# 研发中心学习计划

2018年12月24日　　10:14

第一周

开发环境和开发规范

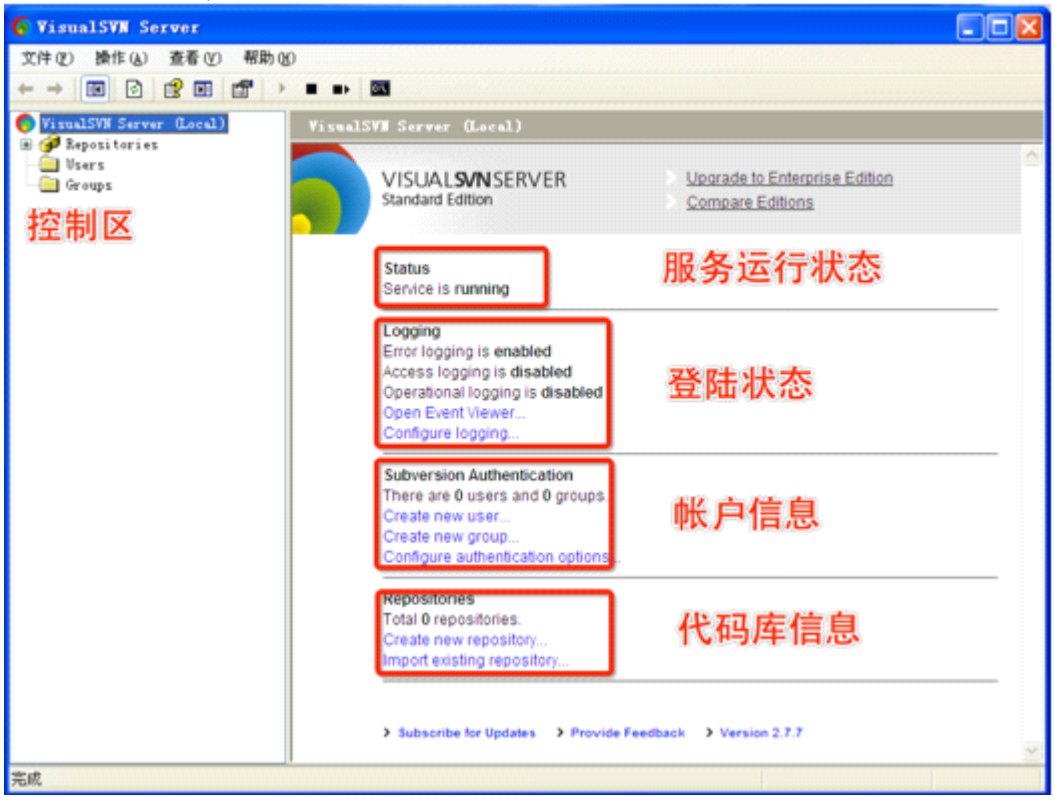
开发工具eclipse

运行环境 jdk1.8 maven3.5.0

svn代码管理 svn3.9.2

安装路径：C:\Program Files\VisualSVN Server

仓库路径：F:\Repositories




代码规范 https://github.com/waylau/java-code-conventions （java编码规范）

https://waylau.com/java-code-conventions/

命名规范

数据库

plsql工具的运用

https://www.yiibai.com/plsql/plsql_environment_setup.html

oracle安装、配置、创建、备份

https://www.oracle.com/technetwork/cn/database/enterprise-edition/downloads/index.html

应用服务器

应用服务器tomcat安装配置

https://tomcat.apache.org/index.html

jquery基础

Juery 下载

http://jquery.com/download/

标签库

Jstl

Structs

SSH框架

Structs + Spring + Hibernate

# JSP 标准标签库（JSTL）

星期一，十二月 24，2018　　11:49 上午

## JSP 标准标签库（JSTL）

JSP标准标签库（JSTL）是一个JSP标签集合，它封装了JSP应用的通用核心功能。

JSTL支持通用的、结构化的任务，比如迭代，条件判断，XML文档操作，国际化标签，SQL标签。 除了这些，它还提供了一个框架来使用集成JSTL的自定义标签。

根据JSTL标签所提供的功能，可以将其分为5个类别。

- **核心标签**
- **格式化标签**
- **SQL 标签**
- **XML 标签**
- **JSTL 函数**

### JSTL 库安装

Apache Tomcat安装JSTL 库步骤如下：

从Apache的标准标签库中下载的二进包(jakarta-taglibs-standard-current.zip)。

下载 **jakarta-taglibs-standard-1.1.2.zip** 包并解压，将 **jakarta-taglibs-standard-1.1.2/lib/** 下的两个 jar 文件：**standard.jar** 和 **jstl.jar** 文件拷贝到 **/WEB-INF/lib/** 下。

将 tld 下的需要引入的 tld 文件复制到 WEB-INF 目录下。

接下来我们在 web.xml 文件中添加以下配置：

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app version="2.4"
    xmlns="http://java.sun.com/xml/ns/j2ee"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee
        http://java.sun.com/xml/ns/j2ee/web-app_2_4.xsd">
    <jsp-config>
    <taglib>
    <taglib-uri>http://java.sun.com/jsp/jstl/fmt</taglib-uri>
    <taglib-location>/WEB-INF/fmt.tld</taglib-location>
    </taglib>
    <taglib>
    <taglib-uri>http://java.sun.com/jsp/jstl/fmt-rt</taglib-uri>
    <taglib-location>/WEB-INF/fmt-rt.tld</taglib-location>
    </taglib>
    <taglib>
    <taglib-uri>http://java.sun.com/jsp/jstl/core</taglib-uri>
    <taglib-location>/WEB-INF/c.tld</taglib-location>
    </taglib>
```

```
    <taglib>
    <taglib-uri>http://java.sun.com/jsp/jstl/core-rt</taglib-uri>
    <taglib-location>/WEB-INF/c-rt.tld</taglib-location>
    </taglib>
    <taglib>
    <taglib-uri>http://java.sun.com/jsp/jstl/sql</taglib-uri>
    <taglib-location>/WEB-INF/sql.tld</taglib-location>
    </taglib>
    <taglib>
    <taglib-uri>http://java.sun.com/jsp/jstl/sql-rt</taglib-uri>
    <taglib-location>/WEB-INF/sql-rt.tld</taglib-location>
    </taglib>
    <taglib>
    <taglib-uri>http://java.sun.com/jsp/jstl/x</taglib-uri>
    <taglib-location>/WEB-INF/x.tld</taglib-location>
    </taglib>
    <taglib>
    <taglib-uri>http://java.sun.com/jsp/jstl/x-rt</taglib-uri>
    <taglib-location>/WEB-INF/x-rt.tld</taglib-location>
    </taglib>
    </jsp-config>
</web-app>
```

使用任何库，你必须在每个 JSP 文件中的头部包含 <taglib> 标签。

## 核心标签

核心标签是最常用的 JSTL标签。引用核心标签库的语法如下：

<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %> 标签 描述

| | |
|---|---|
| <c:out> | 用于在JSP中显示数据，就像<%= ... > |
| <c:set> | 用于保存数据 |
| <c:remove> | 用于删除数据 |
| <c:catch> | 用来处理产生错误的异常状况，并且将错误信息储存起来 |
| <c:if> | 与我们在一般程序中用的if一样 |
| <c:choose> | 本身只当做<c:when>和<c:otherwise>的父标签 |
| <c:when> | <c:choose>的子标签，用来判断条件是否成立 |
| <c:otherwise> | <c:choose>的子标签，接在<c:when>标签后，当<c:when>标签判断为false时被执行 |
| <c:import> | 检索一个绝对或相对 URL，然后将其内容暴露给页面 |
| <c:forEach> | 基础迭代标签，接受多种集合类型 |
| <c:forTokens> | 根据指定的分隔符来分隔内容并迭代输出 |
| <c:param> | 用来给包含或重定向的页面传递参数 |
| <c:redirect> | 重定向至一个新的URL. |

<c:url> 使用可选的查询参数来创造一个URL

## 格式化标签

JSTL格式化标签用来格式化并输出文本、日期、时间、数字。引用格式化标签库的语法如下：

```
<%@ taglib prefix="fmt"
           uri="http://java.sun.com/jsp/jstl/fmt" %>
```

## SQL标签

JSTL SQL标签库提供了与关系型数据库（Oracle，MySQL，SQL Server等等）进行交互的标签。引用SQL标签库的语法如下：

```
<%@ taglib prefix="sql"
           uri="http://java.sun.com/jsp/jstl/sql" %> 标签 描述
```

| | |
|---|---|
| <sql:setDataSource> | 指定数据源 |
| <sql:query> | 运行SQL查询语句 |
| <sql:update> | 运行SQL更新语句 |
| <sql:param> | 将SQL语句中的参数设为指定值 |
| <sql:dateParam> | 将SQL语句中的日期参数设为指定的java.util.Date 对象值 |
| <sql:transaction> | 在共享数据库连接中提供嵌套的数据库行为元素，将所有语句以一个事务的形式来运行 |

## XML 标签

JSTL XML标签库提供了创建和操作XML文档的标签。引用XML标签库的语法如下：

```
<%@ taglib prefix="x"
           uri="http://java.sun.com/jsp/jstl/xml" %>
```

在使用xml标签前，你必须将XML 和 XPath 的相关包拷贝至你的<Tomcat 安装目录>\lib下:

标签 描述

| | |
|---|---|
| <x:out> | 与<%= ... >,类似，不过只用于XPath表达式 |
| <x:parse> | 解析 XML 数据 |
| <x:set> | 设置XPath表达式 |
| <x:if> | 判断XPath表达式，若为真，则执行本体中的内容，否则跳过本体 |
| <x:forEach> | 迭代XML文档中的节点 |
| <x:choose> | <x:when>和<x:otherwise>的父标签 |

| | |
|---|---|
| <x:when> | <x:choose>的子标签，用来进行条件判断 |
| <x:otherwise> | <x:choose>的子标签，当<x:when>判断为false时被执行 |
| <x:transform> | 将XSL转换应用在XML文档中 |
| <x:param> | 与<x:transform>共同使用，用于设置XSL样式表 |

## JSTL函数

JSTL包含一系列标准函数，大部分是通用的字符串处理函数。引用JSTL函数库的语法如下：

```
<%@ taglib prefix="fn"
           uri="http://java.sun.com/jsp/jstl/functions" %> 函数 描述
```

| | |
|---|---|
| fn:contains() | 测试输入的字符串是否包含指定的子串 |
| fn:containsIgnoreCase() | 测试输入的字符串是否包含指定的子串，大小写不敏感 |
| fn:endsWith() | 测试输入的字符串是否以指定的后缀结尾 |
| fn:escapeXml() | 跳过可以作为XML标记的字符 |
| fn:indexOf() | 返回指定字符串在输入字符串中出现的位置 |
| fn:join() | 将数组中的元素合成一个字符串然后输出 |
| fn:length() | 返回字符串长度 |
| fn:replace() | 将输入字符串中指定的位置替换为指定的字符串然后返回 |
| fn:split() | 将字符串用指定的分隔符分隔然后组成一个子字符串数组并返回 |
| fn:startsWith() | 测试输入字符串是否以指定的前缀开始 |
| fn:substring() | 返回字符串的子集 |
| fn:substringAfter() | 返回字符串在指定子串之后的子集 |
| fn:substringBefore() | 返回字符串在指定子串之前的子集 |
| fn:toLowerCase() | 将字符串中的字符转为小写 |
| fn:toUpperCase() | 将字符串中的字符转为大写 |
| fn:trim() | 移除首位的空白符 |

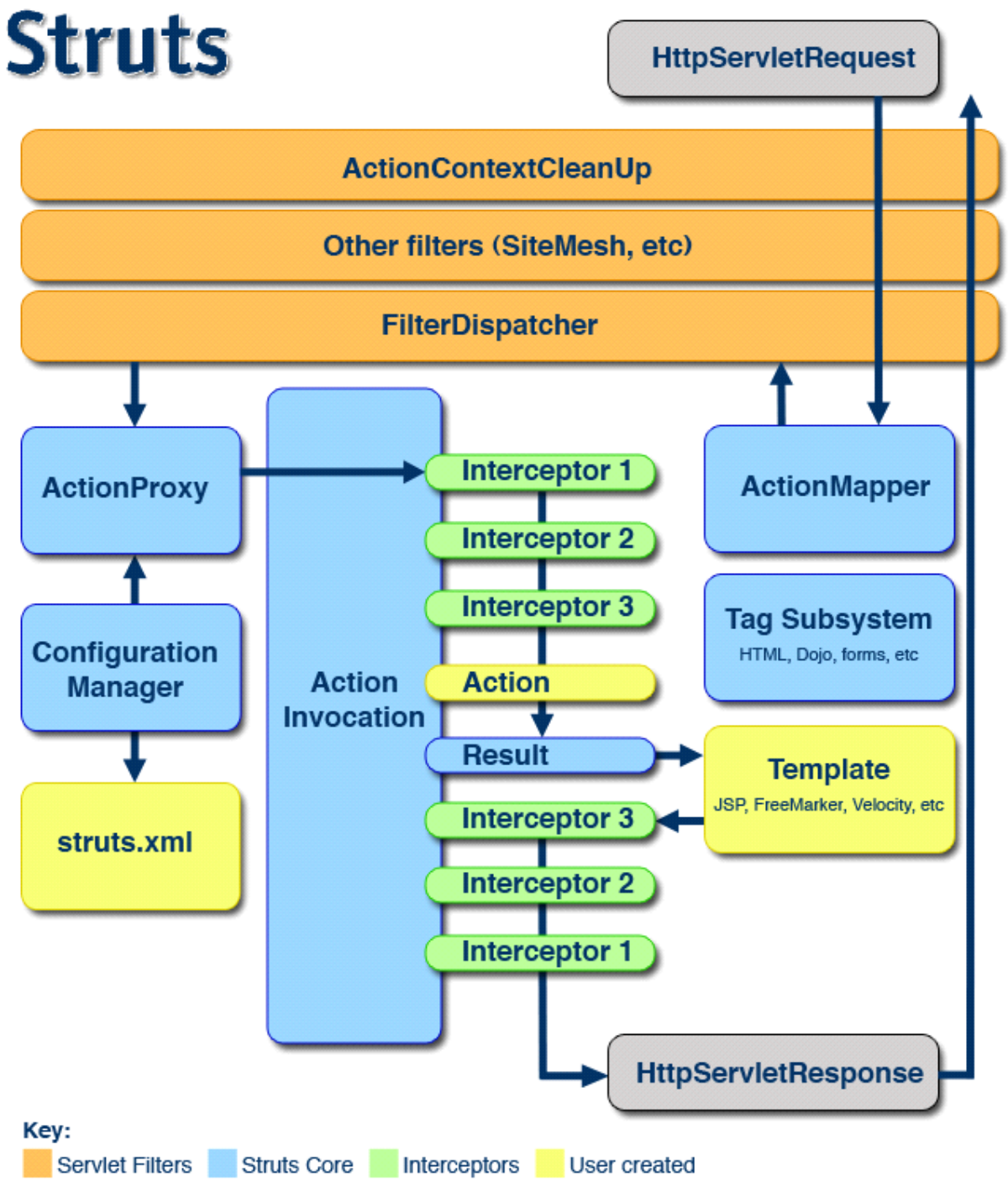
# structs2框架介绍

星期一，十二月 24，2018　　2:00 下午

已剪辑自：https://blog.csdn.net/chainiao_zhang/article/details/76825171

## Struts2面试题

### 1、struts2工作流程




Struts 2框架本身大致可以分为3个部分：

核心控制器FilterDispatcher、业务控制器Action和用户实现的企业业务逻辑组件。

核心控制器FilterDispatcher是Struts 2框架的基础，

包含了框架内部的控制流程和处理机制。

业务控制器Action和业务逻辑组件是需要用户来自己实现的。

用户在开发Action和业务逻辑组件的同时，还需要编写相关的配置文件，

供核心控制器FilterDispatcher来使用。

Struts 2的工作流程相对于Struts 1要简单，与WebWork框架基本相同，

所以说Struts 2是WebWork的升级版本。基本简要流程如下：

1 、客户端初始化一个指向Servlet容器的请求；

2、 这个请求经过一系列的过滤器（Filter）

（这些过滤器中有一个叫做ActionContextCleanUp的可选过滤器，

这个过滤器对于Struts2和其他框架的集成很有帮助，例如：SiteMesh Plugin）

3 、接着FilterDispatcher被调用，

FilterDispatcher询问ActionMapper来决定这个请是否需要调用某个Action

4、如果ActionMapper决定需要调用某个Action，

FilterDispatcher把请求的处理交给ActionProxy

5、ActionProxy通过Configuration Manager询问框架的配置文件，

找到需要调用的Action类

6、ActionProxy创建一个ActionInvocation的实例。

7、ActionInvocation实例使用命名模式来调用，

在调用Action的过程前后，涉及到相关拦截器（Intercepter）的调用。

8、一旦Action执行完毕，ActionInvocation负责根据struts.xml中的配置找到对应的返回结果 。返回结果通常是（但不总是，也可 能是另外的一个Action链）一个需要被表示的JSP或者FreeMarker的模版。 在表示的过程中可以使用Struts2 框架中继承的标签。 在这个过程中需要涉及到ActionMapper

9、响应的返回是通过我们在web.xml中配置的过滤器

10、如果ActionContextCleanUp是当前使用的，则FilterDispatecher将不会清理sreadlocal ActionContext;如果ActionContextCleanUp不使用，则将会去清理

sreadlocals。

## 2、说下Struts的设计模式

MVC模式：web应用程序启动时就会加载并初始化ActionServler。用户提交表单时，一个配置好的ActionForm对象被创建，并被填入表单相应的数据，ActionServler根据Struts-config.xml文件配置好的设置决定是否需要表单验证，如果需要就调用ActionForm的Validate（）验证后选择将请求发送到哪个Action，如果Action不存在，ActionServlet会先创建这个对象，然后调用Action的execute（）方法。Execute（）从ActionForm对象中获取数据，完成业务逻辑，返回一个ActionForward对象，ActionServlet再把客户请求转发给ActionForward对象指定的jsp组件，ActionForward对象指定的jsp生

成动态的网页，返回给客户。

## 3、拦截器和过滤器的区别

1、拦截器是基于java反射机制的，而过滤器是基于函数回调的。
2、过滤器依赖于servlet容器，而拦截器不依赖于servlet容器。
3、拦截器只能对Action请求起作用，而过滤器则可以对几乎所有请求起作用。
4、拦截器可以访问Action上下文、值栈里的对象，而过滤器不能。
5、在Action的生命周期中，拦截器可以多次调用，而过滤器只能在容器初始化时被调用一次。

## 4、struts1于struts2的比较

1、Action 类:
Struts1要求Action类继承一个抽象基类。Struts1的一个普遍问题是使用抽象类编程而不是接口。
Struts 2 Action类可以实现一个Action接口，也可实现其他接口，使可选和定制的服务成为可能。Struts2提供一个ActionSupport基类去 实现常用的接口。Action接口不是必须的，任何有execute标识的POJO对象都可以用作Struts2的Action对象。

2、线程模式:
Struts1 Action是单例模式并且必须是线程安全的，因为仅有Action的一个实例来处理所有的请求。单例策略限制了Struts1 Action能作的事，并且要在开发时特别小心。Action资源必须是线程安全的或同步的。
Struts2 Action对象为每一个请求产生一个实例，因此没有线程安全问题。（实际上，servlet容器给每个请求产生许多可丢弃的对象，并且不会导致性能和垃圾回收问

题）

3、Servlet 依赖:

Struts1 Action 依赖于Servlet API ,因为当一个Action被调用时HttpServletRequest 和 HttpServletResponse 被传递给execute方法。

Struts 2 Action不依赖于容器，允许Action脱离容器单独被测试。如果需要，Struts2 Action仍然可以访问初始的request和response。但是，其他的元素减少或者消除了直接访问HttpServetRequest 和 HttpServletResponse的必要性。

4、可测性:

测试Struts1 Action的一个主要问题是execute方法暴露了servlet API（这使得测试要依赖于容器）。一个第三方扩展－－Struts TestCase－－提供了一套Struts1的模拟对象（来进行测试）。

Struts 2 Action可以通过初始化、设置属性、调用方法来测试，“依赖注入”支持也使测试更容易。

5、捕获输入:

Struts1 使用ActionForm对象捕获输入。所有的ActionForm必须继承一个基类。因为其他JavaBean不能用作ActionForm，开发者经 常创建多余的类捕获输入。动态Bean（DynaBeans）可以作为创建传统ActionForm的选择，但是，开发者可能是在重新描述(创建)已经存 在的JavaBean（仍然会导致有冗余的javabean）。

Struts 2直接使用Action属性作为输入属性，消除了对第二个输入对象的需求。输入属性可能是有自己(子)属性的rich对象类型。Action属性能够通过 web页面上的taglibs访问。Struts2也支持ActionForm模式。rich对象类型，包括业务对象，能够用作输入/输出对象。这种 ModelDriven 特性简化了taglib对POJO输入对象的引用。

6、表达式语言：

Struts1 整合了JSTL，因此使用JSTL EL。这种EL有基本对象图遍历，但是对集合和索引属性的支持很弱。

Struts2可以使用JSTL，但是也支持一个更强大和灵活的表达式语言－－"Object Graph Notation Language " (OGNL).

7、绑定值到页面（view）:

Struts 1使用标准JSP机制把对象绑定到页面中来访问。

Struts 2 使用 "ValueStack "技术，使taglib能够访问值而不需要把你的页面（view）和对象绑定起来。ValueStack策略允许通过一系列名称相同但类型不同的属性重用页面（view）。

8、类型转换：

Struts 1 ActionForm 属性通常都是String类型。Struts1使用Commons-Beanutils进行类型转换。每个类一个转换器，对每一个实例来说是不可配置的。

Struts2 使用OGNL进行类型转换。提供基本和常用对象的转换器。

9、校验：

Struts 1支持在ActionForm的validate方法中手动校验，或者通过Commons Validator的扩展来校验。同一个类可以有不同的校验内容，但不能校验子对象。

Struts2支持通过validate方法和XWork校验框架来进行校验。XWork校验框架使用为属性类类型定义的校验和内容校验，来支持chain校验子属性

10、Action执行的控制：

Struts1支持每一个模块有单独的Request Processors（生命周期），但是模块中的所有Action必须共享相同的生命周期。

Struts2支持通过拦截器堆栈（Interceptor Stacks）为每一个Action创建不同的生命周期。堆栈能够根据需要和不同的Action一起使用。

## 为什么要使用Struts2

Struts2 是一个相当强大的Java Web开源框架，是一个基于POJO的Action的MVC Web框架。它基于当年的Webwork和XWork框架，继承其优点，同时做了相当的改进。

1.Struts2基于MVC架构，框架结构清晰，开发流程一目了然，开发人员可以很好的掌控开发的过程。

2使用OGNL进行参数传递。

OGNL提供了在Struts2里访问各种作用域中的数据的简单方式，你可以方便的获取Request，Attribute，Application，Session，Parameters中的数据。大大简化了开发人员在获取这些数据时的代码量。

3强大的拦截器

Struts2 的拦截器是一个Action级别的AOP，Struts2中的许多特性都是通过拦截器来实现的，例如异常处理，文件上传，验证等。拦截器是可配置与重用的，可以将一些通用的功能如：登录验证，权限验证等置于拦截器中以完成一些Java Web项目中比较通用的功能。在我实现的的一Web项目中，就是使用Struts2的拦截器来完成了系统中的权限验证功能。

4易于测试

Struts2的Action都是简单的POJO，这样可以方便的对Struts2的Action编写测试用例，大大方便了5Java Web项目的测试。

易于扩展的插件机制在Struts2添加扩展是一件愉快而轻松的事情，只需要将所需要的Jar包放到WEB-INF/lib文件夹中，在struts.xml中作一些简单的设置就可以实现扩展。

6模块化管理

Struts2已经把模块化作为了体系架构中的基本思想，可以通过三种方法来将应用程序模块化：将配置信息拆分成多个文件把自包含的应用模块创建为插件创建新的框架特性，即将与特定应用无关的新功能组织成插件，以添加到多个应用中去。

7全局结果与声明式异常

为应用程序添加全局的Result，和在配置文件中对异常进行处理，这样当处理过程中出现指定异常时，可以跳转到特定页面。

他的如此之多的优点，是很多人比较的青睐，与spring ,Hibernate进行结合，组成了现在比较流行的ssh框架，当然每个公司都要自己的框架，也是ssh变异的产品。

## struts2有哪些优点？

1）在软件设计上Struts2的应用可以不依赖于Servlet API和struts API。 Struts2的这种设计属于无侵入式设计；

2）拦截器，实现如参数拦截注入等功能；

3）类型转换器，可以把特殊的请求参数转换成需要的类型；

4）多种表现层技术，如：JSP、freeMarker、Velocity等；

5）Struts2的输入校验可以对指定某个方法进行校验；

6）提供了全局范围、包范围和Action范围的国际化资源文件管理实现

## struts2是如何启动的？

struts2框架是通过Filter启动的，即StrutsPrepareAndExecuteFilter，此过滤器为struts2的核心过滤器；

StrutsPrepareAndExecuteFilter的init()方法中将会读取类路径下默认的配置文件struts.xml完成初始化操作。struts2读取到struts.xml的内容后，是将内容封装进javabean对象然后存放在内存中，以后用户的每次请求处理将使用内存中的数据，而不是每次请求都读取struts.xml文件。

## struts2框架的核心控制器是什么？它有什么作用？

1）Struts2框架的核心控制器是StrutsPrepareAndExecuteFilter。

2）作用：

负责拦截由<url-pattern>/*</url-pattern>指定的所有用户请求，当用户请求到达时，该Filter会过滤用户的请求。默认情况下，如果用户请求的路径

不带后缀或者后缀以.action结尾，这时请求将被转入struts2框架处理，否则struts2框架将略过该请求的处理。

可以通过常量"struts.action.extension"修改action的后缀，如：

<constant name="struts.action.extension" value="do"/>

如果用户需要指定多个请求后缀，则多个后缀之间以英文逗号（,）隔开。

<constant name="struts.action.extension" value="do,go"/>

## struts2配置文件的加载顺序？

struts.xml ——> struts.properties

常量可以在struts.xml或struts.properties中配置，如果在多个文件中配置了同一个常量，则后一个文件中配置的常量值会覆盖前面文件中配置的常量值.

struts.xml文件的作用：通知Struts2框架加载对应的Action资源

## struts2常量的修改方式？

常量可以在struts.xml或struts.properties中配置，两种配置方式如下：

1）在struts.xml文件中配置常量

<constant name="struts.action.extension" value="do"/>

2）在struts.properties中配置常量（struts.properties文件放置在src下）：

struts.action.extension=do

## struts2如何访问HttpServletRequest、HttpSession、ServletContext三个域对象？

方案一：

HttpServletRequest request =ServletActionContext.getRequest();

HttpServletResponse response =ServletActionContext.getResponse();

HttpSession session= request.getSession();

ServletContext servletContext=ServletActionContext.getServletContext();

方案二：

类 implements ServletRequestAware,ServletResponseAware，SessionAware，ServletContextAware

注意：框架自动传入对应的域对象

## struts2是如何管理action的？这种管理方式有什么好处？

struts2框架中使用包来管理Action，包的作用和java中的类包是非常类似的。

主要用于管理一组业务功能相关的action。在实际应用中，我们应该把一组业务功能相关的Action放在同一个包下。

## struts2中的默认包struts-default有什么作用？

1）struts-default包是由struts内置的，它定义了struts2内部的众多拦截器和Result类型，而Struts2很多核心的功能都是通过这些内置的拦截器实现，如：从请求

中

把请求参数封装到action、文件上传和数据验证等等都是通过拦截器实现的。当包继承了struts-default包才能使用struts2为我们提供的这些功能。

2）struts-default包是在struts-default.xml中定义，struts-default.xml也是Struts2默认配置文件。 Struts2每次都会自动加载 struts-default.xml文件。

3）通常每个包都应该继承struts-default包。

## struts2如何对指定的方法进行验证？

1）validate()方法会校验action中所有与execute方法签名相同的方法；

2）要校验指定的方法通过重写validateXxx()方法实现， validateXxx()只会校验action中方法名为Xxx的方法。其中Xxx的第一个字母要大写；

3）当某个数据校验失败时，调用addFieldError()方法往系统的fieldErrors添加校验失败信息（为了使用addFieldError()方法，action可以继承ActionSupport）， 如果系统 的fieldErrors包含失败信息，struts2会将请求转发到名为input的result；

4）在input视图中可以通过<s:fielderror/>显示失败信息。

5）先执行validateXxxx()->validate()->如果出错了，会转发<result name="input"/>所指定的页面，如果不出错，会直接进行Action::execute()方法

## struts2默认能解决get和post提交方式的乱码问题吗？

不能。struts.i18n.encoding=UTF-8属性值只能解析POST提交下的乱码问题。

## 请你写出struts2中至少5个的默认拦截器？

fileUpload　　提供文件上传功能

i18n　　记录用户选择的locale

cookies　　使用配置的name,value来是指cookies

checkbox　　添加了checkbox自动处理代码，将没有选中的checkbox的内容设定为false，而html默认情况下不提交没有选中的checkbox。

chain　　让前一个Action的属性可以被后一个Action访问，现在和chain类型的result（）结合使用。

alias　　在不同请求之间将请求参数在不同名字件转换，请求内容不变

## 值栈ValueStack的原理与生命周期？

1）ValueStack贯穿整个 Action 的生命周期，保存在request域中，所以ValueStack和request的生命周期一样。当Struts2接受一个请求时，会迅速创建ActionContext，

ValueStack，action。然后把action存放进ValueStack，所以action的实例变量可以被OGNL访问。 请求来的时候，action、ValueStack的生命开始，请求结束，action、 ValueStack的生命结束；

2）action是多例的，和Servlet不一样，Servelt是单例的；

3）每个action的都有一个对应的值栈，值栈存放的数据类型是该action的实例，以及该action中的实例变量，Action对象默认保存在栈顶；

4）ValueStack本质上就是一个ArrayList；

5）关于ContextMap，Struts 会把下面这些映射压入 ContextMap 中：

parameters ： 该 Map 中包含当前请求的请求参数

request ： 该 Map 中包含当前 request 对象中的所有属性 session ：该 Map 中包含当前 session 对象中的所有属性

application :该 Map 中包含当前 application 对象中的所有属性

attr:该 Map 按如下顺序来检索某个属性: request, session, application

6）使用OGNL访问值栈的内容时，不需要#号，而访问request、session、application、attr时，需要加#号；

7）注意： Struts2中，OGNL表达式需要配合Struts标签才可以使用。
如：<s:property value="name"/>

8）在struts2配置文件中引用ognl表达式 ,引用值栈的值 ，此时使用的"$"，而不是#或者%;

## ActionContext、ServletContext、pageContext的区别？

1）ActionContext是当前的Action的上下文环境，通过ActionContext可以获取到request、session、ServletContext等与Action有关的对象的引用；

2）ServletContext是域对象，一个web应用中只有一个ServletContext，生命周期伴随整个web应用；

3）pageContext是JSP中的最重要的一个内置对象，可以通过pageContext获取其他域对象的应用，同时它是一个域对象，作用范围只针对当前页面，当前页面结束时，pageContext销毁，

生命周期是JSP四个域对象中最小的。

## result的type属性中有哪几种结果类型？

一共10种：

dispatcher

struts默认的结果类型，把控制权转发给应用程序里的某个资源不能把控制权转发给一个外部资源，若需要把控制权重定向到一个外部资源，应该使用

redirect结果类型

redirect　把响应重定向到另一个资源（包括一个外部资源）

redirectAction　　把响应重定向到另一个 Action

freemarker、velocity、chain、httpheader、xslt、plainText、stream

## 拦截器的生命周期与工作过程？

1）每个拦截器都是实现了Interceptor接口的 Java 类；

2）init(): 该方法将在拦截器被创建后立即被调用，它在拦截器的生命周期内只被调用一次．可以在该方法中对相关资源进行必要的初始化；

3）intercept(ActionInvocation invocation): 每拦截一个动作请求，该方法就会被调用一次；

4）destroy: 该方法将在拦截器被销毁之前被调用，它在拦截器的生命周期内也只被调用一次；

5）struts2中有内置了18个拦截器。

## struts2如何完成文件的上传？

1、JSP页面：

1）JSP页面的上传文件的组件：<s: file name="upload" />，如果需要一次上传多个文件, 就必须使用多个 file 标签, 但它们的名字必须是相同的，即：

name="xxx"的值必须一样；

2）必须把表单的enctype属性设置为：multipart/form-data；

3）表单的方法必须为post，因为post提交的数据在消息体中，而无大小限制。

2、对应的action：

1）在 Action 中新添加 3 个和文件上传相关的属性；

2）如果是上传单个文件, uploadImage属性的类型就是 java.io.File, 它代表被上传的文件, 第二个和第三个属性的类型是 String, 它们分别代表上传文

件的文件名和文件类型，定义方式是分别是：

jsp页面file组件的名称+ContentType， jsp页面file组件的名称+FileName

3）如果上上传多个文件, 可以使用数组或 List

# Hibernate--HQL查询

星期四, 十二月 27, 2018 10:31 上午

Hibernate提供了强大的查询系统，使用Hibernate有多种查询方法可以选择：可以使用Hibernate的HQL查询，也可以使用条件查询，甚至可以使用原生的SQL查询语句。其中HQL查询时Hibernate配置的功能强大的查询语句。HQL是非常有意识的被设计为完全面向对象的查询，它可以理解如继承、多态 和关联之类的概念。

一、HQL查询

HQL的语法和SQL很相似，但是HQL是一种面向对象的查询语句，它的操作对象是类、实例、属性等，而SQL的操作对象 是数据表、列等数据库对象。

由于HQL是完全面向对象的查询语句，因此可以支持继承、多态等特性。

HQL查询依赖于Query类，每一个Query实例对应一个查询对象，它的执行是通过Session的createQuery()方法来获得的。

执行HQL查询的步骤：

1、获得Hibernate Session对象

2、编写HQL语句

3、调用Session的createQuery方法创建查询对象

4、如果HQL语句包含参数，则调用Query的setXxx方法为参数赋值

5、调用Query对象的list等方法返回查询结果。

实例：

