望

6.1 总结

经过近一个学期的努力，巡天数据分析入库和发布系统现已基本实现。现将我们所做的工作总结如下：

第一，实现了整个系统的架构。经过同天文台负责人员的不断交流，确定了该系统的总体需求，在此基础上完成了系统架构的工作。这样一来，我们就得到了三个模块：数据平台设计模块、数据导入模块和数据发布平台设计模块。不同的模块负责不同的工作，有独立的接口可以供外部调用，所以拥有良好的独立性。同时各个模块之间由数据流串在一起，相互作用，组成一个有机的整体为用户提供服务。

第二，完成了数据平台的搭建。数据平台的主要部件就是关系型数据库管理系统，我们在这部分所做的工作就是：选择合适的数据库管理系统，创建数据库，设计和创建表，设置权限，建立索引等。

第三，完成了数据导入的代码实现，数据导入的目的就是将数据送入已搭建好的数据平台。它完成的工作包括五个部分：解析文件标题；使用哈希表保存解析后的文件标题信息；解析文件名；创建合适的数据结构保存解析后的文件名信息；将哈希表和文件名信息存入数据库表中。

第四，完成了数据发布平台的创建和部署。数据发布平台建立在数据平台的基础之上，给用户提供了一个友好的查询界面，并对用户的请求做出回应。为此，我们创建了一个 Java Web 应用去实现主要功能。其中，查询界面使用了 HTML 技术，对用户请求的处理是用 Servlet 实现的。此外，还涉及到一些 Web 应用的配置信息。

6.2 展望

经过一段时间的努力，我们已经实现了最初的设计目标。在现有成果的基础之上，我们更深入地考虑了是否可以在一些方面有进一步的改善。但是其中存在着一个值得我们思索的问题，那就是发布平台赋予用户的主动权不够。用户在获取数据的过程中始终处于被动的地位,只能按照发布平台提供的查询界面进行有限的操作，却不能自行编写程序决定怎样处理数据。目前，数据的发布是面向大众的，现有的实现方案应该已经能够满足需求了。但是在不远的将来，我们希望这个发布平台可以被越来越多的专业人士所认可，成为天文研究的有用工具。抱着这样的想法，我们比较粗浅地探究了一下使用别的技术的可能性，最终选择 Web Service 作为下一步努力的方向。

使用Web Service的原因有三个：第一，Web Service服务可以认为是一个作为服务而通过Internet标准发布的简单的应用程序[15]，它向外界暴露出一个能够通过Web进行调用的API。这就是说，你能够用编程的方法通过Web调用来实现某个功能的应用程序。这很好的满足了我们的需求。第二，从我们现有的Web应用转为Web Service服务的代价相对于别的技术要小很多。第三，Web Service技术使用基于XML的消息处理作为基本的数据通讯方式，消除使用不同组件模型、操作系统和编程语言的系统之间存在的差异，有很强的生命力。

当然，这只是初步的构想，面临的问题还有很多，还需要在以后的实践过程中逐一解决。

参考文献

[1] Wells D.C., Greisen E.W. .FITS: a Flexible Image Transport System .Astronomy & Astrophysics supplement，1981，44：363