```
1 private void query(){
 2         Session session = HibernateUtil.getSession();
 3         Transaction tx = session.beginTransaction();
 4         //以HQL语句创建Query对象，执行setString方法为HQL语句的参数赋值
 5         //Query调用list方法访问查询的全部实例
 6         List list = session.createQuery("select distinct p from Person p
where p.name=:name")
 7                     .setString("name", "chenssy").list();
 8
 9         //遍历查询结果
10         for (Iterator iterator = list.iterator();iterator.hasNext();) {
11             Person p = (Person) iterator.next();
12             System.out.println("id="+p.getId()+",age="+p.getAge());
13         }
```

```
14          session.close();
15      }
```

上面的程序先编写HQL语句后，使用Session的createQuery(hql)方法创建一个Query，Query对象使用setXxx方法为HQL语句的参数赋值，最后调用list()方法返回查询的全部结果。

在这里Query对象可以连续多次调用setXxx方法为HQL参数赋值。这是因为Hibernate Query的setXxx方法的返回值为Query本身，因此程序创建Query后，可以直接多次调用setXxx方法为HQL语句的参数赋值。

Query对象还包含如下两个方法：

setFirstResult(int firstResult)：设置返回的结果集从第几条记录开始

setMaxResult(int maxResult)：设置本次查询返回的结果数目

这两个方法用于对HQL查询实现分页控制

二、HQL查询的from子句

Hibernate中最简单的查询语句的形式如下：

```
1  from Person
```

From关键字后面紧跟持久化类的类名。

大多数情况下，你需要指定一个别名，原因是你可能需要在查询语句的其它部分引用到Person

```
1 from Person as p
```

子句中可以同时出现多个类，其查询结果是产生一个笛卡儿积或产生跨表的连接

```
1 from Person  as p ,MyEvent as e
```

三、关联与连接

当程序需要从多个数据表中获取数据时，Hibernate使用关联映射来处理底层数据表之间的连接，一旦我们提供了正确的关联映射后，当程序通过Hibernate进行持久化访问时，将可利用Hibernate的关联来进行连接。

HQL支持两种关联join的形式：implicit(隐式) 与explicit（显式）。

显式form子句中明确给出了join关键字，而隐式使用英文点号(.)来连接关联实体。

受支持的连接类型是从ANSI SQL中借鉴来的。

inner join（内连接）

left outer join（左外连接）

right outer join（右外连接）

full join (全连接，并不常用)

使用显式连接，可以通过with关键字来提供额外的join条件

```
1 From Person as p inner join p.myEvent e with p.id=e.id
```

从上面可以看出with关键字的作用等同于SQL中on关键字的作用：用于指定连接条件。还有，一个"fetch"连接允许仅仅使用一个选择语句就将相关联的对象或一组值的集合随着他们的父对象的初始化而被初始化，这种方法在使用到集合的情况下尤其有用，对于关联和集合来说，它有效的代替了映射文件中的外联接与延迟声明（lazy declarations）。

对于隐式连接和显示连接有如下两个区别：

1、显式连接底层将转换成SQL99的交叉连接，显式连接底层将转换成SQL99的inner join、left join、right join等连接。

2、隐式连接和显式连接查询后返回的结果不同。使用隐式连接查询返回的结果是多个被查询实体组成的集合。使用显式连接的结果分为两种：如果HQL语句中省略select关键字时，返回的结果也是集合，但集合元素是被查询持久化对象、所有被关联的持久化对象所组成的数组。如果没有省略select关键字，返回的结果同样是集合，集合中的元素是跟在select关键字后的持久化对象组成的数组。

```
1 Form Person as p inner join p.myEvent as e with p.id=e.id
//...........1
2
3 Select p from Person as p inner join p.myEvent as e with p.id=e.id
//.........2
```

.....1中返回的结构是有Person实体和MyEvent实体组成的数组集合。而.....2 返回的结果是只有Person组成的集合。

对于有集合属性的。Hibernate默认采用延迟加载策略。如对于持久化类Person，有集合属性myEvent。加载Person实例时，默认是不加载myEvent的，如果session被关闭了，Person实例将会无法访问到关联的myEvent属性的。为了解决这个问题，可以再Hibernate映射文件中配置指定:lazy="false"来关闭延迟加载。或者使用join fetch:

```
1 From Person as p join fetch p.myEvent
```

一个fetch连接通常不需要被指定别名, 因为相关联的对象不应当被用在 where 子句 (或其它任何子句)中。同时，相关联的对象并不在查询的结果中直接返回，但可以通过他们的父对象来访问到他们。

使用fetch关键字时需要注意以下几个问题：

1、fetch不应该与setMaxResults()和setFirstResults()共用，

2、fetch不能与独立的with条件一起使用

3、如果在一次查询中fetch多个集合，可以查询返回的笛卡尔积

4、full join fetch和right join fetch没有任何意义

5、对于bag映射而言，同时join fetch多个结合时可能会出现非预期结果

四、select子句

Select子句用于选择将哪些对象与属性返回到查询结果集中。当然select选择的属性必须是from后持久化类包含的属性。

```
1 select p.name from Person as p
```

select查询语句可以返回值为任何类型的属性，包括返回类型为某种组件(Component)的属性：

```
1 select p.myEvent.title from Person as p
```

select查询语句可以返回多个对象和（或）属性，存放在 Object[]队列中：

```
1 select p,e from Person as p inner join p.myEvent as e with p.id=e.id
```

Select查询语句也支持将选择出的属性存放到一个List对象中

```
1 select new List(p.name,p.age) from Person as p
```

Select查询语句还可以将选择出的属性直接封装成一个对象。

```
1 select new ClassTest(p.id,p.name,p.age) from Person as p
```

但前提是ClassTest支持p.id，p.name，p.age的构造前，假如p.id的数据类型是int，p.name的数据类型是String，p.age的数据类型是int，那么ClassTest必须有如下构造器：

```
1 ClassTest(int id,String name,int age)
```

Select还支持给选定的表达式名别名：

```
1 Select p.name as personname from Person as p
```

种做法在与子句select new map一起使用时最有用：

```
1 Select new map(p.name as personname) from Person as p
```

五、聚集函数

受支持的聚集函数如下：

avg(...), sum(...), min(...), max(...)

count(*)

count(...), count(distinct ...), count(all...)

Select子句也支持使用distinct和all关键字，此时的效果与SQL中的效果相同。

六、多态查询

Hibernate可以理解多态查询，from后跟持久化类名，不仅会查出该持久化类的全部实例还好查询出该类的全部子类的全部实例。

```
from Person as p
```

该查询语句不仅会查询出Person的全部实例，还会查询出Person的子类：Teacher的全部属性。

Hibernate 可以在from子句中指定任何 Java 类或接口. 查询会返回继承了该类的所有持久化子类 的实例或返回声明了该接口的所有持久化类的实例。下面的查询

语句返回所有的被持久化的对象：

```
1 From java.lang.Object o
```

七、Where子句

where子句允许你将返回的实例列表的范围缩小.如果没有指定别名，你可以使用属性名来直接引用属性:

```
1 From Person  where age < 40
```

如果指派了别名，需要使用完整的属性名:

```
1   From Person as p where p.age<40
```

复合属性表达式增强了where子句的功能：

?

```
1       From person p where p.myEvent.title like "%你是"
```

该查询语句被翻译为一个含有内连接的SQL查询语句。

只要没有出现集合属性，HQL语句可使用点号来隐式连接多个数据表：

```
1    From person p where p.myEvent.event.name like "%hhh";
```

上的语句SQL需要连接三张表。

=运算符不仅可以被用来比较属性的值，也可以用来比较实例：

```
1 from Cat cat, Cat rival where cat.mate = rival.mate
2 from Cat cat, Cat rival where cat.mate = rival.mate
```

特殊属性（小写）id可以用来表示一个对象的唯一的标识符

```
1 From Person p where p.id=1
2
3 From Person p where p.myEvent.id=1
```

第二个查询是有效的。此时不需要进行表连接，而完全使用面向对象的方式查询！

在进行多态持久化的情况下，class关键字用来存取一个实例的鉴别之。嵌入where自己中的java类名将会被作为该类的鉴别值。

```
1 from Person as p where p.class = Teacher
```

在执行多态的时候，默认会选出Person及其所有子类的实例，但是上面的HQL语句，将只会选出Teacher类的实例。

当where子句的运算符只支持基本类型或者字符串时，where子句中的属性表达式必须以基本类型或者字符串结尾，不要使用组件类型属性结尾。

八、order by 子句

查询返回的列表(list)可以按照一个返回的类或组件（components)中的任何属性（property）进行排序：

```
1 From Person as p ordery by p.id
```

可选的asc或desc关键字指明了按照升序或降序进行排序.

九、group by子句

一个返回聚集值(aggregate values)的查询可以按照一个返回的类或组件（components)中的任何属性（property）进行分组：

```
1 Select p.id,p.name from Person p group by p.age
```

可以使用having子句对分组进行过滤

```
1 Select p.id,p.name from Person p group by p.age having p.age between 10
and 40
```

注意:group by子句与 order by子句中都不能包含算术表达式。也要注意Hibernate目前不会扩展group的实体,因此你不能写group by cat,除非cat的所有属性都不是聚集的。你必须明确的列出所有的非聚集属性。

十、子查询

对于支持子查询的数据库，Hibernate支持在查询中使用子查询。一个子查询必须被圆括号包围起来（经常是SQL聚集函数的圆括号）。 甚至相互关联的子查询（引用到外部查询中的别名的子查询）也是允许的。

```
1 From Person p where p.age > (select avg(p1.age) from Person p1 )
```

与SQL子查询相同，如果子查询是多行结果集，则应该使用多行运算符。同时HQL子查询只可以在select子句或者where子句中出现。

如果在select子查询或的列表中包含多项，则在HQL中需要使用一个元组构造符：

```
1 From Person as p where (p.name,p.age) in (select s.name,s,age from
Student as s)
```

十一、命名查询

HQL查询还支持将查询所用的HQL语句放在配置文件中，而不是程序代码中。通过这种方式，可以大大提高程序的解耦。

在Hibernate映射文件中使用<query.../>元素来定义命名查询。使用<query.../>元素是需要指定一个name属性，该属性指定该命名查询的名字。

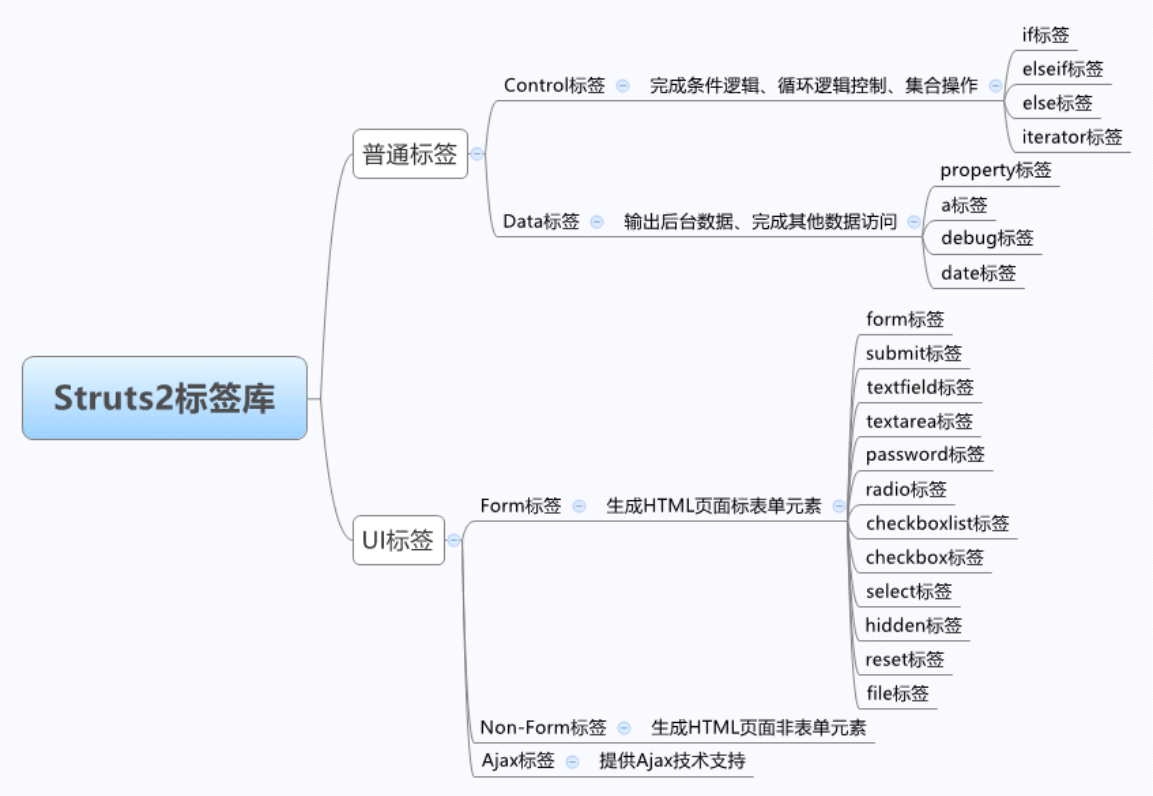
```
1         <query name="namedQuery">
2             <!-- 此处确定命名查询的HQL语句 -->
3             from Person as p
4         </query>
```

Session提供一个getNamedQuery(String name)方法用于获取指定命名HQL查询并且创建Query对象。

```
1     public void namedQuery(){
 2         Session session = HibernateUtil.getSession();
 3         Transaction tx = session.beginTransaction();
 4         //执行命名查询
 5         List list = session.getNamedQuery("namedQuery").list();
 6         for (Iterator iterator = list.iterator();iterator.hasNext();) {
 7             Person p = (Person) iterator.next();
 8            System.out.println(p.getName());
 9        }
10         tx.commit();
11         session.close();
12     }
```

# structs标签库

2018年12月25日　　8:55




对于一个MVC框架而言，重点是实现两部分：业务逻辑控制器部分和视图页面部分。Struts2作为一个优秀的MVC框架，也把重点放在了这两部分上。控制器主要由Action来提供支持，而视图则是由大量的标签来提供支持。

Struts2标签库是一个比较完善，而且功能强大的标签库，它将所有标签都统一到一个标签库中，从而简化了标签的使用，它还提供主题和模板的支持，极大地简化了视图页面代码的编写，同时它还提供对ajax的支持，大大的丰富了视图的表现效果。与JSTL(JSP Standard Library，JSP 标准标签库)相比，Struts2标签库更加易用和强大。

使用标签，需要引入Struts2核心jar包，在jsp头部，加入<%@ taglib uri="/struts-tags" prefix="s" %>

Struts2 默认提供了4 种主题，分别为 simple、xhtml、css_xhtml和Ajax。

- simple主题：这是最简单的主题，使用该主题时，每个UI标签只生成最基本的HTML元素，没有任何附加功能。
- xhtml主题：这是Struts2的默认主题，它对simple主题进行了扩展，提供了布局功能、Label显示名称、以及与验证框架和 国际化框架的集成。
- css_xhtml：该主题是xhtml的扩展，在对xhtml的基础之上添加对CSS的支持和控制。
- Ajax：继承自对xhtml，提供 Ajax支持。

```
<constant name="struts.ui.theme" value="simple"/>
```

控制标签

if-elseif-else

```
<%--1、if elseif else的使用 --%>
<% //存入请求域中一个学生的成绩
   request.setAttribute("score", 89);
%>
<!-- 判断学生成绩，输出成绩所对应的ABCD -->
<s:if test="#request.score>90">
   优秀
</s:if>
<s:elseif test="#request.score>80">
   良好
</s:elseif>
<s:else>
   一般
</s:else>
```

iterator

```
<s:iterator value="customers">
<%--s:iterator是struts2的一个迭代标签，它的value属性取值是一个OGNL表达式
var属性：它的取值就是一个普通的字符串.
用了var：把每次遍历的对象作为value，把var的值作为key，存入ContextMap中
没用var：把每次遍历的对象压入栈顶，再下次遍历之前弹栈(从栈顶移走)。
begin：开始遍历的索引
end：遍历的结束索引
step：遍历的步长。
status：计数器对象
   count 已经遍历的集合元素个数
   index 当前遍历元素的索引值
   odd  是否奇数行
   even 是否偶数行
   first 是否第一行
   last  是否最后一行
--%>
    <TR style="FONT-WEIGHT: normal; FONT-STYLE: normal; BACKGROUND-COLOR: white; TEXT-
DECORATION: none">
        <TD><s:property value="custName"/></TD>
        <TD><s:property value="custLevel"/></TD>
        <TD><s:property value="custSource"/></TD>
        <TD><s:property value="custIndustry"/</TD>
        <TD><s:property value="custAddress"/></TD>
        <TD><s:property value="custPhone"/></TD>
    </TR>
</s:iterator>
```

数据标签

property

- id：可选属性，指定该元素的标识。
- default：可选属性，如果要输出的属性值为null，则显示default属性的指定值。
- escape：可选属性，指定是否忽略HTML代码。
- value：可选属性，指定需要输出的属性值，如果没有指定该属性，则默认输出ValueStack栈顶的值

```
<%-- 输出值栈中的值 --%>
<s:property value="custName"/>
```

```
<%--使用struts2的超链接标签发送请求：
```

```
  <s:a>连接内容</s:a>
  属性：
  href：指定url路径
  action：请求的动作名称（类名）
  namespace：动作名称所在的名称空间
  id：指定id
  method：指定Action调用方法
--%>
<s:a action="addUIUser" namespace="/user">添加用户</s:a>
<s:a action="editUIUser" namespace="/user"> 编辑用户
   <s:param name="userid" value="%{'1'}"></s:param>
</s:a>
<a href="${pageContext.request.contextPath}/user/editUIUser.action?userid=1">原始超链接标签-编辑用户</a>
```

debug

debug标签用于调试Struts2，使用它会在页面中生成一个debug标签，点击后会显示服务器各种对象信息，包括值栈、ContextMap等。

```
<s:debug/>
```

date

```
<%--s:date标签的使用：
  它是用于格式化输出日期
  name属性：取值是一个ognl表达式，表示要格式化的日期对象
  format属性：指定格式
  var属性：取值是一个普通的字符串。
      把格式化好的日期字符串作为value，把var的取值作为key。存入contextMap中
--%>
<% request.setAttribute("myDate",new Date());
 /*  SimpleDateFormat format = new SimpleDateFormat("yyyy年MM月dd日");
  String date = format.format(new Date());
  out.println(date); */
%>
${requestScope.myDate}<br/>
<s:property value="#request.myDate"/>
<s:date name="#request.myDate" format="yyyy年MM月dd日" var="sdate"/>
<br>
格式化后的日期：<s:property value="#sdate"/>
<br/>
${sdate}
```

url

```
<%-- url标签
  作用：用于存放一个路径
  属性：
    action：动作名称
    namespace：名称空间
    var：取值是一个普通字符串。他会把action和namespace组成一个url作为value，把var的取值作为一个key，存入contextMap中
--%>
<s:url action="addUIUser" namespace="/user" var="myurl"/>
<a href="<s:property value='#myurl'/>">添加用户——url</a>
```

## 表单标签

### 表单标签的通用属性

| 属性名 | 主题 | 数据类型 | 说明 |
| --- | --- | --- | --- |
| title | simple | String | 设置表单元素的title属性 |
| disabled | simple | String | 设置表单元素是否可用 |
| label | xhtml | String | 设置表单元素的label属性 |
| labelPosition | xhtml | String | 设置label元素的显示位置，可选值 ： top 和 left(默认) |
| name | simple | String | 设置表单元素的name属性，与Action中的属性名对应 |
| value | simple | String | 设置表单元素的值 |
| cssClass | simple | String | 设置表单元素的class |
| cssStyle | simple | String | 设置表单元素的style属性 |
| required | xhtml | Boolean | 设置表单元素为必填项 |
| requiredposition | xhtml | String | 设置必填标记(默认为*)相对于label元素的位置，可选值：left 和right(默认) |
| tabindex | simple | String | 设置表单元素的tabindex属性 |

### <form>标签的常用属性及描述

| 属性名 | 是否必填 | 类型 | 说明 |
| --- | --- | --- | --- |
| action | 否 | String | 指定提交时对应的action，不需要action后缀 |
| enctype | 否 | String | HTML表单enctype属性 |
| method | 否 | String | HTML表单method属性 |
| namespace | 否 | String | 所提交action的命名空间 |

### <s:submit>标签的常用属性

| 属性名 | 是否必填 | 类型 | 说明 |
| --- | --- | --- | --- |
| action | 否 | String | 指定提交时对应的action |
| method | 否 | String | 指定action中调用的方法 |

### <s:password>标签的常用属性说明

| 属性名 | 说明 |
| --- | --- |
| Name | 用于指定密码输入框的名称 |
| Size | 用于指定密码输入框的显示宽度，以字符数为单位 |
| MaxLength | 用于限定密码输入框的最大输入字符串个数 |
| showPassword | 是否显示初始值，即使显示也仍为密文显示，用掩码代替 |

### <s:radio>标签的属性及说明

| 属性名 | 是否必填 | 类型 | 说明 |
| --- | --- | --- | --- |
| List | 是 | Cellection、Map、Enmumeration、Iterator，array | 用于生成单选框中的集合 |
| listKey | 否 | String | 指定集合对象中的哪个属性作为选项的value |
| listValue | 否 | String | 指定集合对象中的哪个属性作为 |

| | | | |
|---|---|---|---|
| | | | 选项的内容 |

<s:checkboxlist>标签的常用属性及说明

| 属性名 | 是否必填 | 类型 | 说明 |
|---|---|---|---|
| name | 否 | String | 指定该元素的name |
| list | 是 | Cellection、Map、Enmumeration、Iterator，array | 用于生成多选框的集合 |
| listKey | 否 | String | 生成checkbox的value属性 |
| listValue | 否 | String | 生成checkbox后面显示的文字 |

select标签的常用属性及说明

| 属性名 | 是否必填 | 类型 | 说明 |
|---|---|---|---|
| list | 是 | Cellection、Map、Enmumeration、Iterator，array | 用于生成下拉框的集合 |
| listKey | 否 | String | 生成选项的value属性 |
| listValue | 否 | Strmg | 生成选项的显示文字 |
| headerKey | 否 | String | 在所有的选项前再加额外的一个选项作为其标题的value值 |
| headerValue | 否 | String | 显示在页面中header选项的内容 |
| Multiple | 否 | Boolean | 指定是否多选，默认为 false |
| emptyOption | 否 | Boolean | 是否在标题和真实的选项之间加一个空选项 |
| size | 否 | Int | 下拉框的高度，即最多可以同时显示多少个选项 |

example

```
<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
<%@ taglib uri="/struts-tags" prefix="s" %>
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>注册界面</title>
  </head>
  <body>
    <s:form action="register" namespace="">
      <s:textfield name="username" label="用户名" key="Switch"/>
      <s:password name="password" label="密码"/>
      <s:radio name="gender" list="#{'0':'男','1':'女'}" label="性别" value="0" />
      <s:textfield name="age" label="年龄"/>
      <s:select name="city" list="#{'bj':'北京','sh':'上海','gz':'广州','sz':'深圳'}" label="城市"
headerKey="-1" headerValue="---请选择城市---" emptyOption="true"/>
      <s:checkboxlist name="hibbies" list="#{'code':'写代码','algorithm':'算法','movie':'电影'}"
label="爱好"/>
      <s:checkbox name="married" label="是否已婚" value="true" labelposition="left"/>
      <s:textarea name="description" label="自我介绍" rows="5" cols="20"/>
      <s:file name="phone" label="头像"/>
      <s:submit value="提交"/>
      <s:reset value="重置"/>
```

```
    </s:form>
  </body>
</html>
```

# jQuery

2018年12月24日 14:52

jQuery安装的两种方法

1、本地下载安装

2、联网使用

| | |
|---|---|
| 本地安装 | <head> <script src="jquery-1.10.2.min.js"> </script> </head> |
| Staticfile CDN | https://cdn.staticfile.org/jquery/1.10.2/jquery.min.js |
| 百度CDN | https://apps.bdimg.com/libs/jquery/2.1.4/jquery.min.js |
| 又拍云CDN | https://upcdn.b0.upaiyun.com/libs/jquery/jquery-2.0.2.min.js |
| 新浪CDN | https://lib.sinaapp.com/js/jquery/2.0.2/jquery-2.0.2.min.js |
| Google CDN | https://ajax.googleapis.com/ajax/libs/jquery/1.10.2/jquery.min.js |

jQuery选择器

| | |
|---|---|
| ("p") | 元素选择器 |
| ("#test") | id选择器 |
| (".class") | .class选择器 |
| | |
| $("*") | 选取所有元素 |
| $(this) | 选取当前 HTML 元素 |
| $("p.intro") | 选取 class 为 intro 的 <p> 元素 |
| $("p:first") | 选取第一个 <p> 元素 |
| $("ul li:first") | 选取第一个 <ul> 元素的第一个 <li> 元素 |
| $("ul li:first-child") | 选取每个 <ul> 元素的第一个 <li> 元素 |
| $("[href]") | 选取带有 href 属性的元素 |
| $("a[target='_blank']") | 选取所有 target 属性值等于 "_blank" 的 <a> 元素 |
| $("a[target!='_blank']") | 选取所有 target 属性值不等于 "_blank" 的 <a> 元素 |
| $(":button") | 选取所有 type="button" 的 <input> 元素 和 <button> 元素 |
| $("tr:even") | 选取偶数位置的 <tr> 元素 |
| $("tr:odd") | 选取奇数位置的 <tr> 元素 |

jQuery事件

| 鼠标事件 | 键盘事件 | 表单事件 | 文档/窗口事件 |
|---|---|---|---|

| | | | |
|---|---|---|---|
| click | keypress | submit | load |
| dblclick | keydown | change | resize |
| mouseenter | keyup | focus | scroll |
| mouseleave | hover | blur | unload |

jQuery效果

| | |
|---|---|
| 隐藏/显示 | Hide/show |
| 来回切换显示和隐藏 | Toggle |
| 淡入淡出方法 | 淡入Fadeln()<br>淡出fadeout() |
| 来回切换淡入淡出 | fadeToggle() |
| 渐变成给定的不透明度 | fadeTO() |
| 向下滑动元素 | slideDown() |
| 向上滑动元素 | slideUp() |
| 可以来回切换 | slideToggle() |
| 创建自定义动画 | animate（） |
| | 可以操作多个属性变化 |
| | 可以使用相对值 |
| | 可以使用预定义的值 |
| | 可以使用队列功能（会逐一进行调用） |
| 停止动画 | stop（）（在动画完成之前停止） |
| Callback()回调函数 | 可以在动画100%完成之后再执行 |
| jQuery链（chaining） | 可以把动作/方法连接在一起（即允许在相同的元素上使用一条语句运行多个jQuery方法） |

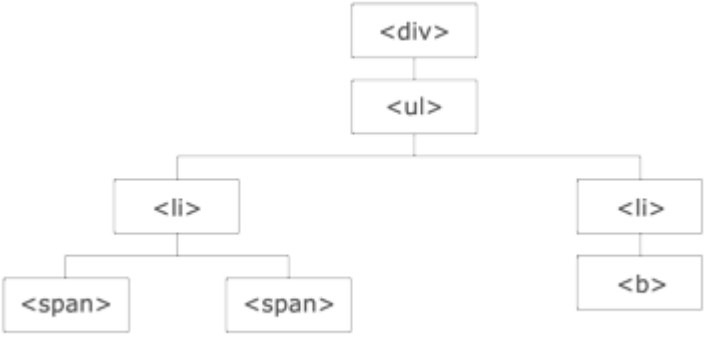
jQuery HTML

| | |
|---|---|
| DOM操作 | DOM=Document Object Model 文件对象模型<br>定义访问HTML和XML文档的标准<br>text()--设置或返回所选元素的文本内容<br>html()--设置或返回所选元素的内容（包括html标记）<br>value（）--设置或返回表单字段的值<br>attr()--获取属性值 |
| 添加元素 | 为元素添加内容或添加新元素<br>在选择的元素中<br>append()在被选元素的结尾插入内容<br>prepend()在被选元素的开头插入内容<br>在选择元素外<br>after()在被选元素之后插入内容<br>before（）在被选元素之前插入内容 |
| 删除元素 | remove()删除被选元素（及其子元素）可以 |

| | |
|---|---|
| | 选择过滤元素删除<br>empty()从被选元素中删除子元素 |
| 设置CSS | addClass()向被选元素中添加一个或多个类<br>removeClass（）从被选元素中删除一个或多个类<br>toggleClass()对被选元素进行添加/删除类的切换操作<br>css()设置或返回样式属性（一个或多个） |
| jQuery处理元素和浏览器的尺寸 | width()设置和返回元素的宽度（不包括内边距、边框或外边距）<br>height()设置和返回元素的高度<br>innerWidth()返回元素的宽度（包括内边距）<br>innerHeight()返回元素的高度<br>outerWidth()返回元素的宽度（包括内边距和边框）<br>outerHeight()返回元素的高度 |
| | |

jQuery遍历

根据其相对于其他元素的关系来查找或选取HTML元素，以某项选择开始，并沿着这个选择移动，直至抵达期望的元素




对DOM进行遍历

# Scrapy

2018年12月27日　　16:51

安装 pip install scrapy

有3个最低依赖包

**scrapy依赖的一些包：**

lxml：一种高效的XML和HTML解析器，

PARSEL：一个HTML / XML数据提取库，基于上面的lxml，

w3lib：一种处理URL和网页编码多功能辅助

twisted,：一个异步网络框架

cryptography and pyOpenSSL，处理各种网络级安全需求

**以上包需要的最低版本：**

Twisted 14.0

lxml 3.4

pyOpenSSL 0.14

http://www.scrapyd.cn/doc/186.html

# Echarts

2018年12月28日　　9:10

官方文档：http://echarts.baidu.com/tutorial.html



数据的异步加载和更新
使用dataset进行数据的管理
对于图表进行交互---可以拖拽图表元素、可以局部放大缩小
移动端适应--整体大小的改变
图表事件--点击、放置、双击、下钻
支持富文本标签--丰富显示项--文字样式的丰富
渲染器的选择--Canvas--SVG
三维立体可视化
支持微信小程序

前端技术中好玩而且比较实用的我想应该要数前端的数据可视化这一方面，目前市面上的数据可视化的框架琳琅满目，例如：D3.js、hightcharts.js、echarts.js……。由于公司对这个项目的需求是1、开发时间短，所以也就限制了D3.js的使用。2、要尽量的减少开发的成本，所以也就不能使用hightcharts.js(hightcharts是一款个人免费，商业付费的框架)。所以在再三的比对之下最终选择了echarts.js

## 二、echarts.js的优势与总体情况

echarts.js作为国内的IT三巨头之一的百度的推出一款相对较为成功的开源项目，总体上来说有这样的一些优点

**1、echarts.js容易使用**

echarts.js的官方文档比较详细，而且官网中提供大量的使用示例供大家使用

**2、echarts.js支持按需求打包**

echarts.js官网提供了在线构建的工具，可以在线构建项目时，选择项目所需要使用到的模块，从而达到减小JS文件的体积

**3、echarts.js开源**

**4、支持中国地图功能**

这个在其他的一些框架中是没有的，所以为这个功能点个赞

但是echarts.js也存在着一些不好的地方，比如说：

**1、echarts.js的体积较大**

一个基础的echarts.js都要400K左右，相对于D3.js和hightcharts.js来说都是比较大的

**2、echarts.js的可定制性差**

说到echarts.js的定制性差，其实不止是包括echarts.js，hightcharts.js也是如此，因为这一类型的数据可视化框架主要是高度的进行分装，所以你在使用的时候只需要设置一下配置就可以了，但是如果是出现了要绘制配置中不支持的图表怎么办，那么你就只能放弃，尝试着使用其他的框架了

**总的来说:**从大的方向上面来看，echarts.js还是值得去了解学习使用的，因为echarts.js得到了百度团队的重视，在git上面的更新也是比较的频繁，所以不会出现一些比较严重的bug之类的，最后这款框架一点就是框架的配置文件相当的详细，但是交互API文档虽然有说明，但是还是没有示例来举证，这个可能就是我认为的一个不足之处吧

## 三、echarts的应用

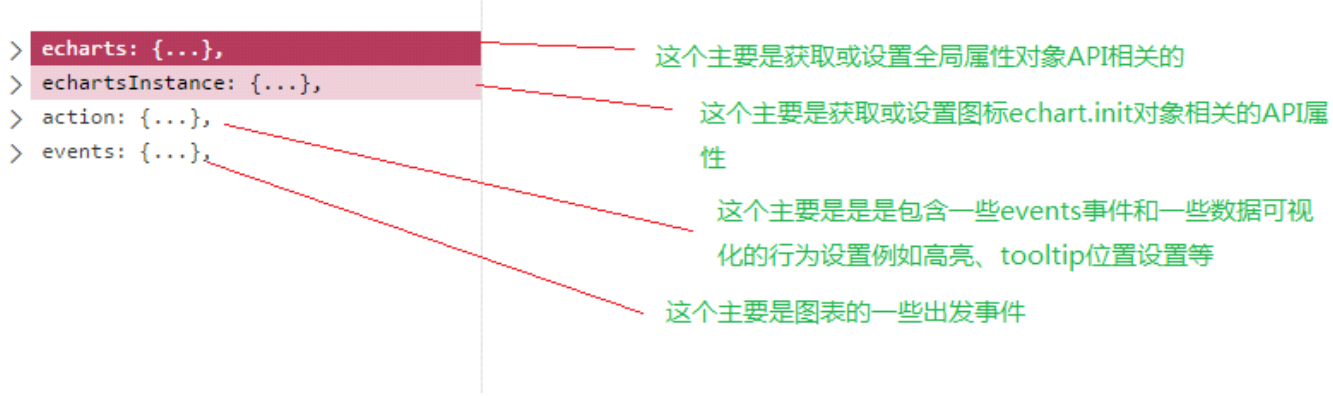
首先要说明一点是，echarts这个框架的配置内容很是多，所以不要尝试着把这个框架中的方法都给记住，这是不太可能的事。但是由于这个框架的配置文件参数比较多，所以我们就需要来学习一下echarts是怎样来对其进行分类的

1、首先echarts的图形化呈现主要是通过配置方法来实现的(setOption),然后是对图形标签进行初始化，最后把配置方法(setOption)赋值到初始化图形中，详细的配置文件请戳这里，这里我就来介绍一下关于配置文件的学习的经验之谈，比较常见的配置大致如下图：


上面用红色方框标出来的就是echarts的基础配置，也是我认为的学习echarts一定要掌握的配置，其他的一些配置比如什么时间轴.visualMap组件之类，我认为这些异曲同工，所以这一部分也就是只有在当你的业务需要使用的时候才加入，也就是说，这一部分的知识我认为到时候现炒现卖就可以了（更正：图标悬停的提示内容应该更正为鼠标悬停的提示内容），下面我就来讲解一下echarts.js的使用，首先我在官网中下载默认的精简版，下载地址如下：http://echarts.baidu.com/builder.html，直接下载即可（建议在开发期间使用源码版，方便调试）

### 3.4 echarts的API交互

首先我们来理清官方文档中的API的分类，大致的API可以分成这样的四类




这里我们就来解释一下echarts对象里面主要是包括一些销毁对象(dispose)，注册地图(registerMap)，初始化对象(echarts.init),关联对象(connect)，属于全局属性的设置，echartsInstance这个是包含echarts图中的某些属性的获取或者设置，获取宽高(getWidth、getHeight),获取配置(getOption)，设置配置(setOption)等操作action和events这两个操作在上图中已经很明白了，所以就不多做解释，具体的使用方法要与业务进行挂钩才有意义，所以在这里就不提供DEMO了，我相信大家看一下文档都能够看懂个究竟。

## 四、echarts常见问题解决

1、当X轴上面要渲染的数据太多的时候就会出现只渲染出来一部分，但是图表中的数据显示(比如说柱状图中的每个柱子)又会自动的进行宽度的缩放，所以会导致X轴与图中的信息不相匹配的问题，解决的方法是在X轴设置axisLabel :{ interval:0 }这个属性问题就可以

解决了，Y轴的使用方法相同

2、为了使echart图表随着浏览器的大小发生响应式变化，所以需要在设置配置之前添加 window.onresize = echart.resize;

## 五、总结

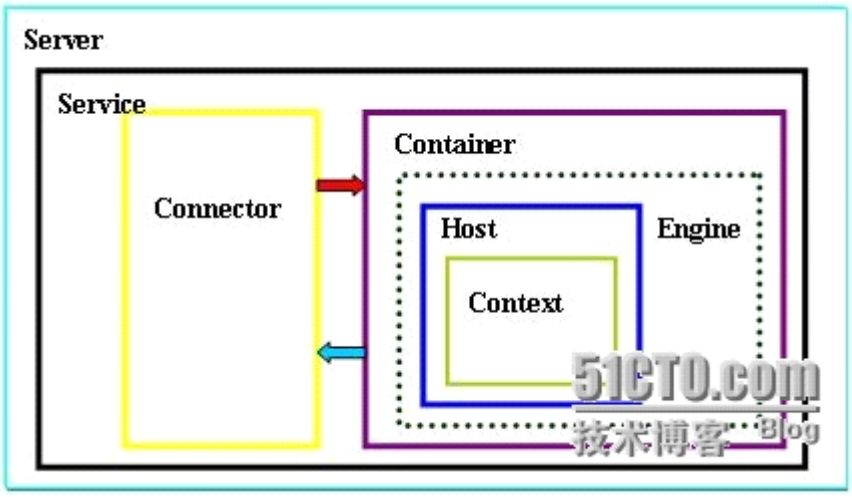
虽然echarts在同类型的数据可视化框架中还算是比较的简单易用的，但是在入手的时候可能也会有一些比较麻烦的地方困扰着你，比如文档这么多我怎么看，实例已经比较齐全了我还需要看API文档吗，这些问题我想在这里统一的说一下，其实echarts的学习无外乎就是先了解一下框架大致上的分类，然后是在将全部的API阅读以便，使对框架有一个全局的认识，然后在通过实践去深入学习，示例只不过是这一步的一个辅助而已，只有扎扎实实的通过学习API一步一个脚印去了解学习，才能做到在使用的时候心中有底。以上的文章是博主在学习和项目中使用echarts的一些体会，希望对大家有帮助

# Tomcat基础配置详解

星期五, 十二月 28, 2018　　3:12 下午

**Tomcat基础配置详解**

组件原理图如下：




任何tomcat实例就是一个server，而一个server内部要想能够解析jsp页面转义编译serlet程序，要靠其引擎来实现

而引擎才是真正意义上执行jsp代码的容器，都是tomcat用类来描述这些组件的

同时，为了接受用户的请求，需要基于connector组件，所谓监听的套接字的程序，能够接手用户的请求，被称为连接器

一个server内部可以完全运行N个引擎，无非就是运行多个虚拟机而已

**war包的概念**

放在网页目录可以直接访问，而部署的时候可以自动将其展开装载，而这是部署本身所完成的工作

在实现web站点的时候，在出程序之下有目录比如/web/htdoc，而在其目录下有一子目录/web/htdoc/bbs 而在其下有个论坛程序

这个目录程序跟其他程序可能不在同一组，而新加的子目录则需要重新部署，因为其属于独立的应用程序，因为每个独立程序都需要独立部署

对于tomcat来讲每一个host内部还有一子组件，叫做conntest,其作用是为了实现程序的独立部署

对于tomcat来讲要想记录日志的话，因此这些都需要使用特殊接口来实现，所以，任何一个需要操作硬件资源的进程，资源都有一个类在实现我们的需求，tomcat也一样。

**常见组件**

1.服务器（server）

实例，通常一个jvm只能包含一个实例，一般情况下，一个物理服务器可以启动多个jvm，从而启动多个实例，但一般不这么做

2.服务（service）

一个服务组件通常包含一个引擎和此引擎相关联的一个或多个链接服务器

3.连接器（connectors）

一个引擎能配置多个连接器 但是每个连接器的端口不能冲突

同时，tomcat也支持AJP JSERV和JK2连接器，实现让apache反向代理到后端服务器的非常高效的传输协议

**容器类组件**

4.引擎

可以自己接收用户的http请求，并构建响应报文，而且可以在内部处理java程序的整个套间

5.主机

6.上下文

**被嵌套类组件**

7.阀门

能够过滤也可以做访问控制

8.日志记录器

9.领域（Realm）

用来实现用户的认证和授权

对tomcat来讲，每种模型上的实现必须开发一种程序，才能完成相应组件的功能，而java中任何程序都是一个类

**JAVA常用类型文件**

EJB程序通常以.jar结尾

web程序通常打包为.war

资源适配器通常为.rar

企业级应用长须为.ear

web服务通常会打包为.ear或.war

## tomcat配置文件详解

tomcat的配置文件位于/path/to/tomcat/conf/目录下

[root@node1 conf]#cd /usr/local/tomcat/conf

[root@node1 conf]#cpserver.xml server.xml.bak

查看文件：

[root@node1 conf]#cat server.xml

Listener 为侦听器，通常实现tomcat内部进行通信的，可在各组件之间完成通信

<Serverport="8005" shutdown="SHUTDOWN">

```
  <Listener
className="org.apache.catalina.core.AprLifecycleListener"SSLEngine="o
n" />

  <Listener className="org.apache.catalina.core.JasperListener"/>

  <Listener
className="org.apache.catalina.core.JreMemoryLeakPreventionListener"/
>

  <Listener
className="org.apache.catalina.mbeans.GlobalResourcesLifecycleListene
r"/>

  <Listener
className="org.apache.catalina.core.ThreadLocalLeakPreventionListener"
/>
```

全局命名资源，方便全局引用，所以为其起完名称后可以随便调用的

```
  <GlobalNamingResources>

    <Resource name="UserDatabase"auth="Container"

            type="org.apache.catalina.UserDatabase"

             description="User databasethat can be updated and saved"

            factory="org.apache.catalina.users.MemoryUserDatabaseFactory
```

"

pathname="conf/tomcat-users.xml" /> #其调用tomcat-user.xml配置文件进行用户认证

</GlobalNamingResources>

服务类组件配置信息，将连接器关联至引擎上

<Servicename="Catalina">

<Connector port="8080" protocol="HTTP/1.1" #所在监听端口，以及协议版本号

connectionTimeout="20000" #连接超时时间，单位毫秒

redirectPort="8443" /> #必要的时候可以做重定向，定义在8443

<Connector port="8443"protocol="HTTP/1.1" SSLEnabled="true" #端口监听在8443，协议http1.1

maxThreads="150" scheme="https" secure="true" #最大线程，协议版本，安全的

clientAuth="false" sslProtocol="TLS" /> #不验证客户端 ssl协议用的是tls

<Connector port="8009" protocol="AJP/1.3"redirectPort="8443" />

<Engine name="Catalina" defaultHost="localhost"> #引擎，名为catalina

<RealmclassName="org.apache.catalina.realm.LockOutRealm">

<RealmclassName="org.apache.catalina.realm.UserDatabaseRealm"

resourceName="UserDatabase"/>

</Realm>

<Host name="localhost" appBase="webapps" #应用程序存放的位置，相对路径

unpackWARs="true" autoDeploy="true"> #如果是war文件格式，是否解压，是否自动部署

#定义阀门，java中类的记录方式，当前所处域名反过来写的记录方式

<ValveclassName="org.apache.catalina.valves.AccessLogValve"directory="logs"

prefix="localhost_access_log." suffix=".txt" #日志的命名

suffix表示时间戳
pattern="%h %l %u %t "%r" %s %b" /> #访问日志的格式

</Host>
</Engine>
</Service>
</Server>

tomcat的运行方式

tomcat可以自我独立运行，因此可以直接监听在某个端口，从而接收用户的http请求，并构建响应报文

另外也可以做为容器，不接受用户的http请求只接受用户的对于某个jsp文件的请求

当用户第一次去请求tomcat的时候，会自动部署好一个程序的

来查看配置文件：

明确说明了主机名默认为localhost，而unpackWARs的值为真，意为是可以自动部署的

<Hostname="localhost" appBase="webapps"

unpackWARs="true" autoDeploy="true">

配置Tomcat

·定义管理页面用户名及密码

用户名及密码是通过tomcat-users.xml配置文件进行调用的，所以我们只需要配置tomcat-users.xml即可

[root@node1 conf]#vim tomcat-users.xml

加入以下参数信息

<rolerolename="manager-gui"/> #定义manager组

<role rolename="admin-gui"/> #定义admin组

<user username="tomcat"password="tomcat" roles="manager-gui,admin-gui"/> #定义用户名及密码都为tomcat，并将tomcat用户加入至manager、admin组中

保存退出并重启服务

[root@node1 conf]#/etc/init.d/tomcat stop

[root@node1 conf]#/etc/init.d/tomcat start

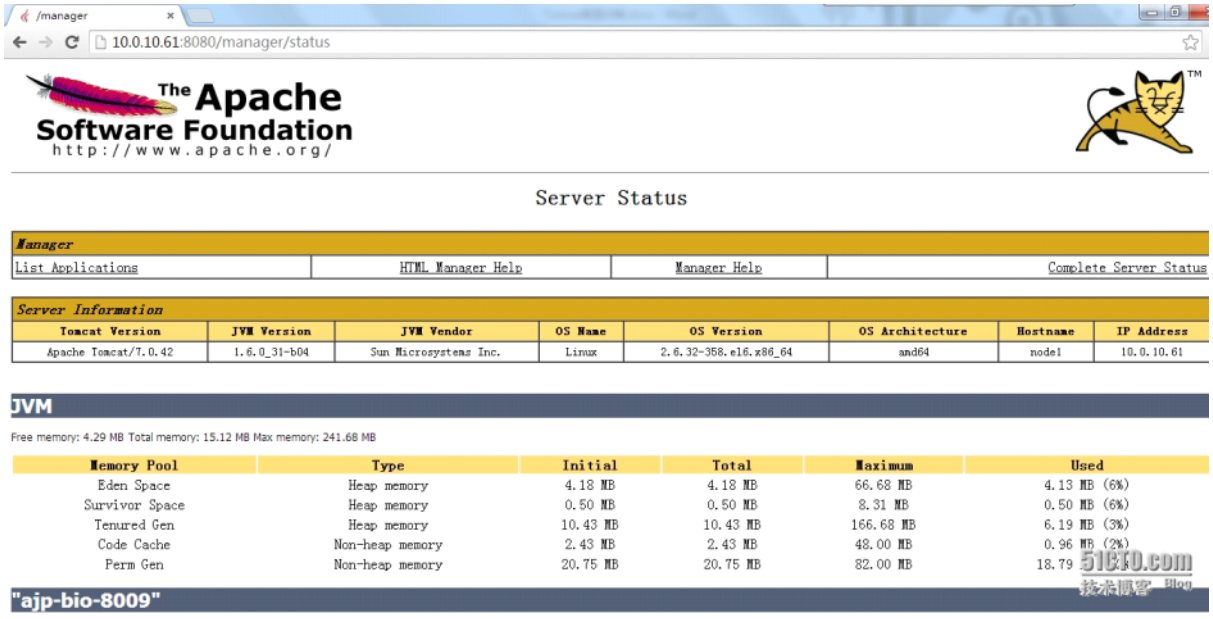
查看监听端口是否正常