[2] 杨涛，杨晓云，王群等.MySQL权威指南.北京：机械工业出版社，2004，4

[3] 侯捷．Java 编程思想．北京：机械工业出版社，2004，20

[4] 刘东华，王巍，唐刚．Java 网络编程．北京：中国电力出版社，2001，17

[5] 朱恩从．Tomcat 权威指南．北京：中国电力出版社，2004，55

[6] 邓英材．Servlet与JSP核心技术．北京：人民邮电出版社，2001，2

[7] 王军．MySQL 4从入门到精通．北京：电子工业出版社，2003，112

[8] 孙卫琴，李洪成.Tomcat与Java Web开发技术详解.北京：电子工业出版社，2004，173

[9] 陈海山．深入Java Servlet网络编程．北京：清华大学出版社，2002，110

[10] 梁晶, 王志勇, 付俊明．HTML网络编程实例．北京：中国电力出版社，2002，78

[11] 吴悦, 吴冲华．HTML & Web设计技术与技巧．北京：机械工业出版社，2002，36

[12] 战晓苏．精通 Servlets Java 平台的服务器端编程．北京：清华大学出版社，2002，156

[13] 孙一林, 彭波．Java 数据库编程实例．北京：清华大学出版社，2003,78

[14] 何建波,孙建树,李容莉．Web 站点创建与信息发布．北京：机械工业出版社，1997，88

[15] 张尧学，方存好．主动服务——概念、结构与实现．北京：科学图书馆，2005，3

附　　录

A.数据导入模块中含有 main 方法的 SaveFile 类实现：

```
package org.cvo.BAIS;

import java.io.File;
import java.io.IOException;
import java.sql.SQLException;
import java.util.Hashtable;

public class SaveFile{
	public static void main(String[] args){ //主程序入口
		init();
		System.out.println("The path you have choose is:"+path);
		 File file = new File(path);
		try{
			intoDir(file);
		}catch (IOException e){
			System.out.println("intoDir error");
			e.printStackTrace();
		}
	}

	private static Config conf = null;
	private static String path = null;

	private static void init(){
		conf = new Config("../../../MyVO.properties"); //读取配置文件信息
		path = conf.getAttribute("filepath"); //取得文件主目录路径
		System.out.println(path);
	}

	private static void intoDir(File f) throws IOException{
		try {
			DbControl.connectDb(conf); //连接数据库
		} catch (ClassNotFoundException e1) {
```

```
            e1.printStackTrace();
        } catch (InstantiationException e1) {
            e1.printStackTrace();
        } catch (IllegalAccessException e1) {
            e1.printStackTrace();
        } catch (SQLException e1) {
            e1.printStackTrace();
        }
      File[] file = f.listFiles();
       if(f.isDirectory()){
            for(int i = 0; i < file.length; i++){
               intoDir(file[i]); //递归搜索文件
           }
      }
    else if (f.isFile() && f.getName().endsWith("fit") &&
                                              (f.getName().length()==20)){
           try{
                  FitsFile bf = NameAnalysis.getBaseFits(f);//解析文件名
                  try{
                        if(DbControl.isexist(bf)){
                              System.out.println("the file existed");
                              return;
                        }
                  }catch(Exception e){
                        System.out.println("DbControl.isexist error");
                        e.printStackTrace();
                  }
                  Hashtable htable = HeadAnalysis.getHead(f); //解析文件标
                                                                  题部分
                  FileHead fh = new FileHead(htable);
                  DbControl.insertDb(bf,fh); //插入数据库
           }catch(IOException e){
                  System.out.println("DbControl.inSert error");
                  e.printStackTrace();
           }
      }
      DbControl.closeConn(); //关闭数据库
```

```
    }
}
```