```
[root@node1 conf]#netstat -lntup | grep java

tcp        0      0 :::8080                     :::
*                                   LISTEN      2359/java

tcp        0      0 ::ffff:127.0.0.1:8005       :::
*                                   LISTEN      2359/java

tcp        0      0 :::8009                     :::
*                                   LISTEN      2359/java
```

访问测试：

输入用户名和密码，选择登陆


如下所示，已经成功登陆至后台信息页面



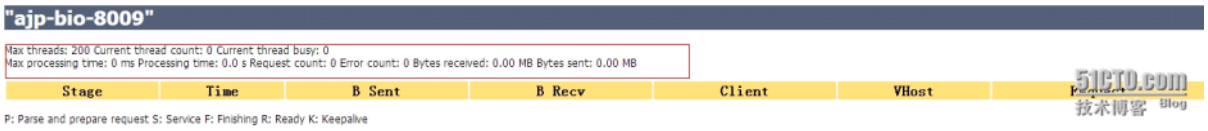

Type　　#所属类型类型

Initial　　#初始化空间多大

Total

Maximum　　#最大空间有多大

Used　　#已经使用了多少

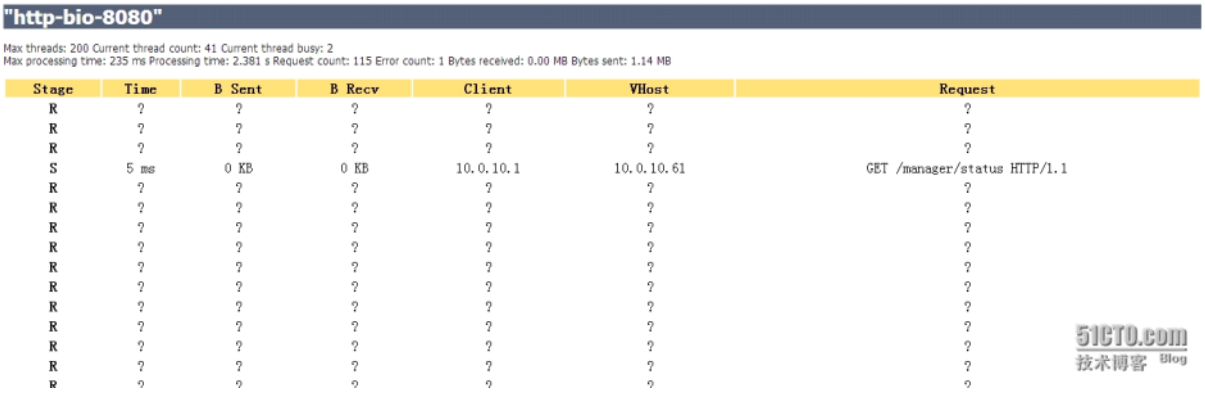



Max threads: 200　　默认最大并发连接数200

Current thread count: 0　　当前连接数

Current thread busy: 0　　繁忙连接数

对其做压力测试并观察其连接状态，并再次刷新页面




状态说明：

R：　已经准备好

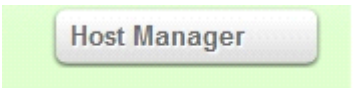
S：　正在提供服务

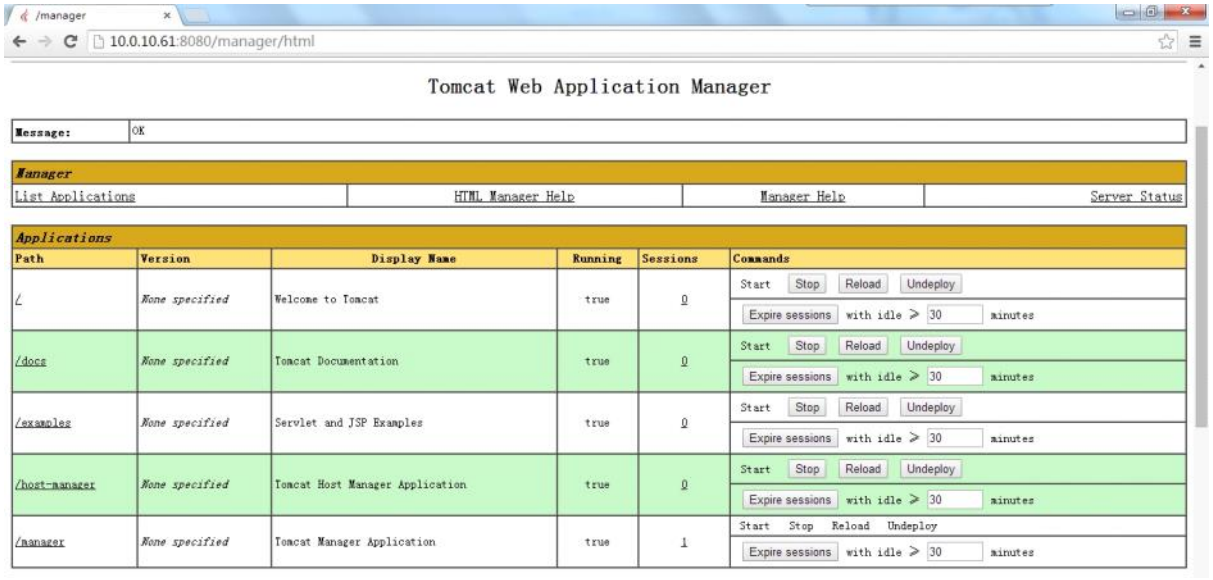
k：持久连接

**登陆后台管理页面**

由于我们刚才定义tomcat-user.xml的时候已经将用户tomcat加入到admin-gui组中，所以我们直接登陆就可以了

打开主页面，选择Host Manager






*Host Manager* 表示有多少个虚拟主机

如果想部署新的虚拟主机可以在下面的图形界面进行部署，如下所示

**tomcat应用程序目录结构**

/WEB-INF/web.xml：包含当前webapp的deploy描述符，如所有的servlet和JSP等动态文件的详细信息，会话超时时间和数据源等；因此，其也通常用于定义当前webapp特有的资源；

/WEB-INF/classes: 包含所有服务器端类及当前应用程序相关的其它第三方类等；
/WEB-INF/lib: 包含JSP所用到的JAR文件；
/META-INF/ :也是用来存放资源信息的，只不过一般用的不是特别多，就算有的话里面程序也未必有文件；

webapps/manager/META-INF/context.xml :上下文配置文件，有些配置信息可以放在META配置当中；

**使用自定义位置**

如果我们自己要发布一个资源，使用其他特定的目录来发布应用程序的话，也是按照这种格式：

[root@node1tomcat]# mkdir /tomcat/app1/WEB-INF/{classess,lib} -p

进入目录

[root@node1 app1]#pwd

/tomcat/app1

创建index.jsp,内容如下：

[root@node1 app1]#cat index.jsp

<%@ pagelanguage="java" %>

```
<html>
 <head><title>TomcatA</title></head>
  <body>
    <h1><fontcolor="red">TomcatA </font></h1>
    <tableborder="1">
      <tr>
        <td>Session ID</td>
    <%session.setAttribute("abc","abc"); %>
        <td><%= session.getId()%></td>
      </tr>
      <tr>
        <td>Created on</td>
        <td><%=session.getCreationTime() %></td>
     </tr>
    </table>
  </body>
</html>
```

**定义虚拟主机**

编辑server.xml

```
[root@node1 conf]#vim server.xml
```

加入如下内容

```
      <Host name="www.test.com" appBase="webapps"
            unpackWARs="true"autoDeploy="true">
        <ValveclassName="org.apache.catalina.valves.AccessLogValve"dire
ctory="logs"
               prefix="www_access_log."suffix=".txt"
               pattern="%h %l %u %t&amp;quot;%r&amp;quot; %s %b" />
        <Contextpath="/" docBase="/tomcat/app1" />                #定义访
```

问的路径当于整个路径的根，docBase表示访问的路径的位置

</Host>

这里没有主机名所以将默认主机设置为www.test.com

<Engine name="Catalina" defaultHost="www.test.com">

启动服务

```
[root@node1 conf]#/etc/init.d/tomcat start
```

或者

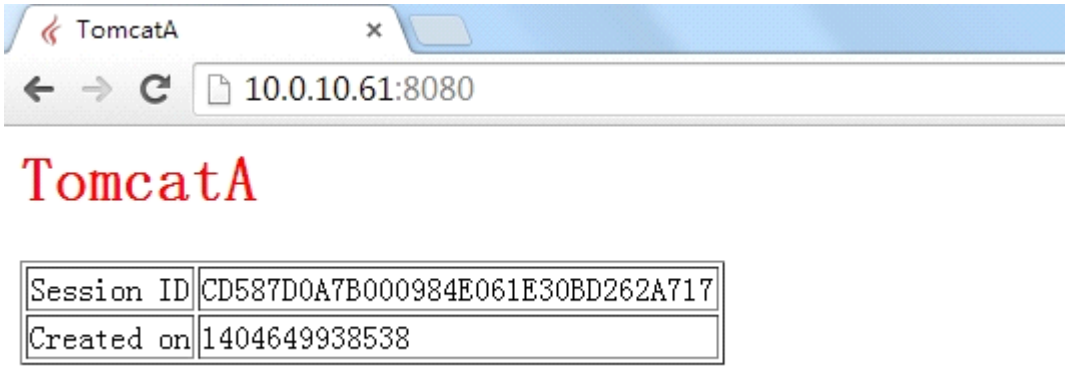
```
[root@node1 conf]#catalina.sh start
```

查看监听端口是否正常

```
[root@node1 conf]#netstat -lntup | grep java

tcp        0      0 :::8080                     :::
*                       LISTEN      2079/java

tcp        0      0 ::ffff:127.0.0.1:8005       :::
*                       LISTEN      2079/java

tcp        0      0 :::8009                     :::
*                       LISTEN      2079/java
```

访问测试：

http://10.0.10.61:8080/




第一次访问时，页面会很慢，是因为第一次访问jsp的时候都需要对其进行编译；

**编译jsp页面的执行次序**

jsp页面通过jasper进行编译 再由编译器 javac编译为 class文件因此每一次都需要编译的，如果不改文件刷新速度是非常快

如果是更改了文件的话，第一次访问可能会等待半天，所以任何时候更改完源程序都需要重新编译文件的

一旦页面被编译后会有什么样的结果：

[root@node1 conf]#cd Catalina/

[root@node1Catalina]# ll

total 8

drwxr-xr-x. 2 rootroot 4096 Jun 26 10:48 localhost

drwxr-xr-x. 2root root 4096 Jul 1 10:45 www.test.com #每一个host都有一个独立的目录

[root@node1 Catalina]# pwd

/usr/local/tomcat/work/Catalina

[root@node1Catalina]# ll www.test.com/ /org/apache/jsp/index_jsp.

index_jsp.class index_jsp.java

#此处已验证了，首先由jasper翻译成serverlet，再由其编译成class文件

**实例：部署论坛程序**

论坛程序包：JavaCenter_Home_2.0_GBK.zip

下载源码包可以去jsprun官方进行下载

解压源码包并拷贝至相关目录

[root@node1 ~]# unzip JspRun\!_6.0.0_UTF8.zip

[root@node1 ~]# cdJspRun\!_6.0.0_UTF8/

[root@node1upload]# cp -fra * /tomcat/app1

启动tomcat

[root@node1 app1]#catalina.sh start

启动mysql

[root@node1 app1]#/etc/init.d/mysqld start

将mysql赋予授权

mysql> grantall on jchome.* to 'jcuser'@'localhost' identified by 'jdpass';
Query OK, 0 rows affected (0.00 sec)

mysql> flushprivileges;

创建数据库

mysql> createdatabase jchome;

修改论坛程序的config.properties配置文件

[root@node1 app1]#pwd

/tomcat/app1

[root@node1 app1]#vim config.properties

修改为：

dbhost = localhost

dbport=3306

dbuser = jcuser

dbpw = jdpass

dbname = jchome

pconnect = 0

访问 http://10.0.10.61:8080/upload/install.jsp 进行安装

安装过程略，最后迎接我们的则是崭新的论坛界面，如下所示：


END，感谢各位。

# SSH

2018年12月26日 14:44

Struts2 + Spring + Hibernate

# Windows下Oracle安装图解----oracle-win-64-11g详细安装步骤

星期五, 十二月 28, 2018　　1:27 下午

**一、 Oracle 下载**

官方下地址

http://www.oracle.com/technetwork/database/enterprise-edition/downloads/index.html

win 32位操作系统 下载地址：

http://download.oracle.com/otn/nt/oracle11g/112010/win32_11gR2_database_1of2.zip

http://download.oracle.com/otn/nt/oracle11g/112010/win32_11gR2_database_2of2.zip

win 64位操作系统 下载地址：

http://download.oracle.com/otn/nt/oracle11g/112010/win64_11gR2_database_1of2.zip

http://download.oracle.com/otn/nt/oracle11g/112010/win64_11gR2_database_2of2.zip

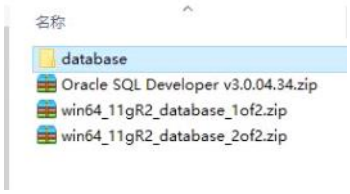
360网盘下载：https://yunpan.cn/cYDxKeQTRtVei 访问密码 f871

注意Oracle分成两个文件，下载完后，将两个文件解压到同一目录下即可。 路径名称中，最好不要出现中文，也不要出现空格等不规则字符。

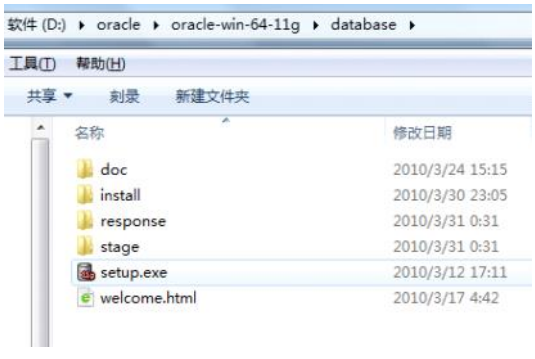
**二、Oracle安装**

1. 安装前提，以及准备

（1） 解压缩文件，将两个压缩包一起选择， 鼠标右击 -> 解压文件。




（2）两者解压到相同的路径中。



（3） 到相应的解压路径上面，找到可执行安装文件【 setup.exe 】双击安装。

中间会有个检测的过程：


2. 详细的安装步骤开始：

（1）安装第一步：配置安全更新。

这步可将自己的电子邮件地址填写进去（也可以不填写，只是收到一些没什么用的邮件而已）。取消下面的“我希望通过My Oracle Support接受安全更新(W)”。

直接忽略填写订阅信息：


可以点击“是”，进入下一步。

直接填写订阅信息：


我这里是连接失败。对我们的安装无影响。


（2）安全选项。

直接选择默认创建和配置一个数据库(安装完数据库管理软件后，系统会自动创建一个数据库实例)。


（3）系统类。

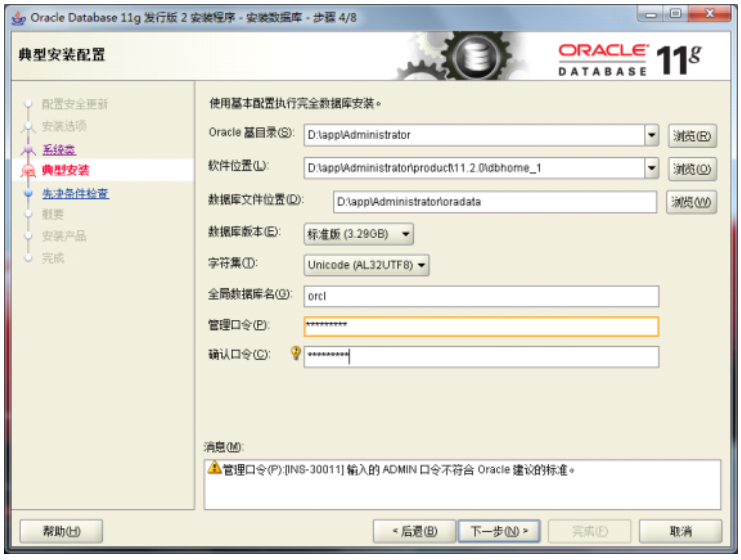
直接选择默认的桌面类就可以了。(若安装到的电脑是，个人笔记本或个人使用的电脑使用此选项)。


（4） 典型安装。

重要步骤。建议只需要将Oracle基目录更新下，目录路径不要含有中文或其它的特殊字符。全局数据库名可以默认，且口令密码，必须要牢记。密码输入时，有提示警告，不符合Oracel建议时不用管。（因Oracel建议的密码规则比较麻烦， 必须是大写字母加小写字母加数字，而且必须是8位以上。麻烦，可以输入平常自己习惯的短小密码即可） 。


默认账号为：orcl ，自己填写好密码。


提示可以不用理会。




若输入的口令短小简单，安装时会提示如下。直接确认Y继续安装就是了。


（5） 先决条件检查。

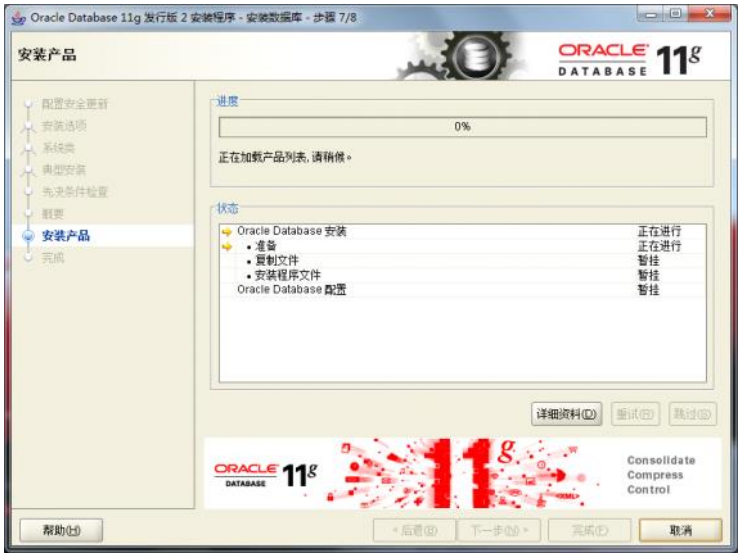
安装程序会检查软硬件系统是否满足，安装此Oracle版本的最低要求。 直接下一步就 OK 了。


（6）概要。

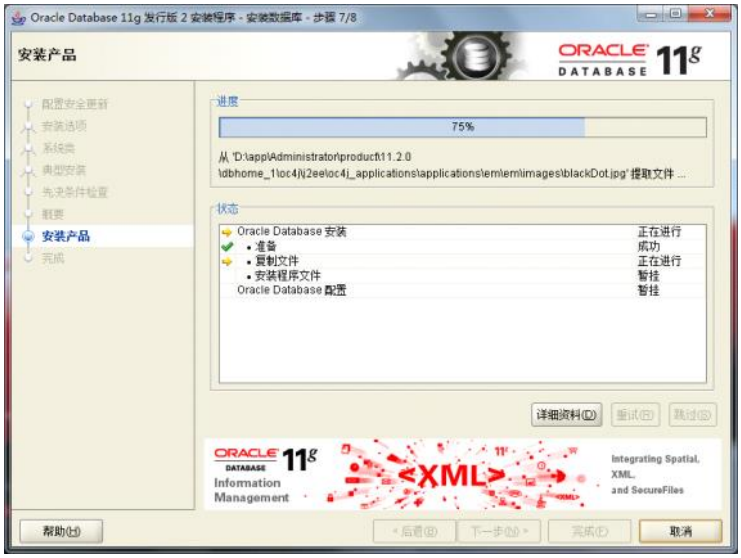
安装前的一些相关选择配置信息。 可以保存成文件 或 不保存文件直接点完成即可。

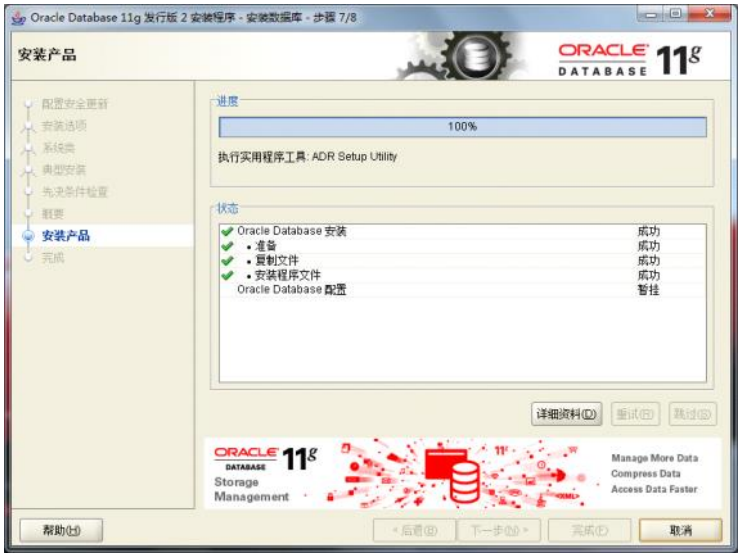

(7) 安装产品 。

自动进行，不用管。



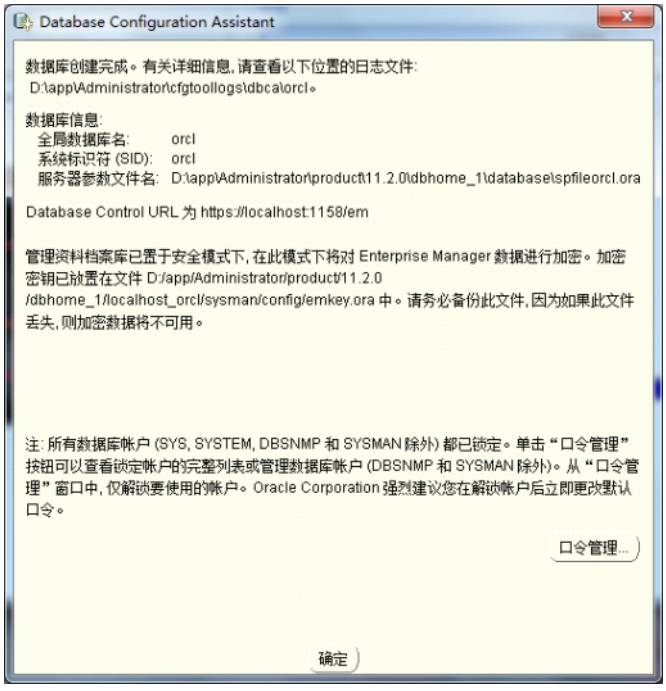

等待一下




数据库管理软件文件及dbms文件安装完后，会自动创建安装一个实例数据库默认前面的orcl名称的数据库。


实例数据库创建完成了，系统 默认是把所有账户都锁定不可用了(除sys和system账户可用外)，建议点右边的口令管理，将常用的scott账户解锁并输入密码。




解锁scott账户， 去掉前面的绿色小勾，输入密码。同样可以输入平常用的短小的密码，不必非得按oracle建议的8位以上大小写加数字，麻烦。

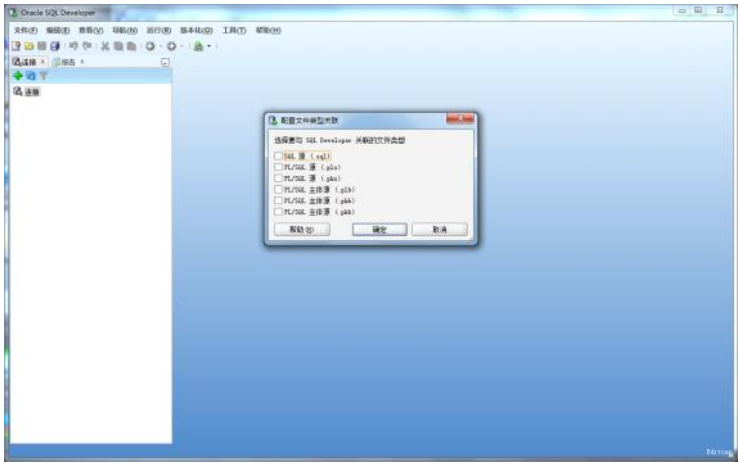
可以不用理会，后面用到再设置。


同样，密码不符合规则会提示。不用管它，继续Y即可。

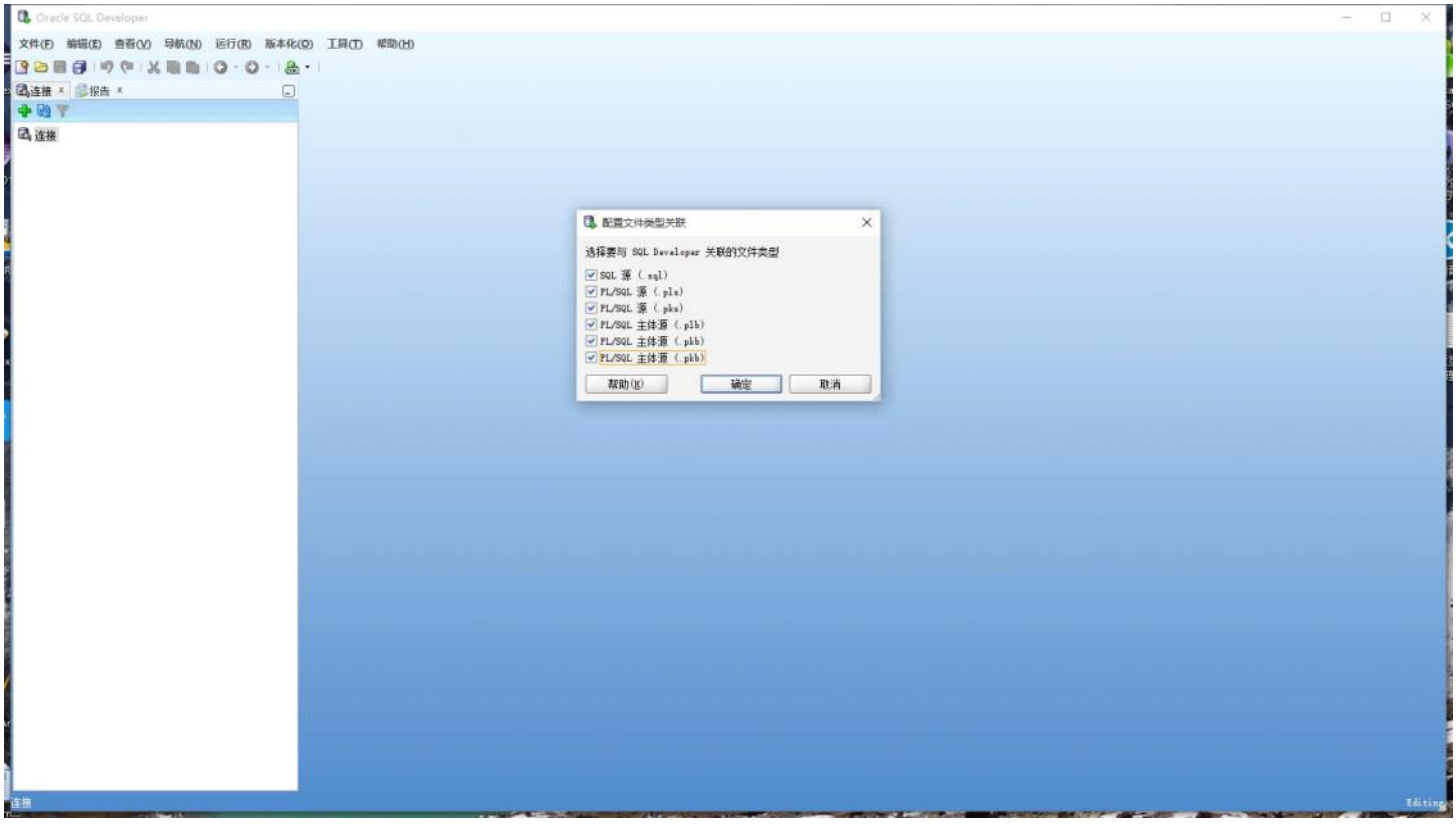
(8) 安装成功, 完成即可。


**三、安装oracle界面工具。**

1. 关联文件。

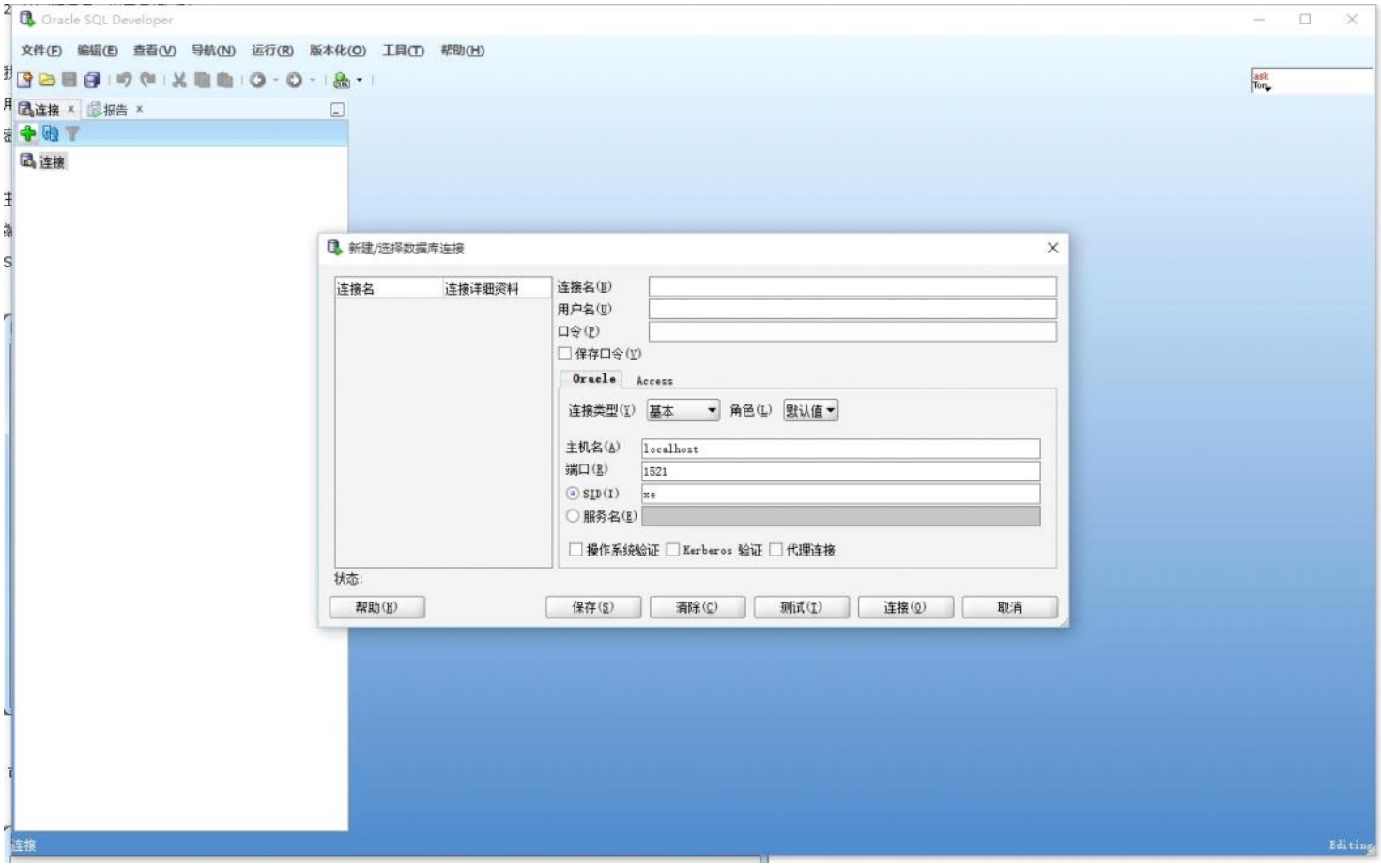


我这里是全选。




2. 新建连接，填写链接名，用户名和密码。

默认界面：




我们默认的连接名是使用orcl

用户名 ： system

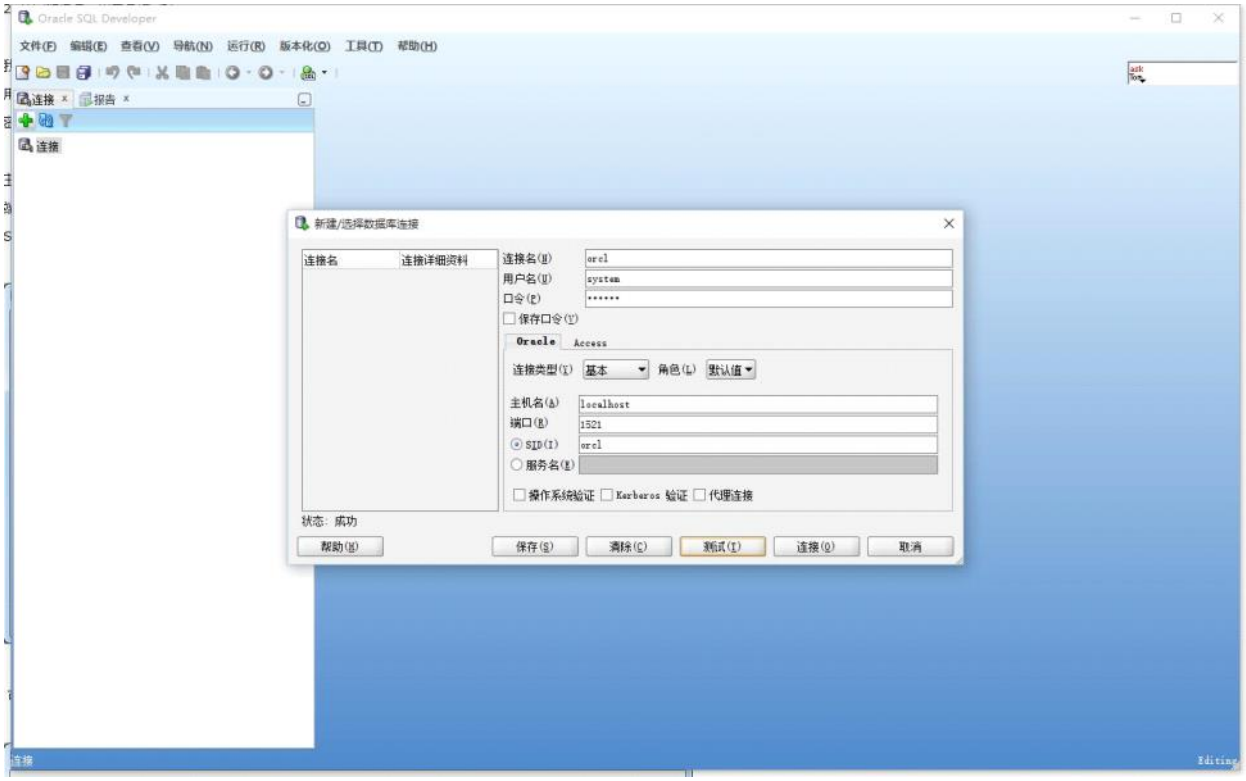
密码：刚刚设置的

主机名为：默认为本地（localhost）

端口号：1521

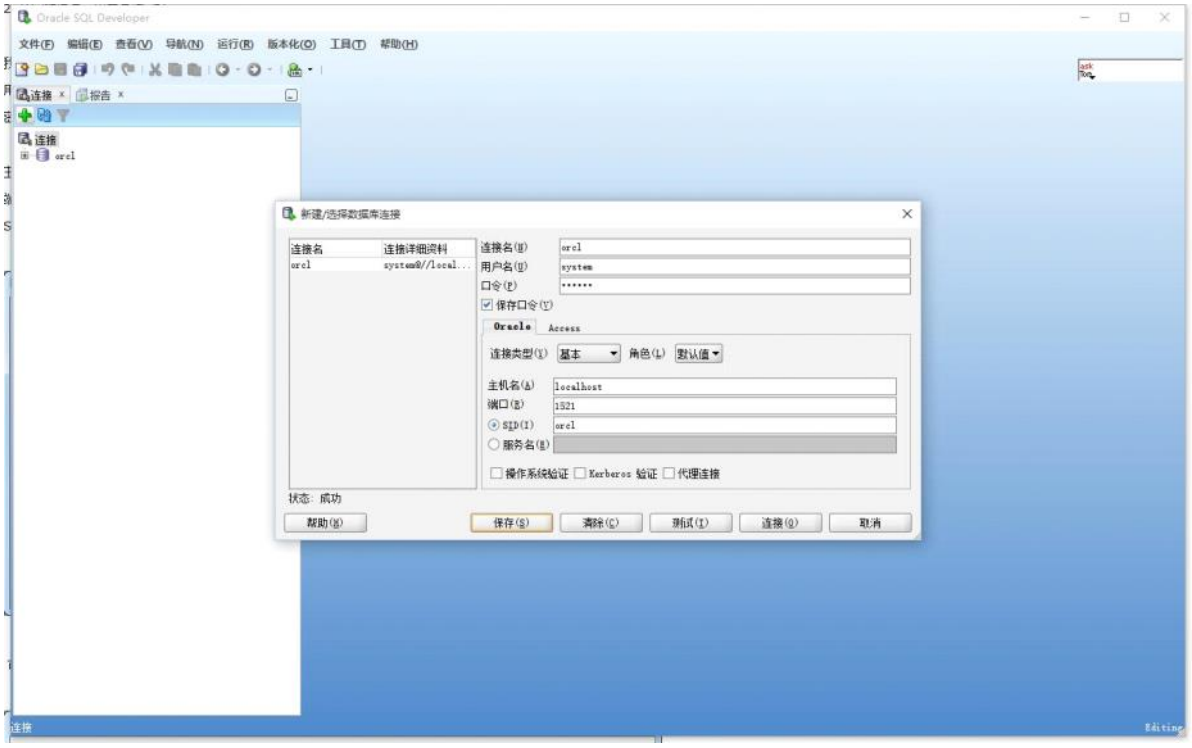
SID： xe

注意我们的需要修改xe 为 orcl 。可以点击 测试 来进行测试，是否连接上去。

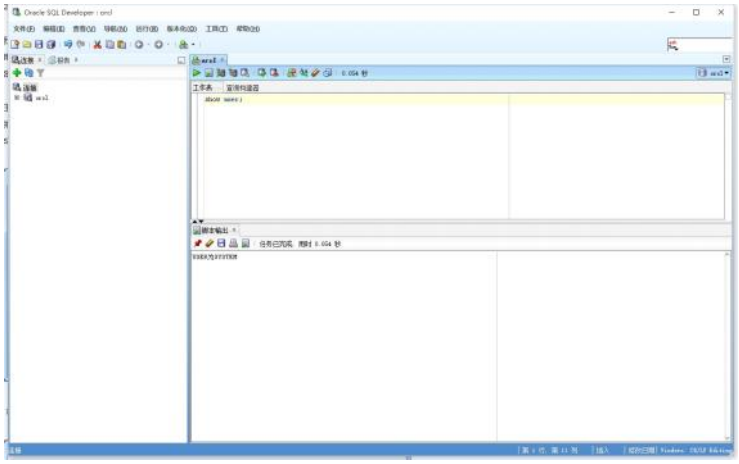



测试成功后，可以填写保存口令。方便以后登录用户。

保存一下信息：




执行一条语句, 效果如下：



# Oracle教程

SQL 几个要点

附录：
1、SQL 简介
2、SQL 操作符
3、Oracle 常用数据类型
4、Oracle 函数
5、[转] Oracle 常用SQL语法

字符串函数
LENGTH() 字符长度
LENTTHB() 字节长度；一个汉字内存中占用 2字节

LTRIM、RTRIM、TRIM

截串
SUBSTR(表达式，位置，长度)
Oracle 无左右取串函数，但可以使用变通方式完成。
左取串： SUBSTR('abcdefg', 1, 3)
右取串： SUBSTR('abcedfg', LENGTH('abcdefg')-3+1, 3)

时间函数
sysdate、current_day
设置时间格式： ALERT SESSION SET NLS_DATE_FORMAT = 'dd-mon-yyyy HH:mi:ss'
求时间： NEXT_DAY(sysdate, '星期三')

转换函数
TO_CHAR(sysdate, 'yyyy-mm-dd hh24:mi:ss')
TO_DATE('12-3月-04')
TO_NUMBER('333') 必须是能转换
TO_TIMESTAMP('2007-10-10 00:00:00.0', 'yyyy-mm-dd hh24:mi:ssxff') 转换为时间戳格式

聚合函数
count(*) ：查询表行数
count(column) :查询列行数，会忽略空值，注意
ps.聚合函数不能做为 where 里查询条件出现（因为聚合是对所有查询结果的运算？）

其他函数
USER：当前用户
SUM(DECODE(SEX, '男', 1, 0)) 筛选出行被为男的记录 并加1
SUM(DECODE(SEX, '女', 1, 0)) 筛选出行被为女的记录 并加1
NVL(a2, '非输入') 布尔值判断，利用系统对空值进行处理
SELECT DISTINCT a1 FROM aa

表连接
 内连接：查询时，把能够公共匹配的数据完全查询出来。
 FROM e, d WHERE e.id = d.id
 标准： FROM e JOIN d ON e.id = d.id

外连接：不完全匹配
 左连接： FROM e JOIN d ON e.id = d.id(+)
 左边数据全部显示，右边匹配不上的部分用空值代替
 右连接： FROM e JOIN d ON e.id(+) = d.id
 （同理左连接）

子查询
 无关子查询
 相关子查询

EXISTS()： 根据子查询返回是否存在数据来决定父查询。

UNION： 将多个查询出来的信息行整合成一个结果集。
 SELECT eid, ename FROM e
 UNION
 SELECT id, name FROM d
 ps.UNION 查询出来的重复记录不会显示，UNION ALL 则显示全部（包括重复的）。

INTERSECT: 返回查询出来信息行的交集，Oracle 独有。

利用查询结果批量更新：
 INSERT INTO e(eid, ename) SELECT id, name FROM d
 或者利用查询结果创建新表：
 CREATE TABLE ttt AS ttt (SELECT * FROM e)

附加：

SQL 简介
 SQL （Structured Query Language，结构化查询语言）支持如下类别命令：
 数据定义语言：CREATE（创建）、ALTER（更改）、TRUNCATE（截断）、DROP（删除）命令。
 数据操纵语言：INSERT（插入）、SELECT（选择）、DELETE（删除）、UPDATE（修改）命令。
 事务控制语言：COMMIT（提交）、SAVEPOINT（保存点）、ROLLBACK（回滚）命令。
 数据控制语言：GRANT（授予）、REVOKE（回收）命令。

特点：
 1、非过程语言，它同时可以访问多条记录。
 2、所有关系型数据库的通用型语言，可移植性强。
 3、对于数据和对象的操作简单。

数据定义语言
 用于改变数据库结构，包括创建、修改和删除数据库对象。

1、CREATE TABLE 创建表
 CREATE TABLE [schema.]table

(columname datetype [, .]);

.表名的最大长度为30个字符；
.表名首字母为字母，可以用下划线、数字和字母，但不能使用空格和单引号；
.同一用户模式下的不同表不能有相同的名称；
.表名、列名、用户名、和其他对象名不区分大小写，系统会自动转换成大写。

2、ALTER TABLE 修改表
ALTER TABLE
MODIFY (column definition);
ADD (column definition);
DROP COLUMN column;

3、TRUNCATE TABLE 截取表
TRUNCATE TABLE ;
快速删除记录并释放空间，不使用事务处理，无法回滚，效率高。

DESC 查看表结构

4、DROP TABLE 删除表
DROP TABLE

数据操纵语言
DISTINCT 防止选择重复的行。

事务控制语言
COMMIT 提交并结束事务处理。
SAVEPOINT 保存点，将很长的事务处理划分为较小的部分，用于标记事务中可以应用回滚的点。
ROLLBACK 用来撤销在当前的事务中已完成的操作。可以回滚整个事务处理；也可以将事务回滚到某个保存点。

```
UPDATE xxx;
SAVEPOINT mark1;
DELETE FROM xxx;
SAVEPOINT mark2;
ROLLBACK TO SAVEPOINT mark1;
COMMIT;
```

数据控制语言
为用户提供权限控制命令。

授予对象权限
```
GRANT SELECT,UPDATE ON order_master
TO MARTIN;
```

取消对象权限
```
REVOKE SELECT,UPDATE ON order_master
FROM MARTIN;
```

SQL 操作符

算术操作符
 算术表达式有 NUMBER 数据类型的列名、数值常量和连接它们的算术操作符组成。（+ - * /）

比较操作符
 用于比较两个表达式的值。
 =、!=、<、>、<=、>=、BETWEEN AND （检查是否在两个值之间）
 [NOT] IN（与列表中的值匹配）
 [NOT] LIKE（匹配字符模式， * _ 通配符）
 [NOT] IS NULL（检查是否为空）

逻辑操作符
 用于组合生成一个真或假的结果。AND OR NOT

集合操作符
 集合操作符将两个查询的结果组合成一个结果集合。
 UNION（联合） 返回两个查询选定不重复的行。(删除重复的行)
 UNION ALL（联合所有） 合并两个查询选定的所有行，包括重复的行。
 INTERSECT（交集） 只返回两个查询都有的行。
 MINUS（减集） 在第一个查询结果中排除第二个查询结果中出现的行。 （第一 -- 第二）

使用集合操作符连接起来的 SELECT 语句中的列应遵循以下规则：
 .通过集合操作连接的各个查询相同列数，匹配列的数据类型；
 .这种查询不应含有 LONG 类型的列；
 .列标题来自第一个 SELECT 语句。

```
SELECT orderno FROM order_master
 UNION
 SELECT orderno FROM order_detail;
```

连接操作符 （||）
 用于将两个或者多个字符串合并成一个字符串，或者将一个字符串与一个数值合并在一起。

```
 SELECT ('供应商'|| venname || '的地址是' || venaddress)
 FROM vendor_master
```

Oracle 常用数据类型

1、字符数据类型
 CHAR 固定长度字符串 长度 1～2000个字节，未指定则默认为 1字节
 VARCHAR2 可变长度字符串 长度 1～4000个字节，定义时必须指定大小
 LONG 可变长度字符串 最多能存储 2GB，存储超过 VARCHAR2 的长文本信息
 ps.一个表中只有一列为 LONG 数据类型，
 .LONG 列不能建立索引，
 .存储过程不能接受 LONG 数据类型的参数

2、数值数据类型
 NUMBER 数据类型可以存储 正数、负数、零、定点数(不带小数点的？)和精度为38为的浮点数。
 格式： NUMBER [(precision 精度，数字总位数 1～38间

, scale 范围，小数点右边的位数 -84～127间)]

3、时期时间数据类型
DATE 数据类型，用于存储表中日期和时间数据。SYSDATE 函数功能就是返回当前的日期和时间。
TIMESTAMP 数据类型，存储时期、时间和时区信息。SYSTIMEATAMP 功能就是返回当前日期、时间和时区。

4、二进制数据类型
RAW 二进制数据或字节串 长度 1～2000 字节，定义时应指定大小，可建索引
LONG RAW 可变长度的二进制数据 最大能存储 2GB，限制等同于 LONG 数据类型

5、LOB 数据类型
“大对象”数据类型，最多可存储多达 4GB的信息。LOB 可以是外部的，也可以是内部的，取决于相对于数据库位置。
CLOB Character LOB 存储大量的字符数据
BLOB Binary LOB 存储大量的二进制对象（多媒体对象等）
BFILE Binary FIle 能够将二进制文件存储在数据库外部的操作系统文件中
BFILE 存储一个 BFILE 定位器，它指向位于服务器文件系统上的二进制文件。
ps.一个表中可以有多个 LOB 列，每个 LOB 列可以是不同的 LOB 类型。

6、伪列
Oracle 中的一个表列，但实际上未存储表中。可以从表中查询，但是不能插入，更新或者删除。

ROWID 返回行记录的行地址，通常情况下，ROWID 值可以唯一地标识数据库中的一行。
作用： .能最快形式访问表中的一行。
.能显示表中的行是如何存储的。
.可以作为表中行的唯一标识。
例：SELECT ROWID, * FROM EMP WHERE empno=’7900’;

ROWNUM 返回一个数值单表行的次序，第一行为1，第二行为2.
通过使用 ROWNUM 用户可以限制查询返回的行数(或者分页？)
例：SELECT * FROM EMP WHERE ROWNUM <= 10;

Oracle 函数
函数接受一个或多个参数并返回一个值。

单行函数
也称标量函数，对于从表中查询的每一行，该函数都返回一个值。
单行函数出现在 SLEECT / WHERE 子句中。

1、日期函数
对日期值进行运算，根据用途产生日期/数值类型的结果。
ADD_MONTHS(d, n) 返回 指定日期加上月数后的 日期值
MONTHS_BETWEEN(d1, d2) 返回 两个日期间的 月数
LAST_DAY(d) 返回 指定日期当前的最后一天的 日期值
RONUD(d,[fmt]) 返回 指定日期四舍五入格式(YEAR、MONTH、DAY)后的 日期值
NEXT_DAY(d,day) 返回 指定日期下一个星期几的 日期值
TRUNC(d,[fnt]) 返回 指定日期截断为格式后的 日期值

EXTRACT(fmt FROM d) 返回 指定日期提取的格式的 值

2、字符函数

字符函数接受字符输入，并返回字符或数值。

INITCAP(char) 首字母大写

LOWER(char) 转换为小写

UPPER(char) 转换为大写

LTRIM(char, set) 左裁切

RTRIM(char, set) 右裁切

TRANSLATE(char, from, to) 按字母翻译

REPLACE(char, search_str, replace_str) 字符串替换

INSTR(char, substr[,pos]) 查找子串位置

SUBSTR(char, pos, len) 取子字符串

CONCAT(char1, char2) 连接字符串

CHR(ascii) 根据 ASCII 码返回对应字符串

LPAD / RPAD 左 / 右 填充

LPAD ('function', 15 , '=') 返回 '=======function'

TRAM 开头或结尾(或 开头和结尾)裁剪特定的字符，默认裁剪空格。

TRIM ([LEADING | TRAILING] trim_char)

LENGTH(char) 返回字符串长度

DECODE 逐个值进行字符串替换

DECODE (expr, search1, result1, search2, result2, [ ,default])

DECODE (ostalus, 'p', '准备处理', 'c', '已完成')

3、数字函数

数字函数接受数字输入并返回数值作为输出结果。

ABS(n) 取绝对值

CEIL(n) 向上取值

FLOOR(n) 向下去整

SIN(n) 正弦值

COS(n) 余弦值

POWER(m, n) 指数函数

SQRT(n) 平方根

MOD(m, n) 取余

ROUND(m, n) 小数点后精度四舍五入

TRUNC(m, n) 小数点后精度截断

4、转换函数

转换函数将一种数据类型转换为另一种数据类型。

TO_CHAR (d|n, [,fmt]) 格式化 日期 / 数值

TO_DATE (char [,fmt]) 将 fmt模型格式的字符串 转换为日期型

TO_NUMBER (char) 将 包含数字的的字符串转换为 数值型

5、其他函数

NVL (exp, exp2) 如果 exp 为空返回 exp2；如果非空返回 exp

NVL2 (exp, exp2, exp3) 如果 exp 为空返回 exp3；如果非空返回 exp2

NULLIF (exp1, exp2) 比较两表达式，相等返回空值，不等则返回 exp1

分组函数 / 聚合函数

分组函数基于一组行返回结果，即为每一组行返回单个值。

AVG (columname) 返回指定列的平均值
MAX (columname) 返回指定列的最大值
MIN (columname) 返回指定列的最小值
SUM (columname) 返回指定列的总值
COUNT
COUNT (*) 统计所有行个数，包括重复行和空值得行
COUNT (columname) 统计指定列非空值的个行数
COUNT (DISTINCR columname) 统计指定列中 非重复，非空值得行个数

GROUP BY 子句
用于将信息表划分为组，对查询结果按组进行聚合运算，为每组返回一个结果。
HAVING 子句
用来指定 GROUP BY 子句的检索条件。

分析函数
分析函数根据一组行来计算聚合值。这些函数通常用来完成对聚集的累积排名、移动平均数和报表计算。
分析函数与聚合函数不同的是他们为每组记录返回多个行。

ROW_NUMBER () OVER ([PARTITION BY colum] ORDER BY colum)
为有序组中的每一行返回一个唯一的排序值，序号由 ORDER BY 子句指定，从 1 开始，即使具有相等的值，排位也不同。
PARTITION BY colum 按列值进行区分，各分组内在进行排序。

RANK () OVER ([PARTITION BY colum] ORDER BY colum)
计算一个值在一个组中的地位，由 1 开头，具有相等值得行排位相同，序数随后跳跃相应的数值。

DENSE_RANK () OVER ([PARTITION BY colum] ORDER BY colum)
计算一个值在一个组中的地位，由 1 开头，具有相等值得行排位相同，并且排位是连续的。

5、[转] Oracle 常用SQL语法和数据对象

==========

一.数据控制语句 (DML) 部分

==========

1.INSERT (往数据表里插入记录的语句)

INSERT INTO 表名(字段名1, 字段名2, ……) VALUES ( 值1, 值2, ……);
INSERT INTO 表名(字段名1, 字段名2, ……) SELECT 字段名1, 字段名2, …… FROM 另外的表名;

字符串类型的字段值必须用单引号括起来, 例如: ’GOOD DAY’
如果字段值里包含单引号’ 需要进行字符串转换, 我们把它替换成两个单引号’’.
字符串类型的字段值超过定义的长度会出错, 最好在插入前进行长度校验.

日期字段的字段值可以用当前数据库的系统时间SYSDATE, 精确到秒
 或者用字符串转换成日期型函数TO_DATE(‘2001-08-01’,’YYYY-MM-DD’)
 TO_DATE()还有很多种日期格式, 可以参看ORACLE DOC.
 年-月-日 小时:分钟:秒 的格式YYYY-MM-DD HH24:MI:SS

INSERT时最大可操作的字符串长度小于等于4000个单字节, 如果要插入更长的字符串, 请考虑字段用CLOB类型,
 方法借用ORACLE里自带的DBMS_LOB程序包.

INSERT时如果要用到从1开始自动增长的序列号, 应该先建立一个序列号
 CREATE SEQUENCE 序列号的名称 (最好是表名+序列号标记) INCREMENT BY 1 START WITH 1
 MAXVALUE 99999 CYCLE NOCACHE;
 其中最大的值按字段的长度来定, 如果定义的自动增长的序列号 NUMBER(6) , 最大值为999999
 INSERT 语句插入这个字段值为: 序列号的名称.NEXTVAL

2.DELETE (删除数据表里记录的语句)

DELETE FROM表名 WHERE 条件;

注意：删除记录并不能释放ORACLE里被占用的数据块表空间. 它只把那些被删除的数据块标成unused.

如果确实要删除一个大表里的全部记录, 可以用 TRUNCATE 命令, 它可以释放占用的数据块表空间
 TRUNCATE TABLE 表名;
 此操作不可回退.

3.UPDATE (修改数据表里记录的语句)

UPDATE表名 SET 字段名1=值1, 字段名2=值2, …… WHERE 条件;

如果修改的值N没有赋值或定义时, 将把原来的记录内容清为NULL, 最好在修改前进行非空校验;
 值N超过定义的长度会出错, 最好在插入前进行长度校验..

注意事项:

A. 以上SQL语句对表都加上了行级锁,
 确认完成后, 必须加上事物处理结束的命令 COMMIT 才能正式生效,
 否则改变不一定写入数据库里.
 如果想撤回这些操作, 可以用命令 ROLLBACK 复原.

B. 在运行INSERT, DELETE 和 UPDATE 语句前最好估算一下可能操作的记录范围,
 应该把它限定在较小 (一万条记录) 范围内,. 否则ORACLE处理这个事物用到很大的回退段.
 程序响应慢甚至失去响应. 如果记录数上十万以上这些操作, 可以把这些SQL语句分段分次完成,
 其间加上COMMIT 确认事物处理.

==========

==========
二.数据定义（DDL）部分

==========

1.CREATE（创建表，索引，视图，同义词，过程，函数，数据库链接等）

ORACLE常用的字段类型有
CHAR 固定长度的字符串
VARCHAR2 可变长度的字符串
NUMBER(M,N) 数字型M是位数总长度，N是小数的长度
DATE 日期类型

创建表时要把较小的不为空的字段放在前面，可能为空的字段放在后面

创建表时可以用中文的字段名，但最好还是用英文的字段名

创建表时可以给字段加上默认值，例如 DEFAULT SYSDATE
这样每次插入和修改时，不用程序操作这个字段都能得到动作的时间

创建表时可以给字段加上约束条件
例如 不允许重复 UNIQUE，关键字 PRIMARY KEY

2.ALTER（改变表，索引，视图等）

改变表的名称
ALTER TABLE 表名1 TO 表名2;

在表的后面增加一个字段
ALTER TABLE表名 ADD 字段名 字段名描述;

修改表里字段的定义描述
ALTER TABLE表名 MODIFY字段名 字段名描述;

给表里的字段加上约束条件
ALTER TABLE 表名 ADD CONSTRAINT 约束名 PRIMARY KEY (字段名);
ALTER TABLE 表名 ADD CONSTRAINT 约束名 UNIQUE (字段名);

把表放在或取出数据库的内存区
ALTER TABLE 表名 CACHE;
ALTER TABLE 表名 NOCACHE;

3.DROP（删除表，索引，视图，同义词，过程，函数，数据库链接等）

删除表和它所有的约束条件
DROP TABLE 表名 CASCADE CONSTRAINTS;

4.TRUNCATE（清空表里的所有记录，保留表的结构）

TRUNCATE 表名;

==========

==========

三.查询语句（SELECT）部分

==========

SELECT字段名1, 字段名2, …… FROM 表名1, [表名2, ……] WHERE 条件;

字段名可以带入函数

例如: COUNT(), *MIN(字段名), MAX(字段名), AVG(字段名), DISTINCT(字段名),*
*TO_CHAR(DATE字段名,'YYYY-MM-DD HH24:MI:SS')*

---

*NVL(EXPR1, EXPR2)函数*
*解释:*
*IF EXPR1=NULL*
*RETURN EXPR2*
*ELSE*
*RETURN EXPR1*

---

*DECODE(AA，V1，R1，V2，R2.)函数*
*解释:*
*IF AA=V1 THEN RETURN R1*
*IF AA=V2 THEN RETURN R2*
*..…*
*ELSE*
*RETURN NULL*

---

*LPAD(char1,n,char2)函数*
*解释:*
*字符char1按制定的位数n显示，不足的位数用char2字符串替换左边的空位*

---

*字段名之间可以进行算术运算*
*例如: (字段名1*字段名1)/3

---

查询语句可以嵌套

例如: SELECT …… FROM
(SELECT …… FROM表名1, [表名2, ……] WHERE 条件) WHERE 条件2;

---

两个查询语句的结果可以做集合操作
例如: 并集 UNION (去掉重复记录),
并集 UNION ALL(不去掉重复记录),
差集 MINUS,
交集 INTERSECT

分组查询

SELECT字段名1, 字段名2, …… FROM 表名1, [表名2, ……] GROUP BY字段名1
[HAVING 条件] ;

两个以上表之间的连接查询