B.数据发布平台中 Servlet 类 MainControl 的实现：

```
package org.chinavo.servlets;

import java.io.ByteArrayOutputStream;
import java.io.IOException;
import java.io.OutputStream;
import java.io.PrintStream;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.ResultSetMetaData;
import java.sql.Statement;

import javax.naming.Context;
import javax.naming.InitialContext;
import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.ServletOutputStream;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;
import javax.sql.DataSource;

import voi.vowrite.VOTable;
import voi.vowrite.VOTableField;
import voi.vowrite.VOTableResource;
import voi.vowrite.VOTableStreamWriter;
import voi.vowrite.VOTableTable;

public class MainControl extends HttpServlet {
      private boolean fIsNumber(String sV, String sR) {
            String sTmp;
            if (sV.length() == 0) {
```

```
            return (false);
        }
        for (int i = 0; i < sV.length(); i++) {

            sTmp = sV.substring(i, i + 1);

            if (sR.indexOf(sTmp, 0) == -1) {
                return (false);
            }
        }
        return (true);
    }

    private String doSelect(HttpServletRequest request,
            HttpServletResponse response, String tableName) throws IOException,
            ServletException {
        String _include = request.getParameter("include");
        String _day = request.getParameter("day");
        if (_day.equals(""))
            _day = "xx";
        String _month = request.getParameter("month");
        if (_month.equals(""))
            _month = "xx";
        String _year = request.getParameter("year");
        if (_year.equals(""))
            _year = "xx";
        String _objectname = request.getParameter("objectname");
        String _observername = request.getParameter("observername");
        String _skyarea = request.getParameter("skyarea");
        String _filter = request.getParameter("filter");
        int isFirstCondition = 0; //judging whether is the first query
        String[] ColumnName = request.getParameterValues("ColumnName");
        int number = ColumnName.length;
        System.out.print(number);
        String strSql = "select path,id";
        for (int i = 0; i < number; i++) {
            strSql = strSql + ",";
```

```
        strSql = strSql + ColumnName[i];
    }
    strSql = strSql + " from " + tableName;
    if (!_include.equals("")) {
        strSql += " where INSTR(name,'" + _include + "')!=0 ";
        isFirstCondition++;
    }
    if (!_objectname.equals("")) {
        if (isFirstCondition == 0)
            strSql += " where ";
        else
            strSql += " and ";
        strSql += " INSTR(OBJECT,'" + _objectname + "')!=0 ";
        isFirstCondition++;
    }
    if (!_observername.equals("")) {
        if (isFirstCondition == 0)
            strSql += " where ";
        else
            strSql += " and ";
        strSql += " INSTR(OBSERVER,'" + _observername + "')!=0 ";
        isFirstCondition++;
    }
    if (!_day.equals("xx") || !_month.equals("xx") || !_year.equals("xx")) {
        if (isFirstCondition == 0)
            strSql += " where ";
        else
            strSql += " and ";
        strSql += " INSTR(DATE_OBS,'" + _day + "')=1 || INSTR(DATE_OBS,'"
                + _month + "')=4 || INSTR(DATE_OBS,'" + _year + "')=7 ";
        isFirstCondition++;
    }
    if (!_skyarea.equals("")) {
        if (isFirstCondition == 0)
            strSql += " where ";
        else
            strSql += " and ";
```

```
        strSql += " skyArea='" + _skyarea + "' ";
        isFirstCondition++;
    }
    if (!_filter.equals("")) {
        if (isFirstCondition == 0)
            strSql += " where ";
        else
            strSql += " and ";
        strSql += " filter='" + _filter + "' ";
        isFirstCondition++;
    }
    String ra = request.getParameter("ra");
    String dec = request.getParameter("dec");
    String radius = request.getParameter("radius");
    if (!ra.equals("") && !dec.equals("") && !radius.equals("")) {
        double _ra = Double.parseDouble(ra);
        double _dec = Double.parseDouble(dec);
        double _radius = Double.parseDouble(radius);
        if (isFirstCondition == 0)
            strSql += " where ";
        else
            strSql += " and ";
        strSql += "( DEC_2000 between "
                + (_dec - _radius)
                + " and "
                + (_dec + _radius)
                + " ) and ( RA_2000 between "
                + _ra
                + "- "
                + _radius
                + "/cos(radians("
                + _dec
                + ")) and "
                + _ra
                + " + "
                + _radius
                + "/cos(radians("
```

```
                    + _dec
                    + "))) and (degrees(acos(sin(radians(DEC_2000))*sin(radians("
                    + _dec + "))+cos(radians(DEC_2000))*cos(radians(" + _dec
                    + "))*cos(radians(RA_2000-" + _ra + "))))<" + _radius + ")";
        }
        return strSql;

    }

    protected void doGet(HttpServletRequest request,
            HttpServletResponse response) throws ServletException, IOException {
        HttpSession session = request.getSession(true);
        String url1 = "/ShowHTML?curPge=-1"; //url with responsibility for
        RequestDispatcher dispatcher1 = getServletContext()
                .getRequestDispatcher(url1);
        DataSource ds = null;
        String tableName = "basedata";
        String strSql = new String();
        strSql = doSelect(request, response, tableName); //编写 SQL 语句
        String _format = request.getParameter("format");
        try {
            Class.forName("org.gjt.mm.mysql.Driver").newInstance();
            Context ctx = new InitialContext();
            ds = (DataSource) ctx.lookup("java:comp/env/jdbc/TestDB"); //寻找数据
                                                                        源
            Connection conn = ds.getConnection();
            Statement stmt = conn.createStatement(
                    ResultSet.TYPE_SCROLL_INSENSITIVE,
                    ResultSet.CONCUR_READ_ONLY);
            ResultSet rs = stmt.executeQuery(strSql);
            ResultSetMetaData rsmd = rs.getMetaData();
            session.setAttribute("rs", rs);
            session.setAttribute("rsmd", rsmd);
            if (_format.equals("html")) {
                dispatcher1.forward(request, response); //将请求转发给 dispatcher1
            } else {
                if (_format.equals("xml")) {//创建 votable 格式输出流
```

```
ByteArrayOutputStream fout = new ByteArrayOutputStream();
PrintStream prnStream = new PrintStream(fout);
VOTableStreamWriter voWrite = new VOTableStreamWriter(
        prnStream);
VOTable vot = new VOTable();
String descString = "Here are the VOTable format of the data";
vot.setDescription(descString);
voWrite.writeVOTable(vot);
VOTableResource voResource = new VOTableResource();
voWrite.writeResource(voResource);
VOTableTable voTab = new VOTableTable();
for (int x = 3; x <= rsmd.getColumnCount(); x++) {
    VOTableField voField = new VOTableField();
    voField.setName(rsmd.getColumnName(x));
    voField.setDataType(rsmd.getColumnTypeName(x));
    voTab.addField(voField);
}
voWrite.writeTable(voTab);
rs.beforeFirst();
StringBuffer buffer = new StringBuffer();
while (rs.next()) {
    for (int m = 3; m <= rsmd.getColumnCount(); m++){
     if(rsmd.getColumnTypeName(m).equals("VARCHAR"))
        buffer.append(rs.getString(m));
   else if(rsmd.getColumnTypeName(m).equals("INTEGER"))
        buffer.append(rs.getInt(m));
    else if (rsmd.getColumnTypeName(m).equals("FLOAT"))
        buffer.append( rs.getFloat(m));
    else
        buffer.append( rs.getObject(m));
    if(m != rsmd.getColumnCount())
        buffer.append(",");
    }
    voWrite.addRow(buffer.toString().split(","),
                                    rsmd.getColumnCount()-2);
}
```

```
            voWrite.endTable();
            voWrite.endResource();
            voWrite.endVOTable();
            session.setAttribute("fout", fout.toByteArray());
            response.setContentType("text/html; charset=GBK");
            response.setHeader("Cache-Control", "no-cache");
            ServletOutputStream out = response.getOutputStream();
            out.println("<html>");
            out.println("<head>");
            out.println("<meta http-equiv='Content-Type' content='text/html
                                                        ;charset=GBK'>");
            out.println("<meta http-equiv='expires' content='0'>");
            out.println(
                    "<meta http-equiv='cache-control' content='no-cache'>");
            out.println(
                    "<meta http-equiv='pragma' content='no-cache'>");
            out.println("</head>");
            out.println("<body>");
            out.println("<script languate='JavaScript'>");
            out.println(
                    "window.location.replace(\"./TransIntoXml\");");
            out.println("</script>");
            out.println("</body>");
            out.println("<html>");
            out.flush();
        } else { //创建 csv 格式输出流
            response.setContentType("unknown");
            String filename = "a.csv";
            response.addHeader("Content-Disposition",
                        "attachment; filename=\"" + filename + "\"");
            OutputStream out = response.getOutputStream();
            StringBuffer line = new StringBuffer(1024);
            for (int j = 3; j <= rsmd.getColumnCount(); j++) {
                line.append(rsmd.getColumnName(j));
                if (j != rsmd.getColumnCount())
                    line.append(",");
            }
```

```
                    line.append("\r\n");
                    while (rs.next()) {
                        for (int m = 3; m <= rsmd.getColumnCount(); m++) {
                         if(rsmd.getColumnTypeName(m).equals("VARCHAR"))
                            line.append(rs.getString(m));
                         else if
                            (rsmd.getColumnTypeName(m).equals("INTEGER"))
                            line.append(rs.getInt(m));
                         else if (rsmd.getColumnTypeName(m).equals("FLOAT"))
                            line.append(rs.getFloat(m));
                         else
                            line.append(rs.getObject(m));
                         if (m != rsmd.getColumnCount())
                            line.append(",");
                        }
                        line.append("\r\n");
                    }
                    byte[] b = line.toString().getBytes();
                    out.write(b);
                    out.flush();
                    out.close();
                }
            }
        }catch (Exception x) {
            x.printStackTrace();
        }
    }

    protected void doPost(HttpServletRequest request,
            HttpServletResponse response) throws ServletException, IOException {

        doGet(request, response); //post 请求与 get 请求处理相同
    }
}
```