```
SELECT 字段名1, 字段名2, ……
 FROM 表名1, [表名2, ……]
 WHERE 表名1.字段名 = 表名2. 字段名
 [ AND ……] ;

SELECT字段名1, 字段名2, ……
 FROM 表名1, [表名2, ……]
 WHERE 表名1.字段名 = 表名2. 字段名(+)
 [ AND ……] ;
```

## 有(+)号的字段位置自动补空值

查询结果集的排序操作, 默认的排序是升序ASC, 降序是DESC

```
SELECT字段名1, 字段名2, …… FROM 表名1, [表名2, ……]
 ORDER BY字段名1, 字段名2 DESC;
 ------------------------------------------------------
 字符串模糊比较的方法

INSTR(字段名, '字符串')>0
 字段名 LIKE '字符串%' [ '%字符串%']
```

每个表都有一个隐含的字段ROWID, 它标记着记录的唯一性.

==========

四.ORACLE里常用的数据对象 (SCHEMA)

==========

1.索引 (INDEX)

```
CREATE INDEX 索引名ON 表名 ( 字段1, [字段2, ……] );
 ALTER INDEX 索引名 REBUILD;
```

一个表的索引最好不要超过三个 (特殊的大表除外), 最好用单字段索引, 结合SQL语句的分析执行情况,
 也可以建立多字段的组合索引和基于函数的索引

ORACLE8.1.7字符串可以索引的最大长度为1578 单字节
 ORACLE8.0.6字符串可以索引的最大长度为758 单字节

ORACLE DOC上说字符串最大可以建索引的长度约是:数据块的大小(db_block_size)*40%

2.视图 (VIEW)

```
CREATE VIEW 视图名AS SELECT …. FROM …..;
 ALTER VIEW视图名 COMPILE;
```

视图仅是一个SQL查询语句, 它可以把表之间复杂的关系简洁化.

3.同义词 (SYNONMY)
 CREATE SYNONYM同义词名FOR 表名;
 CREATE SYNONYM同义词名FOR 表名@数据库链接名;

4.数据库链接 (DATABASE LINK)
 CREATE DATABASE LINK数据库链接名CONNECT TO 用户名 IDENTIFIED BY 密码 USING ‘数据库连接字符串’;

数据库连接字符串可以用NET8 EASY CONFIG或者直接修改TNSNAMES.ORA里定义.

数据库参数global_name=true时要求数据库链接名称跟远端数据库名称一样

数据库全局名称可以用以下命令查出
 SELECT * FROM GLOBAL_NAME;

查询远端数据库里的表
 SELECT …… FROM 表名@数据库链接名;

==========

五.权限管理 (DCL) 语句

==========

1.GRANT 赋于权限
 常用的系统权限集合有以下三个:
 CONNECT(基本的连接), RESOURCE(程序开发), DBA(数据库管理)
 常用的数据对象权限有以下五个:
 ALL ON 数据对象名, SELECT ON 数据对象名, UPDATE ON 数据对象名,
 DELETE ON 数据对象名, INSERT ON 数据对象名, ALTER ON 数据对象名

GRANT CONNECT, RESOURCE TO 用户名;
 GRANT SELECT ON 表名 TO 用户名;
 GRANT SELECT, INSERT, DELETE ON表名 TO 用户名1, 用户名2;

2.REVOKE 回收权限

REVOKE CONNECT, RESOURCE FROM 用户名;
 REVOKE SELECT ON 表名 FROM 用户名;
 REVOKE SELECT, INSERT, DELETE ON表名 FROM 用户名1, 用户名2;

==========

**********

==========

==========

二.数据定义（DDL）部分

==========

1.CREATE（创建表，索引，视图，同义词，过程，函数，数据库链接等）

ORACLE常用的字段类型有
CHAR 固定长度的字符串
VARCHAR2 可变长度的字符串
NUMBER(M,N) 数字型M是位数总长度，N是小数的长度
DATE 日期类型

创建表时要把较小的不为空的字段放在前面，可能为空的字段放在后面

创建表时可以用中文的字段名，但最好还是用英文的字段名

创建表时可以给字段加上默认值，例如 DEFAULT SYSDATE
这样每次插入和修改时，不用程序操作这个字段都能得到动作的时间

创建表时可以给字段加上约束条件
例如 不允许重复 UNIQUE，关键字 PRIMARY KEY

2.ALTER（改变表，索引，视图等）

改变表的名称
ALTER TABLE 表名1 TO 表名2;

在表的后面增加一个字段
ALTER TABLE表名 ADD 字段名 字段名描述;

修改表里字段的定义描述
ALTER TABLE表名 MODIFY字段名 字段名描述;

给表里的字段加上约束条件
ALTER TABLE 表名 ADD CONSTRAINT 约束名 PRIMARY KEY（字段名);
ALTER TABLE 表名 ADD CONSTRAINT 约束名 UNIQUE（字段名);

把表放在或取出数据库的内存区
ALTER TABLE 表名 CACHE;
ALTER TABLE 表名 NOCACHE;

3.DROP（删除表，索引，视图，同义词，过程，函数，数据库链接等）

删除表和它所有的约束条件
DROP TABLE 表名 CASCADE CONSTRAINTS;

4.TRUNCATE（清空表里的所有记录，保留表的结构）

TRUNCATE 表名;

==========

==========

**********

==========

==========

三.查询语句（SELECT）部分

==========

SELECT 字段名1, 字段名2, ……
FROM 表名1, [表名2, ……]
WHERE 条件;
---
字段名可以带入函数
例如: COUNT(), *MIN(字段名), MAX(字段名), AVG(字段名), DISTINCT(字段名),*
*TO_CHAR(DATE字段名,'YYYY-MM-DD HH24:MI:SS')*
---
*NVL(EXPR1, EXPR2)函数*
*解释:*
*IF EXPR1=NULL*
*RETURN EXPR2*
*ELSE*
*RETURN EXPR1*
---
*DECODE(AA，V1，R1，V2，R2.)函数*
*解释:*
*IF AA=V1 THEN RETURN R1*
*IF AA=V2 THEN RETURN R2*
*..…*
*ELSE*
*RETURN NULL*
---
*LPAD(char1,n,char2)函数*
*解释:*
*字符char1按制定的位数n显示，不足的位数用char2字符串替换左边的空位*
---
*字段名之间可以进行算术运算*
*例如: (字段名1*字段名1)/3

==========
查询语句可以嵌套
---
例如:

```
SELECT …… FROM
(
SELECT ……
FROM表名1, [表名2, ……]
WHERE 条件1
)
WHERE 条件2;

============================================================
两个查询语句的结果可以做集合操作
------------------------------
例如: 并集 UNION (去掉重复记录),
并集 UNION ALL(不去掉重复记录),
差集 MINUS,
交集 INTERSECT

============================================================
分组查询
------------
SELECT 字段名1, 字段名2, ……
FROM 表名1, 表名2, ……
GROUP BY 字段名1
HAVING 条件

============================================================
两个以上表之间的连接查询
----------------------
SELECT 字段名1, 字段名2, ……
FROM 表名1, [表名2, ……]
WHERE 表名1.字段名 = 表名2. 字段名
[ AND ……] ;

SELECT 字段名1, 字段名2, ……
FROM 表名1, [表名2, ……]
WHERE 表名1.字段名 = 表名2. 字段名(+)
[ AND ……] ;

有(+)号的字段位置自动补空值

============================================================
查询结果集的排序操作, 默认的排序是升序ASC, 降序是DESC
-------------------------------------------------
SELECT 字段名1, 字段名2, ……
FROM 表名1, [表名2, ……]
ORDER BY 字段名1, 字段名2 DESC;

============================================================
字符串模糊比较的方法
----------------------
INSTR(字段名, ‘字符串’)>0
字段名 LIKE ‘字符串%’ [‘%字符串%’]
```

每个表都有一个隐含的字段ROWID, 它标记着记录的唯一性.

==========

==========

**********

**********

==========

==========

==========

==========

四.ORACLE里常用的数据对象 (SCHEMA)

==========

==========

1.索引 (INDEX)

```
CREATE INDEX 索引名ON 表名 ( 字段1, [字段2, ……] );
 ALTER INDEX 索引名 REBUILD;
```

一个表的索引最好不要超过三个 (特殊的大表除外), 最好用单字段索引, 结合SQL语句的分析执行情况,
也可以建立多字段的组合索引和基于函数的索引

ORACLE8.1.7字符串可以索引的最大长度为1578 单字节
ORACLE8.0.6字符串可以索引的最大长度为758 单字节

ORACLE DOC上说字符串最大可以建索引的长度约是:数据块的大小(db_block_size)*40%

---

2.视图 (VIEW)

```
CREATE VIEW 视图名AS SELECT …. FROM …..;
 ALTER VIEW视图名 COMPILE;
```

视图仅是一个SQL查询语句, 它可以把表之间复杂的关系简洁化.

3.同义词 (SYNONMY)

```
 CREATE SYNONYM同义词名FOR 表名;
 CREATE SYNONYM同义词名FOR 表名@数据库链接名;
```

4.数据库链接 (DATABASE LINK)

```
 CREATE DATABASE LINK数据库链接名CONNECT TO 用户名 IDENTIFIED BY 密码 USING
‘数据库连接字符串’;
```

数据库连接字符串可以用NET8 EASY CONFIG或者直接修改TNSNAMES.ORA里定义.

数据库参数global_name=true时要求数据库链接名称跟远端数据库名称一样

数据库全局名称可以用以下命令查出
 SELECT * FROM GLOBAL_NAME;

查询远端数据库里的表
 SELECT …… FROM 表名@数据库链接名;

==========

**********

==========

==========
 五.权限管理 (DCL) 语句

==========

1.GRANT 赋于权限
 常用的系统权限集合有以下三个:
 CONNECT(基本的连接), RESOURCE(程序开发), DBA(数据库管理)
 常用的数据对象权限有以下五个:
 ALL ON 数据对象名, SELECT ON 数据对象名, UPDATE ON 数据对象名,
 DELETE ON 数据对象名, INSERT ON 数据对象名, ALTER ON 数据对象名

GRANT CONNECT, RESOURCE TO 用户名;
 GRANT SELECT ON 表名 TO 用户名;
 GRANT SELECT, INSERT, DELETE ON表名 TO 用户名1, 用户名2;

2.REVOKE 回收权限

REVOKE CONNECT, RESOURCE FROM 用户名;
 REVOKE SELECT ON 表名 FROM 用户名;
 REVOKE SELECT, INSERT, DELETE ON表名 FROM 用户名1, 用户名2;

==========

**********

==========

==========

```
 ------------------------------------------------------------------------------
------

==============================================================================
 使用表达式
 ==============================================================================

Select ename || ’ 是一位 ’ || job As 雇员细节,
 to_char(hiredate,’yyyy-mm-dd’) As 雇佣时间,
 sal*1.2
 From emp;

==============================================================================
 取消重复行 distinct
 ==============================================================================

Select Distinct deptno, job From emp;

==============================================================================
 指定列排序 order by
 ==============================================================================
 asc 升序，desc 降序
 ----------------------------------------
 Select * From emp
 where sal between 1500 and 3000
 Order By deptno Desc,ename;
```

## 如果使用dsitinct,排序列必须是选择列

```
select distinct depton, job
 from emp
 order by job;
 ----------------------------------------
 order by 子句必须是最后一个子句
 ==============================================================================
 ==============================================================================

============================================================
 分组查询语句
 ============================================================
 1。组处理函数不能出现在 where 子句中
 2。选择列表中的列、表达式，必须出现在 group by 子句中
 3。组处理函数中可以指定 all 和 distinct
 ============================================================
 分组函数
 ------------------------------------
 select avg(sal) as avg1,
 avg(distinct sal) as avg2,
 max(sal) as max,
 min(sal) as min,
 sum(sal) as sum,
```

```
 count(*) as cnt1,
 count(sal) as cnt2,
 count(distinct sal) as cnt3
 from emp
 where deptno = 30;

=========================================================
 单列分组
 ---------------------------------------
 select deptno, avg(sal), max(sal) from emp
 group by deptno;

select deptno, avg(sal), max(sal) from emp
 group by deptno
 order by avg(sal);

=========================================================
 多列分组
 ---------------------------------------
 select deptno, job, avg(sal), max(sal), from emp
 group by deptno, job;

=========================================================
====
```

## rollup 用于生成横向统计信息

```
Select deptno, job, Avg(sal), Max(sal) From emp
 Group By rollup(deptno,job)
```

## cube 用于生产纵向统计信息

```
Select deptno, job, Avg(sal), Max(sal) From emp
 Group By Cube(deptno,job)

=========================================================
 having 子句
 =========================================================
 select deptno, avg(sal), max(sal) from emp
 group by deptno
 having avg(sal) > 2000;

=========================================================
====

=========================================================
 不等连接
 ---------------------------
 Select e.ename, e.sal, s.grade
 From emp e,salgrade s
```

```
Where e.sal Between s.losal And s.hisal
And e.deptno = 30

==========
自连接
----------
Select e.ename As 雇员, p.ename As 管理员
From emp e,emp p
Where e.mgr = p.empno
And e.deptno = 30

==========
合并查询
==========
union --- 两个集合的并集，去掉重复行，按第一列结构排序
----------
Select empno, ename, mgr From emp
Where deptno = 30
Union
Select empno, ename, mgr From emp
Where job = 'MANAGER'
==========
union all --- 两个集合并集，不去重复行，不排序
----------
Select empno, ename, mgr From emp
Where deptno = 30
Union All
Select empno, ename, mgr From emp
Where job = 'MANAGER'

==========
Intersect --- 只会显示同时存在两个集合中的数据
----------
Select empno, ename, mgr From emp
Where deptno = 30
Intersect
Select empno, ename, mgr From emp
Where job = 'MANAGER'

==========
minus --- 在一个集合存在，在第二个集合不存在的数据，按第一个排序
----------
Select empno, ename, mgr From emp
Where deptno = 30
Minus
Select empno, ename, mgr From emp
Where job = 'MANAGER'

==========
合并查询中，只能有一个 order by 子句。在这个子句中使用列名或
```

第一个查询的别名。

```
-----------------------------------
Select empno, ename 雇员, mgr From emp
Where deptno = 30
Minus
Select empno, ename 雇员, mgr From emp
Where job = 'MANAGER'
Order By 雇员
```

==========

==========

单行子查询

```
----------------------------
Select ename, deptno, sal From emp
Where sal = (Select Max(sal) From emp);
```

==========

多行子查询 --- where子句中使用多行子查询，必须使用多行运算符，（in,notin,exists,not exists,all,any）

```
-------------------------------------------------
Select ename, deptno, sal, job From emp
Where job In ( Select Distinct job From emp Where deptno = 20 )

Select e.ename, e.job, e.sal
 From emp e
 Where sal > All ( Select sal From emp Where emp.deptno = 20)

Select e.ename, e.job, e.sal
 From emp e
 Where sal > Any ( Select sal From emp Where emp.deptno = 20)
```

==========

相关子查询

```
----------------------------
Select deptno,
( Select Max(sal) From emp b Where b.deptno = a.deptno ) maxsal
From emp a
Order By deptno
----------------------------
Select ename, deptno, sal, job From emp
Where Exists
(
Select 'x' From dept
Where dept.deptno = emp.deptno And dept.loc = 'NEW YORK'
)
```

==========

标量子查询 --- 显示每个部门的最高工资员工信息

```
 ----------------------
 select distinct deptno,
 (select max(sal) from emp b where b.deptno = a.deptno) maxsal
 from emp a
 order by deptno;

==========================================================
 多列子查询 --- 显示与smith部门和岗位完全相同的所有雇员信息。
 ----------------------
 Select ename, deptno, sal, job From emp
 Where (deptno, job) =
 (Select deptno, job From emp Where ename = 'SMITH')
 ----------------------
 显示岗位或者管理员匹配于部门编号为20的所有雇员信息.
 ----------------------
 Select ename, deptno, sal, job, mgr From emp
 Where job In ( Select job From emp Where deptno = 20 )
 Or mgr In ( Select mgr From emp Where deptno = 20 )
 Order By deptno

==========================================================

==========================================================
 DDL 中使用子查询
 ==========================================================
 create table 语句中的子查询
 --------------------------
 create table dept1 (deptno, dname, loc) as
 select deptno, dname, loc from dept;

create table emp1 as
 select * from emp;
 ==========================================================
 create view 中使用子查询
 ----------------------
 create or replace view dept_20 as
 select * from emp1 where deptno = 20 order by empno;

==========================================================
 DML 中使用子查询
 ==========================================================
 update 语句中使用子查询
 ----------------------
 update emp1 set (sal,comm) =
 ( select sal, comm from emp1 where ename = 'WARD' )
 where job = ( select job from emp1 where ename= 'WARD' )

==========================================================
 delete 语句中使用子查询
 ----------------------
```

```
delete from emp1
where deptno = (select deptno from dept1 where dname = 'ACCOUNTING' )
```

==========

insert 语句中使用子查询

---

```
insert into emp1
select * from emp
where deptno = (select deptno from where dname = 'ACCOUNTING')
```

==========

==========

基础查询分类

==========

基本查询 --- 所有列、指定列、where子句、order by子句

---

分组查询 --- 组处理函数、group by子句、having子句

---

连接查询 --- 相等连接、不等连接、自我连接

==========

==========

合并查询 --- UNION，UNION ALL，INTERSECT，MINUS

---

子查询 --- 单行、多行、相关、标量、多列、DDL中、DML中

==========

==========

1。order by 子句必须放在最后。

2。组处理函数只能出现在选择列表、order by子句、having子句中，
不能出现在where子句和group by子句中。

3。在选择列表中包含的列、表达式，则一定要出现在group by子句中。

4。where子句中可以使用单行子查询，可以使用单行运算符。
（ =，>，<，>=，<=，<> ）

5。where子句中可以使用多行子查询，可以使用多行运算符。
（ in，not in，exists，not exists，all，any ）

==========

==========

==========

基本的Sql编写注意事项

---

1。尽量少用IN操作符，基本上所有的IN操作符都可以用EXISTS代替。

2。不用NOT IN操作符，可以用NOT EXISTS或者外连接+替代。

3。Oracle在执行IN子查询时，首先执行子查询，将查询结果放入临时表再执行主查询。
而EXIST则是首先检查主查询，然后运行子查询直到找到第一个匹配项。
NOT EXISTS 比 NOT IN 效率稍高。但具体在选择IN或EXIST操作时，
要根据主子表数据量大小来具体考虑。

4。不用“<>”或者“!=”操作符。对不等于操作符的处理会造成全表扫描，
可以用“<” or “>”代替。

5。Where子句中出现IS NULL或者IS NOT NULL时，Oracle会停止使用索引而执行全表扫描。
可以考虑在设计表时，对索引列设置为NOT NULL。这样就可以用其他操作来取代判断NULL的操作。

6。当通配符“%”或者“_”作为查询字符串的第一个字符时，索引不会被使用。

7。对于有连接的列“||”，最后一个连接列索引会无效。尽量避免连接，
可以分开连接或者使用不作用在列上的函数替代。

8。如果索引不是基于函数的，那么当在Where子句中对索引列使用函数时，索引不再起作用。

9。Where子句中避免在索引列上使用计算，否则将导致索引失效而进行全表扫描。

10。对数据类型不同的列进行比较时，会使索引失效。

11。用“>=”替代“>”。

12。UNION操作符会对结果进行筛选，消除重复，数据量大的情况下可能会引起磁盘排序。
如果不需要删除重复记录，应该使用UNION ALL。

13。Oracle从下到上处理Where子句中多个查询条件，所以表连接语句应写在其他Where条件前，
可以过滤掉最大数量记录的条件必须写在Where子句的末尾。

14。Oracle从右到左处理From子句中的表名，所以在From子句中包含多个表的情况下，
将记录最少的表放在最后。

15。Order By语句中的非索引列会降低性能，可以通过添加索引的方式处理。
严格控制在Order By语句中使用表达式。

16。不同区域出现的相同的Sql语句，要保证查询字符完全相同，以利用SGA共享池，
防止相同的Sql语句被多次分析。

17。当在Sql语句中连接多个表时，使用表的别名，并将之作为每列的前缀。这样可以减少解析时间。

==========

我们可以总结一下可能引起全表扫描的操作：

---

1。在索引列上使用NOT或者“<>”;

2。对索引列使用函数或者计算;

3。NOT IN操作;

4。通配符位于查询字符串的第一个字符;

5。IS NULL或者IS NOT NULL;

# oracle 数据库备份和还原

星期五, 十二月 28, 2018　　2:58 下午

## Oracle数据库备份与恢复的三种方法

发布时间：2006.10.20 05:10　　来源：chinaunix　　作者：renxiao2003

Oracle数据库有三种标准的备份方法，它们分别是导出/导入（EXP/IMP）、热备份和冷备份。导出备件是一种逻辑备份，冷备份和热备份是物理备份。
一、 导出/导入（Export/Import）
利用Export可将数据从数据库中提取出来，利用Import则可将提取出来的数据送回到Oracle数据库中去。
1、 简单导出数据（Export）和导入数据（Import）
Oracle支持三种方式类型的输出：
（1）、表方式（T方式），将指定表的数据导出。
（2）、用户方式（U方式），将指定用户的所有对象及数据导出。
（3）、全库方式（Full方式），瘵数据库中的所有对象导出。
数据导入（Import）的过程是数据导出（Export）的逆过程，分别将数据文件导入数据库和将数据库数据导出到数据文件。
2、 增量导出/导入
增量导出是一种常用的数据备份方法，它只能对整个数据库来实施，并且必须作为SYSTEM来导出。在进行此种导出时，系统不要求回答任何问题。导出文件名缺省为export.dmp，如果不希望自己的输出文件定名为export.dmp，必须在命令行中指出要用的文件名。
增量导出包括三种类型：
（1）、“完全”增量导出（Complete）
即备份三个数据库，比如：

```
exp system/manager inctype=complete file=
040731.dmp
```

（2）、“增量型”增量导出
备份上一次备份后改变的数据，比如：

```
exp system/manager inctype=incremental file=
040731.dmp
```

（3）、“累积型”增量导出
累计型导出方式是导出自上次“完全”导出之后数据库中变化了的信息。比如：

```
exp system/manager inctype=cumulative file=
040731.dmp
```

数据库管理员可以排定一个备份日程表，用数据导出的三个不同方式合理高效的完成。
比如数据库的被封任务可以做如下安排：
星期一：完全备份（A）
星期二：增量导出（B）

星期三：增量导出（C）
星期四：增量导出（D）
星期五：累计导出（E）
星期六：增量导出（F）
星期日：增量导出（G）
如果在星期日，数据库遭到意外破坏，数据库管理员可按一下步骤来回复数据库：
第一步：用命令CREATE DATABASE重新生成数据库结构；
第二步：创建一个足够大的附加回滚。
第三步：完全增量导入A：