中文译文

面向 Web 的组件合成和交互式结构

摘要：

Web 正在经历着从以文档为中心的环境到以服务为中心的环境的转变。这种转变可以被看作是向以组件为中心的环境转变的第一步。我们应该探究一下基于最近发展很快的 Web 服务框架的 Web 组件体系结构的需求。描述性语言、协议、知识库以及目录服务都是其中的关键部分。我们将就这些方面提出一个根本的概念模型去捕获他们的基础。我们将为 Web 组件框架确定两个关键特征——捕获了两种不同行为方面的一个两层类型系统以及组件的语义描述——这两点使得它与服务环境不同。

1. 介绍

自从 1990s 初期诞生以来，Web 在不停的演化过程中。最初它被设计为一个可以允许用户将他们的文档标记为超文本文件而变得可访问同时也能通过允许超文本文件传送的协议去访问他人文档的发布框架，而现在它已经演化为一个更加动态和交互的环境。它现在正被用于当初并没有计划到的用途。Web 从数据传输来说已经变成双向的了。最近，变革性的重要一步正在发展中，那就是将 Web 从以文档为中心的环境转到以应用程序或服务为中心的环境中去。用户将被提供访问服务的可能性而不仅仅是访问数据。这个过程也使得 Web 基础设施中应用程序对应用程序的访问变得可用。这些尝试尚集中于独立的服务，而近几年来组件技术的成功值得我们考虑一下在 Web 环境中组件的合成和交互。

现在 Web 技术研究和发展的焦点在于服务——通常叫做 Web 服务。我们在这里将探究 Web 作为组件合成和组件交互的体系结构的使用。若干个服务组成封装有内部状态的组件。除了通过输出接口提供服务外，组件还有一个用以记录所需服务的明确的输入接口，以便能根据规范工作。大多数组件描述的方法说明了额外的语义信息去描述服务 。以前提条件和后备条件的形式表述的合同信息是一种典型的选择。如果组件用于构造更大的系统，请求方服务和提供方服务必须相配。一致性原则是用以描述对管理组件匹配的限制的。除了匹配之外的另一种活动是在客户和服务提供者之间的交互。组件服务的建立跟独立的服务是一样的，除了组件状态可能会变。

带有输入和输出接口以及匹配性的 Web 组件的日益增长的复杂性提出了 Web 组件框架的构架问题。从本质上讲，也就是要为 Web 组件寻求一个分布式的计算模型。Web 服务推荐的体系结构包括一个服务建立协议、一种服务描述语言以及一个目录工具。如果想把 Web 服务框架扩展为 Web 组件框架，我们需要语言来支持组件接口的语义描述，需要将协议扩展为包括匹配和交互在内的两个方面，还需要一组服务用语查找、匹配、分析、通讯等。这种构架只有当它标准化时才能算是成功。本文的目的就是要就 Web 组件体系结构的发展和标准化提出一些观点，并且评价 Web 服务和其他的像 CORBA 那样的面向对象技术的框架中的一些概念的适用性。最终的目的是组件合成体系结构的发展。

在[4]里已经研究过从 Web 服务到 Web 组件的扩展。我们将他们的工作更进一步。我们阐明了层的概念，反映出组件的合成包括两部分：匹配——有时也叫做链接——以及交互。像 COM 这样的技术是跟交互有关的；模型系统是跟链接有关的。另一个没有深入考虑的是 Web 组件的语义描述，它影响着性能和构架并且暗示了在这样一种环境中可能有的服务。我们将会关注同步交互并且强调组件的描述以及组件间的匹配。为了阐明一些关键的观点，我们将讨论一些概念和服务区支持这些观点，对所有方面的完全讨论不是本文的目标。

在第 2 部分我们展示了 Web 服务框架的原则。然后，我们在第 3 部分讨论了这个框架用于 Web 组件技术的不足之处。第 4 部分描述了构成 Web 组件体系结构基础的规范模型的关键概念。这个 Web 组件体系结构放在第 5 章进行描述。我们的注意力放在描述语言这里。我们以相关工作和一些结论作为结束语。

2.Web 服务——简单介绍

Web 服务框架的目的是将 Web 从以文档为中心的环境移到以服务为中心的环境中去。它的目标是使 Web 中应用程序对应用程序的使用变得可用——Web 目前尚且是一个从根本上为人所使用的环境。Web 技术——语言和协议——被用来提供远程过程调用机制。协议是基于 XML 方式传送的以确保达到最大互操作性。