```
imp system/manager inctype=RESTORE FULL=y FILE=A
```

第四步：累计增量导入E：

```
imp system/manager inctype=RESTORE FULL=Y FILE=E
```

第五步：最近增量导入F：

```
imp system/manager inctype=RESTORE FULL=Y FILE=F
```

二、 冷备份

冷备份发生在数据库已经正常关闭的情况下，当正常关闭时会提供给我们一个完整的数据库。冷备份时将关键性文件拷贝到另外的位置的一种说法。对于备份Oracle信息而言，冷备份时最快和最安全的方法。冷备份的优点是：

１、 是非常快速的备份方法（只需拷文件）

２、 容易归档（简单拷贝即可）

３、 容易恢复到某个时间点上（只需将文件再拷贝回去）

４、 能与归档方法相结合，做数据库“最佳状态”的恢复。

５、 低度维护，高度安全。

但冷备份也有如下不足：

１、 单独使用时，只能提供到“某一时间点上”的恢复。

２、 再实施备份的全过程中，数据库必须要作备份而不能作其他工作。也就是说，在冷备份过程中，数据库必须是关闭状态。

３、 若磁盘空间有限，只能拷贝到磁带等其他外部存储设备上，速度会很慢。

４、 不能按表或按用户恢复。

如果可能的话（主要看效率），应将信息备份到磁盘上，然后启动数据库（使用户可以工作）并将备份的信息拷贝到磁带上（拷贝的同时，数据库也可以工作）。冷备份

中必须拷贝的文件包括：

１、 所有数据文件

２、 所有控制文件

３、 所有联机REDO LOG文件

４、 Init.ora文件（可选）

值得注意的使冷备份必须在数据库关闭的情况下进行，当数据库处于打开状态时，执行数据库文件系统备份是无效的。

下面是作冷备份的完整例子。

（1） 关闭数据库

```
sqlplus /nolog
            sql>;connect /as sysdba
            sql>;shutdown normal;
```

（2） 用拷贝命令备份全部的时间文件、重做日志文件、控制文件、初始化参数文件

```
sql>;cp <file>; <backup directory>;
```

（3） 重启Oracle数据库

```
sql>;startup
```

三、 热备份

热备份是在数据库运行的情况下，采用archivelog mode方式备份数据库的方法。所以，如果你有昨天夜里的一个冷备份而且又有今天的热备份文件，在发生问题时，就可以利用这些资料恢复更多的信息。热备份要求数据库在Archivelog方式下操作，并需要大量的档案空间。一旦数据库运行在archivelog状态下，就可以做备份了。热备份的命令文件由三部分组成：

1. 数据文件一个表空间一个表空间的备份。

（1） 设置表空间为备份状态

（2） 备份表空间的数据文件

（3） 回复表空间为正常状态

2. 备份归档log文件

（1） 临时停止归档进程

（2） log下那些在archive rede log目标目录中的文件

（3） 重新启动archive进程

（4） 备份归档的redo log文件

3. 用alter database bachup controlfile命令来备份控制文件

热备份的优点是：

1. 可在表空间或数据库文件级备份，备份的时间短。
2. 备份时数据库仍可使用。
3. 可达到秒级恢复（恢复到某一时间点上）。
4. 可对几乎所有数据库实体做恢复
5. 恢复是快速的，在大多数情况下爱数据库仍工作时恢复。

热备份的不足是：

1. 不能出错，否则后果严重
2. 若热备份不成功，所得结果不可用于时间点的恢复
3. 因难于维护，所以要特别仔细小心，不允许“以失败告终”。

export 有四种备份方式：完全，表空间，用户，表

exp [user]/[passwd]@[servername] file=文件路径 log=日志路径

例如：exp system/manager@10g file=d:\expdata.dmp log=d:\expdata.log full=y

**如何查看oracle用户权限**

1. oracle用户查看自己的权限和角色
   select * from user_tab_privs;
   select * from user_role_privs;

2. sys用户查看任一用户的权限和角色
   select * from dba_tab_privs;
   select * from dba_role_privs;

# win7安装scrapy_Scrapy1.5中文文档_Scrapy

星期四, 十二月 27, 2018　　3:24 下午

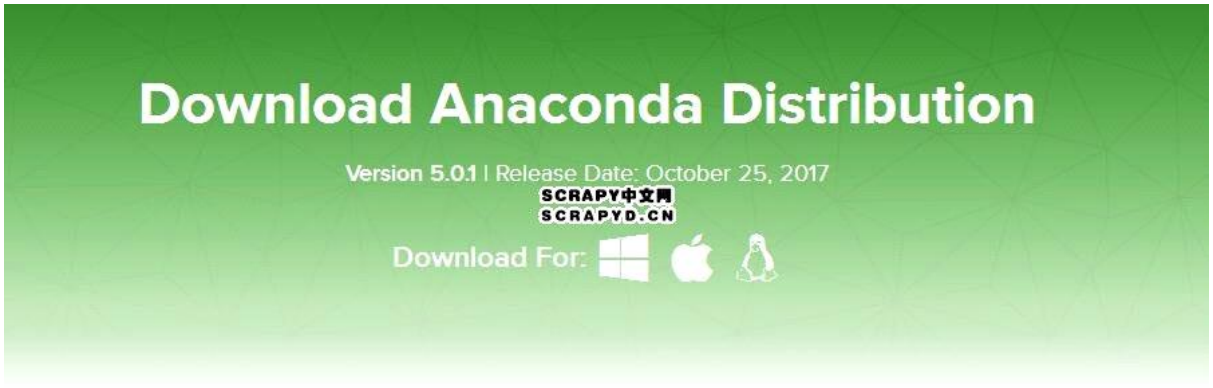
**软件环境：**

win7（64位）
Anaconda3-5.0.1-Windows-x86_64（点击下载）

**一、安装Anaconda**

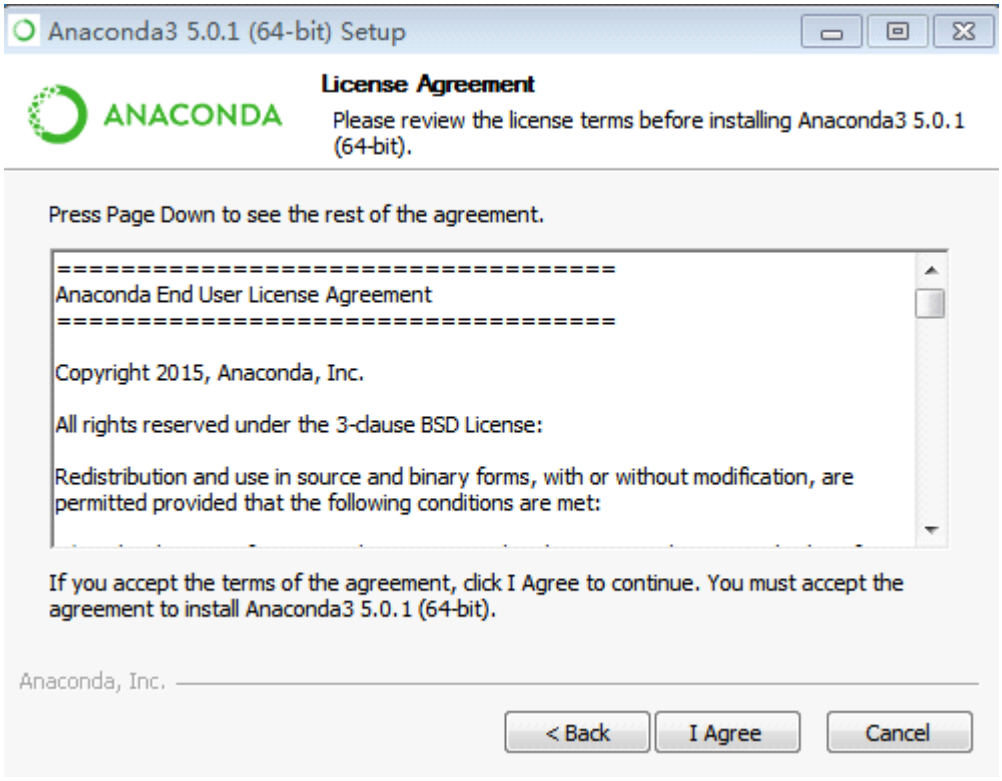
**这里的话简单介绍一下anaconda的下载，下载地址为：**
https://www.anaconda.com/download/

**如果官网无法打开，请移步这里：下载Anaconda**

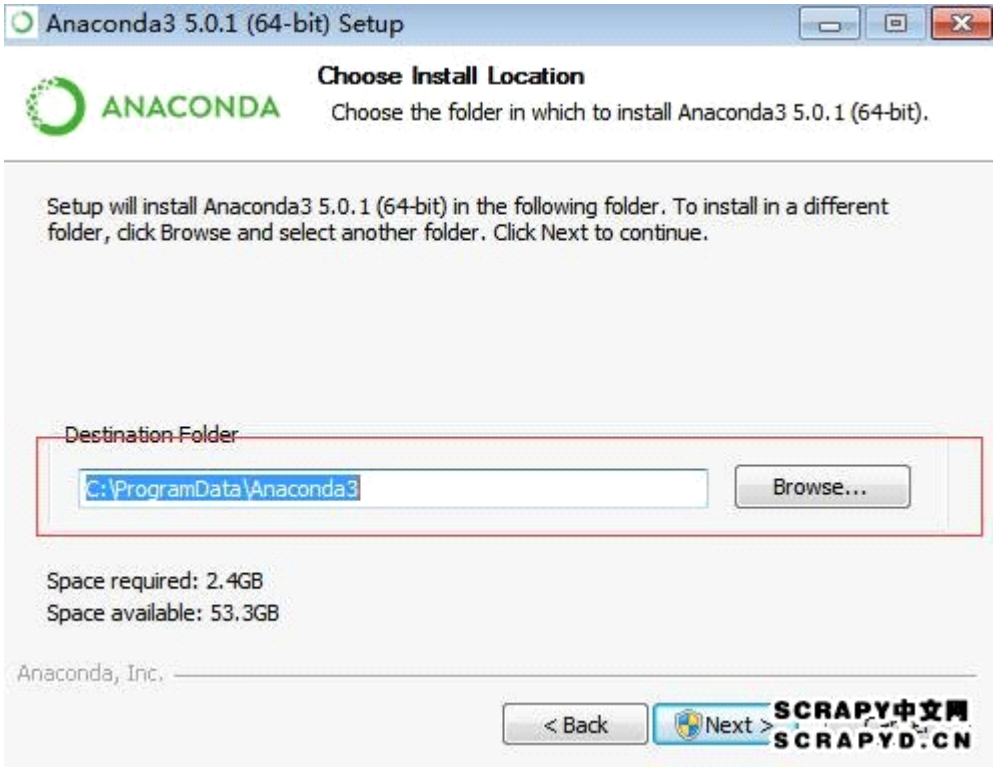



里面有不同的平台，不同的版本，根据您的平台进行下载即可！这里的话建议大家下载Anaconda全包，而不是Miniconda，因为后者虽然小巧，但会出现一些问题，如果你怕填坑就请下载Anaconda，500M左右，这里再说一点，python是不用安装的，当然装了也没关系！下载完成之后开始安装：

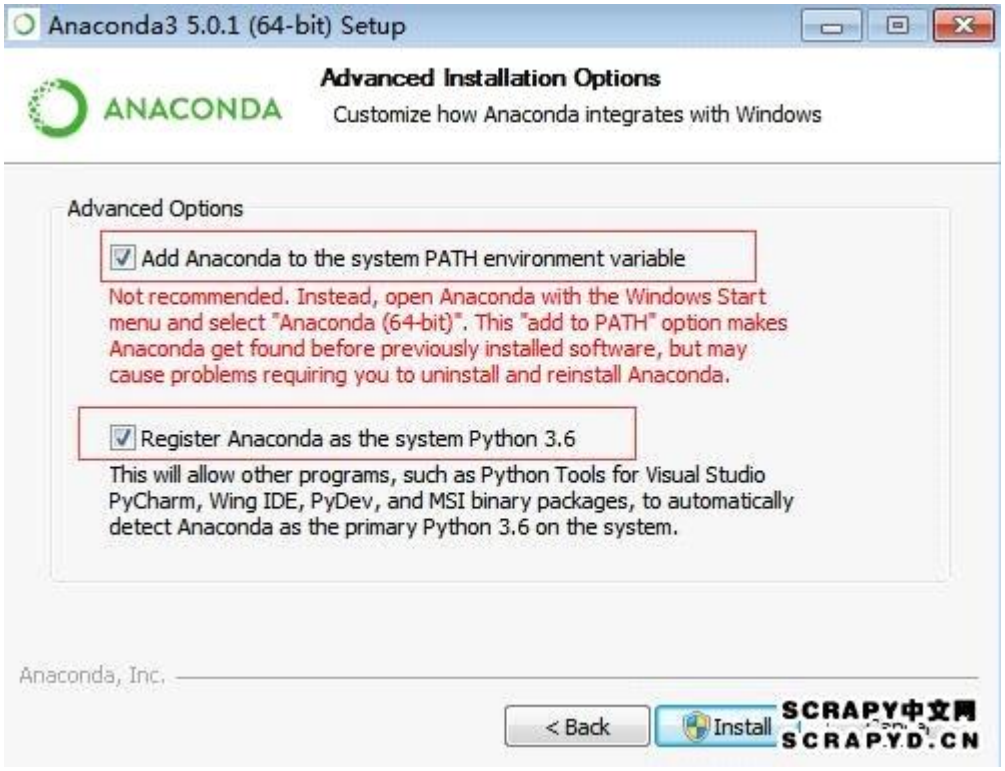

点击下一步



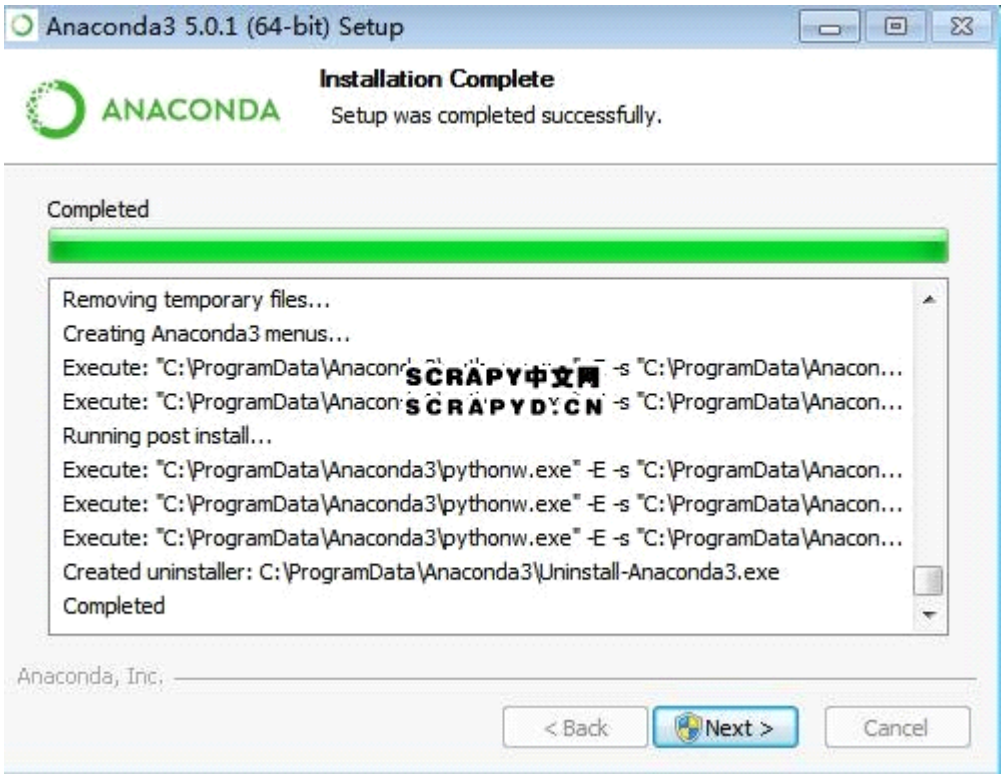

点击：I Agree（同意）




选择安装目录，然后Next，这里尽量不要选择带空格的目录

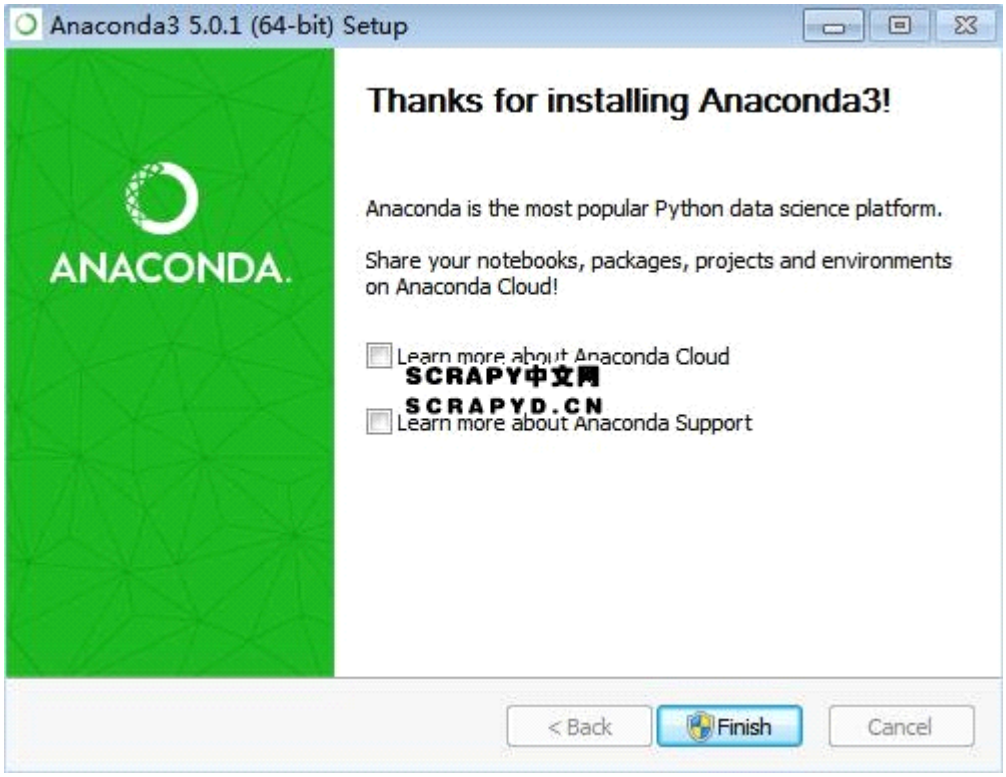



这一步一定两个都勾起了，上面是表示加入环境变量，勾了后期就不用设了，省得麻烦，下面是默认安装python3.6为默认的python解释器，勾选之后点击：Install，进行安装！安装过程有点缓慢，请大家耐心等待哈！





ok，大功告成，点击下一步



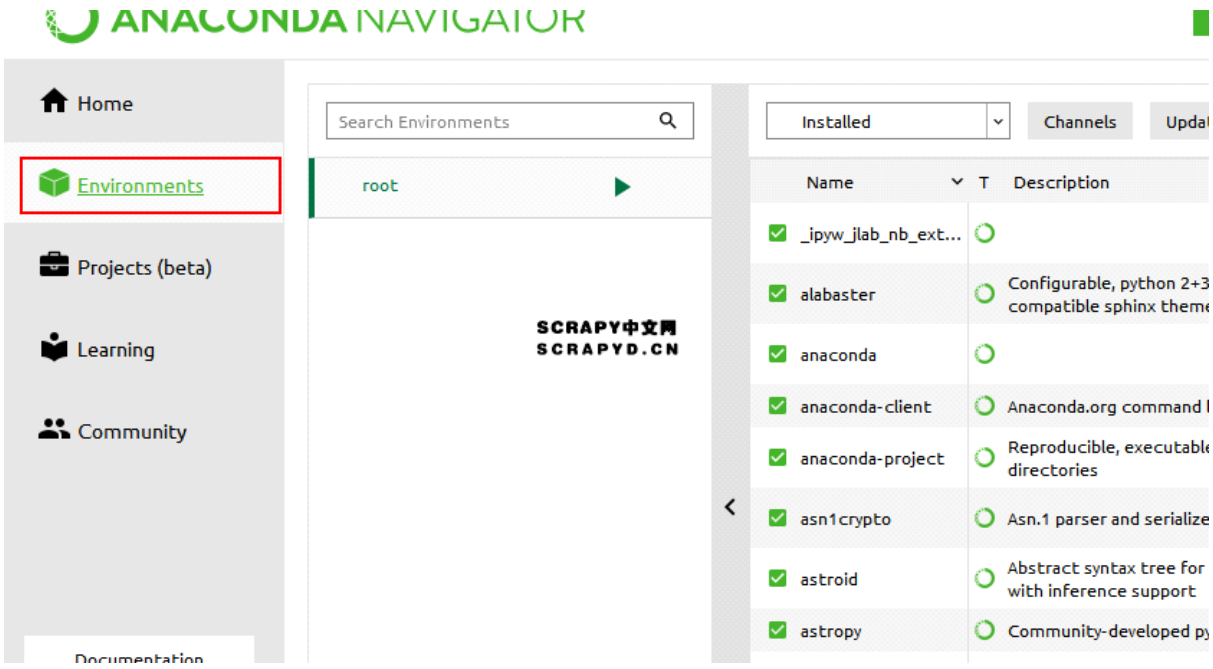


点击完成（Finish），那就安装完成了，接下来我们通过Anaconda安装scrapy

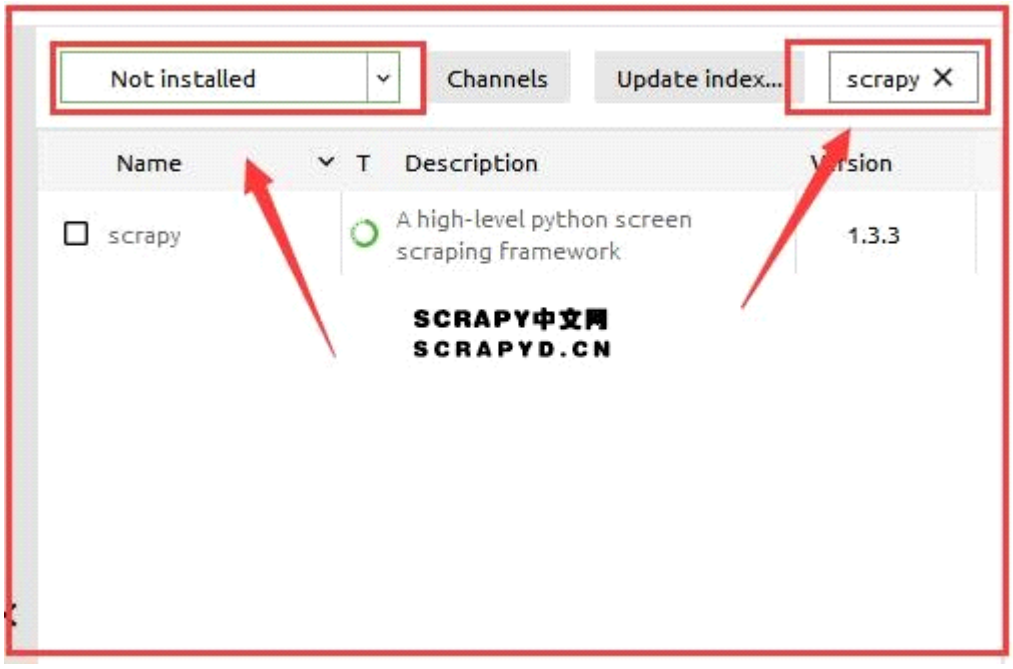
二、安装scrapy

点击开始菜单中的 Anaconda Navigator，启动Anaconda，稍等片刻，正在启动


启动后点击左侧：Environments

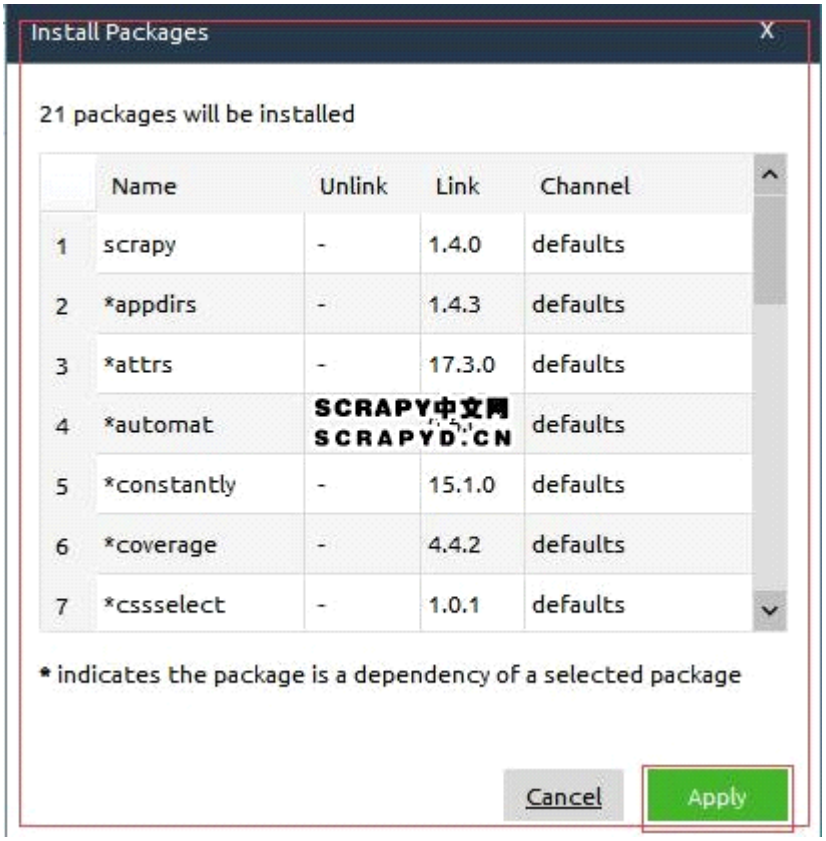


然后选择not install 然后在输入框中输入scrapy

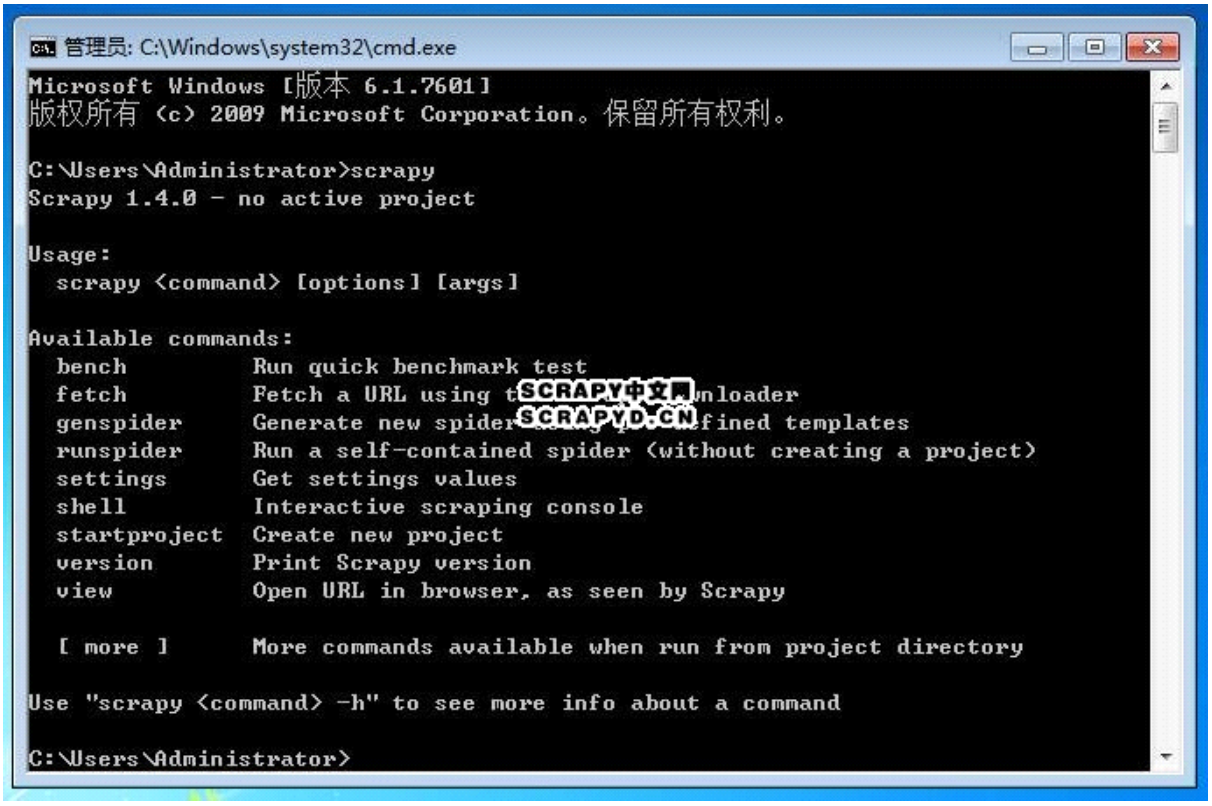



下面就出现了scrapy包，点击选中scrapy，然后点击下面Apply，进行相应安装


接下来会跳出scrapy的包依赖，选择Apply，即可




稍等片刻，即可安装完成，基本整个过程是0出错，非常顺畅，安装完成后在命令行输入：scrapy，进行测试，如果出现以下显示，恭喜你，scrapy已经成功安装！



如果你觉得文档看着不过瘾，你可以看视频教程呀，惊不惊喜、意不意外：《【视频】scrapy安装视频：win7+Anaconda安装scrapy1.4》

你妹，怎么现在才说……
应胃，我想锻炼你的阅读能力呀……
安装好之后，继续吧，下面的scrapy中文文档还很新鲜，快尝尝！

查看


scrapy

scrapy实战、scrapy视频教程

# Spring中配置DataSource的六种方式

星期三, 十二月 26, 2018 10:14 下午

**第一种：**beans.xml

**第二种：**beans.xml