没有语义信息的单一服务可以用 Web 服务描述语言 WSDL 描述。一个 Web 服务描述包括 5 个部分。一个抽象的、独立于协议的部分包括类型、数据和操作描述。操作部分——被叫做端口类型——描述了根据特定类型的输入输出参数并实现了服务功能的操作。这些参数放在数据部分进行描述——被称为消息。消息的类型可以在独立的类型段进行定义。对特定协议的绑定是服务描述中 2 个具体部分项中的一个。它描述了如何使用考虑的协议去激活服务。这一项叫做“绑定”，最后一项叫做服务，它将服务链接到服务可被找到的具体位置。

Web 服务激活和回复的基础设施是由 SOAP 协议实现的——可以影响 XML

协议的标转化。SOAP——简单对象访问协议——是一种用于服务请求和回复的基于 XML 的协议。它被设计用来支持被 WSDL 所描述的服务的远程启动。服务的发现由目录框架 UDDI——通用描述发现集成所支持。UDDI 可以当作是组件的市场。

3.Web 服务——分析

3.1 服务和组件

在介绍部分我们已经指出了服务和组件的一些不同之处。下面我们从细节上继续这个讨论。下面几个方面从一般意义上讨论了组件与服务的不同——不考虑 Web 环境的具体差别。

输入和输出接口：除了组件实现的服务外，组件接口还描述了用来履行组件职
责的服务属性。

语义信息：服务从句法上和语义上进行描述。对服务的语义描述建立在按约定
设计的方法之上。使用前置和后置条件技术的不言自明的描述是允许的。

匹配：请求客户端组件和服务提供者组件的一致性——提供者要匹配客户端需求——是需要被考虑的。我们可以通过一个类型系统去表达一致性的概念。类型一致性和子类型使一致性变得规范。可以使用一个为自明语义描述使用的类型准则的方法。

动态配置：首先介绍一下连接的概念。当两个组件组合在一起时，会创建一个客户和服务器间的私有连接。匹配可能涉及也可能不涉及一个代理和合成者。服务提供者和客户间的连接可以持续，客户可以多次使用同一连接。如果服务进行交互，组件的状态可能会被改变。在演化型系统中，新增组件、但各组件的重新配置或者组件系统的重新配置都会导致组件连接空间结构的一直改变。

生命周期：组件在进行交互前需要先匹配，这是要遵守的一个协议。本质上说，这是包括匹配和交互在内的两个阶段的协议，但如果要考虑动态重配置的话还要对其进行扩展。

3.2 Web 服务框架的适用性

由于前面对组件特征的总结，我们现在开始讨论 Web 服务框架。我们把注意力放在组件的描述上。下面我们讲解 WSDL 中 Web 服务描述的元素。

Types：这个元素建立在通用框架——XML 模式语言的基础上。更高级的顺序连接类型需要被介绍以捕获空间组合结构的动态配置。

Messages：消息可以是一个连接类型，反映了连接需要传送自身以创建和改

变组件间私有连接。

Port types：端口类型需要分成入端口和出端口，以及分成匹配端口和交互端口。后者的类型与阶段有关，前者描述了一个服务是输入接口还是输出接口的一部分。

Binding：对于一个服务有一些绑定需要介绍一下，例如匹配绑定和交互绑定。

Services：与 Web 服务相比没有改变。

一个目前尚未解决的实质性问题是在哪里提供对协议或生命周期的描述。明显的，协议部分应该同匹配约束分开。可能的情况包括端口类型和绑定。Web 服务连接是一次性激活的，而组件连接是由可持续性的可多次使用。

现有的 Web 服务与正在探索过程中的分布式计算模型的最大不同是 Web 服务主要注意力集中在由 WSDL 定义的消息格式上，而实际需要的却是对应用程序编程接口的关注。第二个不同之处在于 Web 服务框架中的合成不是专门的。一个组件框架是需要满足动态合成功能的。对于组件来说，一个协商的过程，例如基于约定的，是需要的。实际上，对于栈中每一层都需要高级支持。

4.Web 组件结构——基本原理

现在我们能列出 Web 组件概念模型的元素了——基本是对这个模型的需求说明，包括像类型系统和传送系统这种公式。

4.1 核心模型的元素

端口是一个指向组件服务的抽象访问。端口描述是组件接口的一部分。端口类型可以反映各种属性——像端口的截然对立和方向（输入或者输出）、作用（是否设计匹配组件或与组件交互）以及传输带宽。端口类型可用于表达结构上的和行为上的限制。一个协议的结束点（例如 SOAP 结束点）通常是担任不同作用的一组端口。

类型系统尤其是子类型在其中扮演了主要角色。子类型可以决定为一个服务请求准备何种合适的匹配。子类型的标准定义——一个子类型的实例可以在任何一个需要超类型实力的上下文中使用——阐明了匹配中一致性的实质。

合成的行为可以被分为不同的阶段。我们可以区分开匹配阶段和交互阶段，成功匹配后连接才被建立起来用以交互。服务激活和服务响应时需要这些连接。这些连接可以是以持续一段时间组件间的私有连接。这些活动反映出了组件生命周期。组件生命周期——在交互前先匹配——需要有合成协议将其正式化。这个会影响各个单独的组件，同时也影响组件的合成。协议限制有合适的转换规则表达。

4.2 高级概念

既然我们将因特网看作 Web 组件框架的基本的基础设施，我们能想到一些在我们核心模型的表述中没有涉及到的一些高级的方面。这些是分布性、可移动性和安全性。因特网是一个分布式的网络环境，分布式位置信息必须要描述出来。Java 是一个典型的以 applet 形式的移动计算的因特网编程平台的例子。安全也是在像因特网这样的开放的和分布式的环境中肯定要考虑的因素。

另一个因素——对因特网不是独一无二的但却是很重要的——就是演化性。变化的环境和需求是会影响任何一种软件系统的。

4.3 适合的正式框架

如果基于语义描述的分析和推理服务被提供，一个对于 Web 组件的正式定义的概念模型是必需的。类型系统和基于状态的变换的概念是至关重要的。阐述这个模型的合适的框架是有着模式性、灵活性和安全性的过程演算——例如圆周率。

在[12]里，我们展示了一个基于典型圆周率演算的满足所列需求的针对组件合成的正式的框架。同样我们还讲述了组件系统的动态配置和替换。

使用典型的过程模型去描述组件或对象间交互并不是很性的技术。Nierstras 开发了一个正式的类型理论化的对象框架。对象具有规范交互过程的特征，一个两层的类型系统区分了服务类型（合同）和规范类型（协议）。两个子概念——服务类型和规则类型——定义了客户和服务提供者的满意性的概念以及对象的可替代性。Nierstrasz 强调了两种不同类型的正交性。

5.Web 组件架构

Web 组件架构应该包括：

- 描述语言：语义组件描述
- 匹配和交互协议：2 个阶段的合成
- 一组服务：发现，匹配，配置，替换，交互

这种架构将描述一个组件中间件平台。描述这些语言、协议和服务的正式模型

的定义在第 4 部分已经介绍。描述语言和协议忽略了关于如何发现组件、组件如何存储和变得可用的细节。这些方面可以有特定服务例如代理服务所支持。大量服务依赖于通过描述语言变得可用的语义规范。有一些支持性协议。尽管如此，我们这里只讲述一个必要的元素——描述语言。

5.1 Web 组件描述语言

在概念模型的基础上，需要定义一个用于描述 Web 组件的语言，我们叫做 Web 组件描述语言。我们通过一个图表的例子讲述这种语言——一个全面的定义是在本文范围之外的——且遵循着底层 WSDL 的结构。类型——数据类型、端口类型、连接类型——在这里不予详述。重要的是子类型定义的支持。

我们定义两条信息——数据项和连接。

```
<message name="InData">
    <part name="body" element="dataType"/> </message>
<message name="InConnection">
    <part name="body" element="connectionType"/> </message>
```

端口类型定义了基于这些信息的服务，在我们的例子中是一个独立的服务操作ServOp。

```
<portType name="service">
    <operation-contract name="servOp"
                        precondition = "..."
                        postcondition = "..."
                        signature = "..." >
        <input message="..." type="connection"/>
    </operation>
    <operation-connection name="servOp">
        <input message="..." type="data"/>
        <input message="..." type="connection"/>
        <output message="..." type="data"/>
    </operation>
</portType>
```