```
<bean id="mappings"
```

在src文件夹里新建一个jdbc.properties文件，里面的内容为如下：

Xml代码

```
1. jdbc.driverClassName=com.microsoft.sqlserver.jdbc.SQLServerDriver
2. jdbcjdbc.url=jdbc:sqlserver://localhost:1433;DatabaseName=spring
3. jdbc.username=sa
4. jdbc.password=********
```

**第三种：**

beans.xml

其中第二种与第三种类似，只是指定配置文件的方法不一样。

第四种：

beans.xml

**第五种：**

Xml代码

```
1. beans.xml
2. <bean id="myDataSource" class="org.apache.commons.dbcp.BasicDataSource"
3.     destroy-method="close"
4.     p:driverClassName="com.microsoft.sqlserver.jdbc.SQLServerDriver"
5.     p:url="jdbc:sqlserver://localhost:1433;DatabaseName=spring"
6.     p:username="sa"
7.     p:password="********"/>
```

再加上命名空间：

Xml代码

```
1. xmlns:p="http://www.springframework.org/schema/p"
```

性能参数要根据实际情况测试得来的数据确定如何配置。
转自：http://www.cppblog.com/fenglin/articles/130494.html

第六种，在网上看到

Spring中提供了一种简便的方式就是context:property-placeholder/元素
只需要在spring的配置文件里添加一句

Xml代码

```
1. <context:property-placeholder location="classpath:jdbc.properties"/>
```

即可，这里location值为参数配置文件的位置，参数配置文件通常放在src目录下，而参数配置文件的格式跟java通用的参数配置文件相同，即键值对的形式，例如：
#jdbc配置

配置文件代码

```
1. test.jdbc.driverClassName=com.mysql.jdbc.Driver
2. test.jdbc.url=jdbc:mysql://localhost:3306/test
3. test.jdbc.username=root
4. test.jdbc.password=root
```

行内#号后面部分为注释
应用：
1.这样一来就可以为spring配置的bean的属性设置值了，比如spring有一个jdbc数据源的类DriverManagerDataSource
在配置文件里这么定义bean：

Java代码

```
1. <bean id="testDataSource" class="org.springframework.jdbc.datasource.DriverManagerDataSource">
2.     <property name="driverClassName" value="${test.jdbc.driverClassName}"/>
3.     <property name="url" value="${test.jdbc.url}"/>
4.     <property name="username" value="${test.jdbc.username}"/>
5.     <property name="password" value="${test.jdbc.password}"/>
6. </bean>
```

注意配置文件中

```
<bean id="accountDao" class="my.dao.impl.AccountDaoImpl">
     <property name="dataSource" ref="dataSource"/>
</bean>
```

2.甚至可以将${ }这种形式的变量用在spring提供的注解当中，为注解的属性提供值
外在化应用参数的配置

在开发企业应用期间，或者在将企业应用部署到生产环境时，应用依赖的很多参数信息往往需要调整，比如LDAP连接、RDBMS JDBC连接信息。对这类信息进行外在化管理显得格外重要。PropertyPlaceholderConfigurer和PropertyOverrideConfigurer对象，它们正是担负着外在化配置应用参数的重任。

<context:property-placeholder/>元素
PropertyPlaceholderConfigurer实现了BeanFactoryPostProcessor接口，它能够对<bean/>中的属性值进行外在化管理。开发者可以提供单独的属性文件来管理相关属性。比如，存在如下属性文件，摘自userinfo.properties。

Properties代码

```
1. db.username=scott
2. db.password=tiger
```

如下内容摘自propertyplaceholderconfigurer.xml。正常情况下，在userInfo的定义中不会出现${db.username}、${db.password}等类似信息，这里采用PropertyPlaceholderConfigurer管理username和password属性的取值。DI容器实例化userInfo前，PropertyPlaceholderConfigurer会修改userInfo的元数据信息（<bean/>定义），它会用userinfo.properties中db.username对应的scott值替换${db.username}、db.password对应的tiger值替换${db.password}。最终，DI容器在实例化userInfo时，UserInfo便会得到新的属性值，而不是${db.username}、${db.password}等类似信息。

通过运行并分析PropertyPlaceholderConfigurerDemo示例应用，开发者能够深入理解PropertyPlaceholderConfigurer。为简化PropertyPlaceholderConfigurer的使用，Spring提供了<context:property-placeholder/>元素。下面给出了配置示例，启用它后，开发者便不用配置PropertyPlaceholderConfigurer对象了。

<context:property-placeholder location="userinfo.properties"/>
PropertyPlaceholderConfigurer内置的功能非常丰富，如果它未找到${xxx}中定义的xxx键，它还会去JVM系统属性（System.getProperty()）和环境变量（System.getenv()）中寻找。通过启用systemPropertiesMode和searchSystemEnvironment属性，开发者能够控制这一行为。

# 整合ssh的时候出现空指针
# java.lang.NullPointerException

星期四, 十二月 27, 2018　8:55 上午

## HTTP Status 500 -

**type** Exception report

**message**

**description** The server encountered an internal error () that prevented it from fulfilling this request.

**exception**

```
java.lang.NullPointerException
	com.dragon.action.IndexAction.execute(IndexAction.java:45)
	sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
	sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:39)
	sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:25)
	java.lang.reflect.Method.invoke(Method.java:597)
	com.opensymphony.xwork2.DefaultActionInvocation.invokeAction(DefaultActionInvocation.java:441)
	com.opensymphony.xwork2.DefaultActionInvocation.invokeActionOnly(DefaultActionInvocation.java:280)
	com.opensymphony.xwork2.DefaultActionInvocation.invoke(DefaultActionInvocation.java:243)
	com.opensymphony.xwork2.interceptor.DefaultWorkflowInterceptor.doIntercept(DefaultWorkflowInterceptor.java:165)
	com.opensymphony.xwork2.interceptor.MethodFilterInterceptor.intercept(MethodFilterInterceptor.java:87)
	com.opensymphony.xwork2.DefaultActionInvocation.invoke(DefaultActionInvocation.java:237)
	com.opensymphony.xwork2.validator.ValidationInterceptor.doIntercept(ValidationInterceptor.java:252)
	org.apache.struts2.interceptor.validation.AnnotationValidationInterceptor.doIntercept(AnnotationValidationInterceptor.java:68)
	com.opensymphony.xwork2.interceptor.MethodFilterInterceptor.intercept(MethodFilterInterceptor.java:87)
	com.opensymphony.xwork2.DefaultActionInvocation.invoke(DefaultActionInvocation.java:237)
	com.opensymphony.xwork2.interceptor.ConversionErrorInterceptor.intercept(ConversionErrorInterceptor.java:122)
	com.opensymphony.xwork2.DefaultActionInvocation.invoke(DefaultActionInvocation.java:237)
	com.opensymphony.xwork2.interceptor.ParametersInterceptor.doIntercept(ParametersInterceptor.java:195)
```

com.opensymphony.xwork2.interceptor.MethodFilterInterceptor.intercept(MethodFilterInterceptor.java:87)
com.opensymphony.xwork2.DefaultActionInvocation.invoke(DefaultActionInvocation.java:237)
com.opensymphony.xwork2.interceptor.ParametersInterceptor.doIntercept(ParametersInterceptor.java:195)
com.opensymphony.xwork2.interceptor.MethodFilterInterceptor.intercept(MethodFilterInterceptor.java:87)
com.opensymphony.xwork2.DefaultActionInvocation.invoke(DefaultActionInvocation.java:237)
com.opensymphony.xwork2.interceptor.StaticParametersInterceptor.intercept(StaticParametersInterceptor.java:179)
com.opensymphony.xwork2.DefaultActionInvocation.invoke(DefaultActionInvocation.java:237)
org.apache.struts2.interceptor.MultiselectInterceptor.intercept(MultiselectInterceptor.java:75)
com.opensymphony.xwork2.DefaultActionInvocation.invoke(DefaultActionInvocation.java:237)
org.apache.struts2.interceptor.CheckboxInterceptor.intercept(CheckboxInterceptor.java:94)
com.opensymphony.xwork2.DefaultActionInvocation.invoke(DefaultActionInvocation.java:237)
org.apache.struts2.interceptor.FileUploadInterceptor.intercept(FileUploadInterceptor.java:235)
com.opensymphony.xwork2.DefaultActionInvocation.invoke(DefaultActionInvocation.java:237)
com.opensymphony.xwork2.interceptor.ModelDrivenInterceptor.intercept(ModelDrivenInterceptor.java:89)
com.opensymphony.xwork2.DefaultActionInvocation.invoke(DefaultActionInvocation.java:237)
com.opensymphony.xwork2.interceptor.ScopedModelDrivenInterceptor.intercept(ScopedModelDrivenInterceptor.java:130)
com.opensymphony.xwork2.DefaultActionInvocation.invoke(DefaultActionInvocation.java:237)
org.apache.struts2.interceptor.debugging.DebuggingInterceptor.intercept(DebuggingInterceptor.java:267)
com.opensymphony.xwork2.DefaultActionInvocation.invoke(DefaultActionInvocation.java:237)
com.opensymphony.xwork2.interceptor.ChainingInterceptor.intercept(ChainingInterceptor.java:126)
com.opensymphony.xwork2.DefaultActionInvocation.invoke(DefaultActionInvocation.java:237)
com.opensymphony.xwork2.interceptor.PrepareInterceptor.doIntercept(PrepareInterceptor.java:138)
com.opensymphony.xwork2.interceptor.MethodFilterInterceptor.intercept(MethodFilterInterceptor.java:87)
com.opensymphony.xwork2.DefaultActionInvocation.invoke(DefaultActionInvocation.java:237)
com.opensymphony.xwork2.interceptor.I18nInterceptor.intercept(I18nInt

erceptor.java:165)
	com.opensymphony.xwork2.DefaultActionInvocation.invoke(DefaultActionInvocation.java:237)
	org.apache.struts2.interceptor.ServletConfigInterceptor.intercept(ServletConfigInterceptor.java:164)
	com.opensymphony.xwork2.DefaultActionInvocation.invoke(DefaultActionInvocation.java:237)
	com.opensymphony.xwork2.interceptor.AliasInterceptor.intercept(AliasInterceptor.java:179)
	com.opensymphony.xwork2.DefaultActionInvocation.invoke(DefaultActionInvocation.java:237)
	com.opensymphony.xwork2.interceptor.ExceptionMappingInterceptor.intercept(ExceptionMappingInterceptor.java:176)
	com.opensymphony.xwork2.DefaultActionInvocation.invoke(DefaultActionInvocation.java:237)
	org.apache.struts2.impl.StrutsActionProxy.execute(StrutsActionProxy.java:52)
	org.apache.struts2.dispatcher.Dispatcher.serviceAction(Dispatcher.java:488)
	org.apache.struts2.dispatcher.ng.ExecuteOperations.executeAction(ExecuteOperations.java:77)
	org.apache.struts2.dispatcher.ng.filter.StrutsPrepareAndExecuteFilter.doFilter(StrutsPrepareAndExecuteFilter.java:91)

note The full stack trace of the root cause is available in the Apache Tomcat/6.0.35 logs.

以上是页面报错的信息但是在控制台没有报错 也进了action

之前我运行了好几次都是报的这个异常，进行了调试他只在action中调用了方法直接返回空指针 刚开始还以为是那块写错了

经过仔细的检查发现原来是 struts。xml中配置的问题

未修改之前是这样

```
<package name="default" namespace="/" extends="struts-default">
  <action name="index"
class="com.dragon.action.IndexAction" >
   <result name="success">index.jsp</result>
  </action>
 </package>
```

修改之后

```
<package name="default" namespace="/" extends="struts-default">
  <action name="index" class="indexAction" >
   <result name="success">index.jsp</result>
  </action>
```

</package>

仔细看了看哦原来是class的问题因为使用了ssh技术 需要用spring中定义的bean的id访问作为struts的class名

# Java从入门到精通——数据库篇Oracle 11g服务详解 - 夏至冬末 - 博客园

星期五, 十二月 28, 2018 3:30 下午

已剪辑自: http://www.cnblogs.com/iplus/p/4490427.html

装上Oracle之后大家都会感觉到我们的电脑慢了下来，如何提高计算机的速度呢？我们应该打开必要的服务，关闭没有用的服务。下面是Oracle服务的详解：

**Oracle ORCL VSS Writer Service**：Oracle卷映射拷贝写入服务，VSS（Volume Shadow Copy Service）能够让存储基础设备（比如磁盘，阵列等）创建高保真的时间点映像，即映射拷贝（shadow copy）。它可以在多卷或者单个卷上创建映射拷贝，同时不会影响到系统的系统能。（非必须启动）

**OracleDBConsoleorcl**：Oracle数据库控制台服务，orcl是Oracle的实例标识，默认的实例为orcl。在运行Enterprise Manager（企业管理器OEM）的时候，需要启动这个服务。（非必须启动）

**OracleJobSchedulerORCL**：Oracle作业调度（定时器）服务，ORCL是Oracle实例标识。（非必须启动）

**OracleMTSRecoveryService**：服务端控制。该服务允许数据库充当一个微软事务服务器MTS、COM/COM+对象和分布式环境下的事务的资源管理器。（非必须启动）

**OracleOraDb11g_home1ClrAgent**：Oracle数据库.NET扩展服务的一部分。 （非必须启动）

**OracleOraDb11g_home1TNSListener**：监听器服务，服务只有在数据库需要远程访问的时候才需要。（非必须启动，下面会有详细详解）。

**OracleServiceORCL**：数据库服务(数据库实例)，是Oracle核心服务该服务，是数据库启动的基础， 只有该服务启动，Oracle数据库才能正常启动。(必须启动)

那么在开发的时候到底需要启动哪些服务呢？

对新手来说，要是只用Oracle自带的sql*plus的话，只要启动OracleServiceORCL即可，要是使用PL/SQL Developer等第三方工具的话，OracleOraDb11g_home1TNSListener服务也要开启。OracleDBConsoleorcl是进入基于web的EM必须开启的，其余服务很少用。

注：ORCL是数据库实例名，默认的数据库是ORCL，你可以创建其他的，即OracleService+数据库名。

资料来源：http://www.t4boys.com/oracle-service.html

Meet so Meet. C plusplus I-PLUS....

# Oracle 11g控制文件全部丢失从零开始重建控制文件_oracle_脚本之家

星期五, 十二月 28, 2018 3:31 下午

已剪辑自: https://www.jb51.net/article/109493.htm

**介绍**

控制文件（control file）是一个相当小的文件（最多能增长到64M左右），其中包含Oracle需要的其他文件的一个目录。参数文件告知实例控制文件的位置，控制文件则告知示例数据库和在线重做日志文件的位置。控制文件还告知了Oracle其他一些事情，如已发生检查点的有关信息、数据库名（必须和db_name参数匹配）、创建数据库的时间戳、归档重做日志的历史（有时这会让控制文件变大）、RMAN信息等。

控制文件应该通过硬件（RAID）多路保存，如果不支持镜像，则要通过Oracle多路保存。应该有不止一个副本，而且它们应该保存在不同的磁盘上，以防止万一出现磁盘故障而丢失控制文件。丢失控制文件并不是致命的，但是会使恢复变得困难很多。

如果丢失了所有的控制文件并且没有任何的备份，我们可以通过重建控制文件来打开数据库。其中，重建控制文件至少需要以下信息：

1.数据库名

2.字符集

3.数据文件名称

4.初始化参数，包括MAXLOGFILES、MAXLOGMEMBERS、MAXDATAFILES、MAXINSTANCES、MAXLOGHISTORY等；

**一、环境准备**

**数据库版本**

我们在Oracle11g中进行测试。

```
SQL>
SQL> select * from v$version;

BANNER
--------------------------------------------------------------------------------
```

```
Oracle Database 11g Enterprise Edition Release 11.2.0.3.0 - Production
PL/SQL Release 11.2.0.3.0 - Production
CORE 11.2.0.3.0 Production
TNS for Linux: Version 11.2.0.3.0 - Production
NLSRTL Version 11.2.0.3.0 - Production

SQL>
```

**删除控制文件**

1.通过查询control_files初始化参数，获取控制文件路径；

```
SQL>
SQL> show parameter control_files

NAME TYPE VALUE
------------------------------------ ----------- ------------------------------
----
control_files string    /u01/app/oracle/oradata/HOEGH/
       control01.ctl, /u01/app/oracle
       /oradata/HOEGH/control02.ctl
SQL>
```

2.然后，使用rm命令删除控制文件；

```
[oracle@HOEGH ~]$ rm /u01/app/oracle/oradata/HOEGH/control01.ctl
[oracle@HOEGH ~]$ rm /u01/app/oracle/oradata/HOEGH/control02.ctl
[oracle@HOEGH ~]$
```

3.此时，强制关闭数据库，然后重启数据库，报ORA-00205错误。需要注意的是，此时执行shutdown immediate命令，数据库无法正常关闭，只能关闭到mounted状态；需要使用shutdown abort命令强制关闭数据库。

```
SQL>
SQL> shutdown immediate
Database closed.
ORA-00210: cannot open the specified control file
ORA-00202: control file: \'/u01/app/oracle/oradata/HOEGH/control01.ctl\'
ORA-27041: unable to open file
Linux Error: 2: No such file or directory
Additional information: 3


SQL> select status from v$instance;

STATUS
------------
MOUNTED

SQL>
```

```
SQL> shutdown abort
ORACLE instance shut down.
SQL>



SQL>
SQL> startup
ORACLE instance started.

Total System Global Area 941600768 bytes
Fixed Size 1348860 bytes
Variable Size 515902212 bytes
Database Buffers 419430400 bytes
Redo Buffers 4919296 bytes
ORA-00205: error in identifying control file, check alert log for more info


SQL>
```

## 二、获取数据库名

首先生成文本格式的参数文件；

```
SQL>
SQL> create pfile from spfile;

File created.

SQL>
```

打开参数文件，查看db_name参数值，即为数据库名称。

```
[oracle@hoegh dbs]$ cat initHOEGH.ora
HOEGH.__db_cache_size=419430400
HOEGH.__java_pool_size=4194304
HOEGH.__large_pool_size=4194304
HOEGH.__oracle_base=\'/u01/app/oracle\'#ORACLE_BASE set from environment
HOEGH.__pga_aggregate_target=377487360
HOEGH.__sga_target=566231040
HOEGH.__shared_io_pool_size=0
HOEGH.__shared_pool_size=130023424
HOEGH.__streams_pool_size=0
*.audit_file_dest=\'/u01/app/oracle/admin/HOEGH/adump\'
*.audit_trail=\'db\'
*.compatible=\'11.2.0.0.0\'
*.control_files=\'/u01/app/oracle/oradata/HOEGH/control01.ctl\',
\'/u01/app/oracle/oradata/HOEGH/control02.ctl\'
*.db_block_size=8192
```

```
*.db_domain=\'\'
*.db_name=\'HOEGH\'
*.diagnostic_dest=\'/u01/app/oracle\'
*.dispatchers=\'(PROTOCOL=TCP) (SERVICE=HOEGHXDB)\'
*.memory_max_target=943718400
*.memory_target=943718400
*.open_cursors=300
*.processes=150
*.remote_login_passwordfile=\'EXCLUSIVE\'
*.undo_tablespace=\'UNDOTBS1\'
[oracle@hoegh dbs]$
```

## 三、启动到nomount状态，获取字符集

由于需要执行查询语句select userenv('language') from dual;来获取字符集，因此需要将数据库启动到nomount状态。

```
SQL>
SQL> startup nomount
ORACLE instance started.

Total System Global Area 941600768 bytes
Fixed Size 1348860 bytes
Variable Size 515902212 bytes
Database Buffers 419430400 bytes
Redo Buffers 4919296 bytes
SQL>
SQL> select userenv(\'language\') from dual;

USERENV(\'LANGUAGE\')
----------------------------------------------------
AMERICAN_AMERICA.US7ASCII

SQL>
SQL>
```

## 四、获取数据文件名称

通过ls命令获取数据文件列表。

```
[oracle@hoegh HOEGH]$ ls -lh
total 1.8G
-rw-r----- 1 oracle oinstall 314M May 30 11:07 example01.dbf
-rw-r----- 1 oracle oinstall 51M May 30 11:07 redo01.log
-rw-r----- 1 oracle oinstall 51M May 30 11:07 redo02.log
-rw-r----- 1 oracle oinstall 51M May 30 11:07 redo03.log
-rw-r----- 1 oracle oinstall 541M May 30 11:07 sysaux01.dbf
```

```
-rw-r----- 1 oracle oinstall 721M May 30 11:07 system01.dbf
-rw-r----- 1 oracle oinstall 30M Oct 13 2014 temp01.dbf
-rw-r----- 1 oracle oinstall 96M May 30 11:07 undotbs01.dbf
-rw-r----- 1 oracle oinstall 5.1M May 30 11:07 users01.dbf
[oracle@hoegh HOEGH]$
```

## 五、生成创建控制文件脚本

这样，创建控制文件所需的基本信息都已经有了，我们来生成创建控制文件脚本。

```
STARTUP NOMOUNT
CREATE CONTROLFILE REUSE DATABASE \"HOEGH\" NORESETLOGS ARCHIVELOG
 MAXLOGFILES 5
 MAXLOGMEMBERS 3
 MAXDATAFILES 100
 MAXINSTANCES 1
 MAXLOGHISTORY 226
LOGFILE
 GROUP 1 \'/u01/app/oracle/oradata/HOEGH/redo01.log\' SIZE 50M,
 GROUP 2 \'/u01/app/oracle/oradata/HOEGH/redo02.log\' SIZE 50M,
 GROUP 3 \'/u01/app/oracle/oradata/HOEGH/redo03.log\' SIZE 50M
DATAFILE
 \'/u01/app/oracle/oradata/HOEGH/system01.dbf\',
 \'/u01/app/oracle/oradata/HOEGH/sysaux01.dbf\',
 \'/u01/app/oracle/oradata/HOEGH/undotbs01.dbf\',
 \'/u01/app/oracle/oradata/HOEGH/users01.dbf\',
 \'/u01/app/oracle/oradata/HOEGH/example01.dbf\',
 \'/u01/app/oracle/oradata/HOEGH/temp01.dbf\'
CHARACTER SET US7ASCII
;
```

## 六、重建控制文件

需要注意的是，在执行上述创建脚本时会报错，系统提示临时文件不属于数据文件，如下所示：

```
SQL> @/u01/app/oracle/oradata/HOEGH/CreateControlFile.sql
ORA-01081: cannot start already-running ORACLE - shut it down first
CREATE CONTROLFILE REUSE DATABASE \"HOEGH\" NORESETLOGS ARCHIVELOG
*
ERROR at line 1:
ORA-01503: CREATE CONTROLFILE failed
ORA-01160: file is not a data file
ORA-01110: data file : \'/u01/app/oracle/oradata/HOEGH/temp01.dbf\'


SQL>
```

修改脚本并重新执行，重建控制文件后，数据库会打开到mount状态。

```
SQL>
SQL> @/u01/app/oracle/oradata/HOEGH/CreateControlFile.sql
ORACLE instance started.

Total System Global Area 941600768 bytes
Fixed Size 1348860 bytes
Variable Size 515902212 bytes
Database Buffers 419430400 bytes
Redo Buffers 4919296 bytes

Control file created.

SQL>
SQL> select status from v$instance;

STATUS
------------
MOUNTED

SQL>
```

## 七、打开数据库

在打开数据库时，会报错，提示system01数据文件需要执行介质恢复，我们执行recover database即可。

```
SQL>
SQL> alater database open;
SP2-0734: unknown command beginning \"alater dat...\" - rest of line
ignored.
SQL>
SQL> alter database open;
alter database open
*
ERROR at line 1:
ORA-01113: file 1 needs media recovery
ORA-01110: data file 1: \'/u01/app/oracle/oradata/HOEGH/system01.dbf\'


SQL>
SQL> recover database;
Media recovery complete.
SQL>
SQL> alter database open;

Database altered.

SQL>
```

```
SQL> select * from v$version;

BANNER
--------------------------------------------------------------------------------
Oracle Database 11g Enterprise Edition Release 11.2.0.3.0 - Production
PL/SQL Release 11.2.0.3.0 - Production
CORE 11.2.0.3.0 Production
TNS for Linux: Version 11.2.0.3.0 - Production
NLSRTL Version 11.2.0.3.0 - Production

SQL>
SQL> select tablespace_name from dba_tablespaces;


TABLESPACE_NAME
------------------------------
SYSTEM
SYSAUX
UNDOTBS1
TEMP
USERS
EXAMPLE


6 rows selected.


SQL>
```

**下面总结一下重建控制文件的步骤：**

1.获取数据库名；

2.获取字符集名；

3.获取数据文件名；

4.重建控制文件；

5.执行介质恢复；

6.打开数据库。

**总结**

以上就是这篇文章的全部内容了，希望本文的内容对大家的学习或者工作能带来一定的帮助，如果有疑问大家可以留言交流，谢谢大家对脚本之家的支持。