WSDL的其余部分是关于具体的部分的，例如协议的绑定和位置的关联等。WCDL的绑定部分需要将匹配绑定和交互绑定分离出来。后者需要描述激活和回复。服务部分表述了服务的位置。这一部分和WSDL没什么不同。

5.2组件合成和交互

独立的服务和合同信息在前边子章节中已经讲过了。一个独立的、本体的方面是以匹配、交互、替换等活动表现出来的组件的整体行为。组件的生命周期可以以协议的形式来规范化。我们打算用 XML 格式的标记描述协议的详细内容。这些操作的使用以组件生命周期的形式表示出来——这里是客户请求服务然后同服务不断交互的例子：

```
<sequence>
```

```
<request name="servOp"
                precondition = "..."
                postcondition = "..."
                signature = "..." />
<repeat>
  <sequence>
      <invoke name="servOp"> ... </invoke>
      <receive name="servOp"> ... </receive>
  </sequence>
</repeat>
</sequence>
```

5.3 实现

为了研究这里所列出的概念和想法的可行性,我们已经开始实现一个基于集中式代理服务的原型。这个代理原型是在一个标准的、基于 Web 的、包含一个匹配服务器和一个交互服务器的三层结构的基础上实现的。匹配服务器在一个只包含组件接口的组件仓库的基础上工作——组件自身实现位于别处。XML 格式的消息用于交换匹配和交互相关的数据。代理原型包含一个用于匹配的接口，留给组件系统开发者使用。

6.相关工作

对于基于 Web 环境的语义信息讨论一定要考虑语义的 Web 活动。RDF——资源描述框架——是这个活动的核心。RDF 的详细说明就是提供一个支持 Web 上信息交流的轻量级本体论系统的 XML 的详细描述。在一个预定义的本体论中，通过注释（或元数据）提供了更有效的发现和自动化。它更适合于定义一个用于 Web 服务和组件语义描述的框架。在这篇文章中讨论的基于逻辑的语言可用来支持组件合成。

结构框架用于分布式对象的交互——例如 CORBA 或者 COM/DCOM。我们已经从 CORBA 里我们感兴趣的分布式组件交互中考虑了一些想法。CORBA 的特征，例如方法调用、存根/构架、服务和协议都有它们在组件技术中的对应性。

这里讨论的第二种适用于 Web 组件的框架是 Web 服务，详见前一节。在文献[4]中，在 Web 服务平台的组件模型都是相同的。众所周知，对组件方面的强化将会极大的提高平台性能，这里我们试图指出那个平台的不足之处。

一些组织已经实现了因特网的组件系统，其中有手机工程和组件 X 交换系统。手机是手机工程的主要实体。它致力于在分布式、对安全要求严的环境中支持动态组件。因特网是目标平台。前者为组件合成实现了一个两阶段的系统。后

者着重于匹配活动——也可以叫做交易。组件 X 交换系统是一个代理系统——叫做交易人——控制在基于网络的环境中合适的服务和组件的发现和匹配。它特别用于像许可认证这样的商业领域。

7.结论

可以提供一个框架，该框架用于比 Web 服务框架或针对对象的分布计算模型需要更多高级特征的 Web 组件。两个方面导致了这些差别。首先，匹配（或链接）和交换需要分开进行，结果是产生了两个不同的行为阶段，或者是两个不同的结构层。其次，语义表现的描述提高了复杂度，但是也提供了新的机会，这些机会还需要合适的服务来支持。

致　　谢

回忆整个论文完成的过程，很多人给了我莫大的关心与帮助。

首先感谢我的指导老师孙济洲教授对我做了精心的指导，给了我许多宝贵的建设性意见，并为我提供了无比优越的研究环境。他严谨、认真的作风深深地影响着我，使我得以顺利地完成毕业设计的内容。

感谢我的组长于策以及项目负责人陆鄂丰，在整个毕业设计过程中，给我耐心、细致的指导，并提出大量很有价值的建议和意见。

非常感谢我的师兄肖健，师姐孙敏。他们给了我大量有价值的建议，耐心解答我不懂的问题，为我提供了丰富的科研资源和重要信息，这也是我能够顺利完成论文的重要原因。

非常感谢与我一起做毕业设计的同学，与他们一起讨论、互通信息，使我受益匪浅。

最后，感谢我的父母及朋友。是他们在生活上和学习上给我无数的鼓励与关心，才使我走到今天。