获取两个日期之间的天数

Calendar c1 = Calendar.getInstance();

c1.set(2005, 1, 1);

Date dateCreatedStop = new java.sql.Date(new java.util.Date().getTime());

Calendar c2 = Calendar.getInstance();

c2.setTime(dateCreatedStop);

long l1 = c1.getTimeInMillis();

long l2 = c2.getTimeInMillis();

// 计算天数

long days = (l2 - l1) / (24 \* 60 \* 60 \* 1000);

System.out.println(days);

子窗口获取父窗口对象

qryObj=window.opener.document.all.elecAuditOrder.selectedItem;

打开子窗口方式为：window.open();

打印功能实现

打印页面打开方式：window.open("printFeeTableGwDetail.jsp",'',"dialogWidth:760px;dialogHeight:680px;help:no");

打印功能实现最小代码：

<OBJECT ID="WebBrowser" WIDTH=0 HEIGHT=0

CLASSID="CLSID:8856F961-340A-11D0-A96B-00C04FD705A2">

</OBJECT>

<TABLE id="tbButton2" height="30" class="noprint" align="center">

<tr>

<td><input id="btnPrintView" value="打印预览" onClick="document.all.WebBrowser.ExecWB(7,1)" type="button" ></td>

<td><input id="btnPrint" value="打印" onClick="document.all.WebBrowser.ExecWB(6,6)" type="button"></td>

<td><input id="btnPrintSet" value="页面设置" onClick="document.all.WebBrowser.ExecWB(8,1)" type="button"></td>

</tr>

</TABLE>

<div id='printToWordOrExcel'>

</div>

子菜单增加：

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

insert into UOS\_PRIV (PRIV\_CODE,PRIV\_NAME,PRIV\_CLASS\_ID,COMMENTS,STATE)values ('maintenaFactoryInfo','维保厂家信息管理',1230,'维保厂家信息管理','10A');

insert into PB\_ACTION (ID,NAME,MODULE\_ID,PRIV\_CODE,ICON\_FILE\_NAME,STATE,STATE\_DATE,COMMENTS)values (301212,'维保厂家信息管理',741,'maintenaFactoryInfo','/IOMPROJ/images/icon/icon16.gif','10A',current timestamp,NULL);

insert into PB\_MENU (ID,URL\_STRING,SHOW\_MODEL,DISPLAY\_INDEX)values (301212,'/IOMPROJ/maintena/maintenaFactoryInfo.jsp','0',351);

insert into UOS\_PRIV\_CLASS (PRIV\_CLASS\_ID,PRIV\_CLASS\_NAME,PARENT\_ID,PATH\_CODE,COMMENTS)VALUES (1213,'批复',1190,'-1.1190.1213',null);

insert into PB\_MODULE(ID,PARENT\_ID,TYPE\_ID,PRIV\_CLASS\_ID,NAME,DISPLAY\_INDEX,PATH\_CODE,ICON\_FILE\_NAME,STATE,STATE\_DATE,COMMENTS,PATH\_NAME)VALUES (724,688,'001',1213,'批复',2,'-1.688.724','/IOMPROJ/images/icon/icon17.gif','10A',CURRENT TIMESTAMP,null,'IOM系统菜单/预算管理/批复');

------------------------

insert into UOS\_PRIV (PRIV\_CODE,PRIV\_NAME,PRIV\_CLASS\_ID,COMMENTS,STATE)values ('maintenaFactoryInfo','维保厂家信息管理',1230,'维保厂家信息管理','10A');

insert into PB\_ACTION (ID,NAME,MODULE\_ID,PRIV\_CODE,ICON\_FILE\_NAME,STATE,STATE\_DATE,COMMENTS)values (301212,'维保厂家信息管理',741,'maintenaFactoryInfo','/IOMPROJ/images/icon/icon16.gif','10A',current timestamp,NULL);

insert into PB\_MENU (ID,URL\_STRING,SHOW\_MODEL,DISPLAY\_INDEX)values (301212,'/IOMPROJ/maintena/maintenaFactoryInfo.jsp','0',351);

-------------------------

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

db2 将null转换成0或其他

coalesce(sum(D.REIMBURSE\_MONEY),0)

db2 对date数据取年月日

AND year(eu.READ\_DATE) = 2011

AND month(eu.READ\_DATE) = 12

js 字符串转数字

parseInt()

parseFloat()

转换成2位小数

parseFloat(10.353).toFixed(2)

db2 内容转换

(CASE W.DISPATCH\_TYPE WHEN 1 THEN '派到个人' WHEN 2 THEN '派到组织' ELSE '' END ) as dispatchTypeName

js去空格

// 去掉字符串的头空格（左空格）

function LTrim(str){

var i;

for(i=0;i<str.length; i++) {

if(str.charAt(i)!=" ") break;

}

str = str.substring(i,str.length);

return str;

}

// 去掉字符串的尾空格（右空格）

function RTrim(str){

var i;

for(i=str.length-1;i>=0;i--){

if(str.charAt(i)!=" ") break;

}

str = str.substring(0,i+1);

return str;

}

// 去掉字符串的头尾空格（左右空格）

function Trim(str){

return LTrim(RTrim(str));

}

js的为null转化

(list[i].netPointCode||'&nbsp;')

相对路径

../ 表示找到此文件的上一级

db2取结果集的第一条记录

select u.ENERGY\_CONSUME\_COMMON\_ID,e.NET\_POINT\_ID,u.READ\_DATE,u.ISAUTO from CP\_ENERGY\_CONSUME\_COMMON E left join cp\_ec\_elec\_use U on e.ENERGY\_CONSUME\_COMMON\_ID=u.ENERGY\_CONSUME\_COMMON\_ID

where e.NET\_POINT\_ID=16026 and year(u.READ\_DATE)=2012 and month(u.READ\_DATE)=6 order by u.READ\_DATE desc

FETCH FIRST 1 ROWS ONLY

js 保留两位小数

var a = 1.23445556;

a.toFixed(2);

db2 修改字段类型

ALTER TABLE a ALTER COLUMN x SET DATA TYPE INTEGER

js 设定日期，取年月日

tmpBeginDate = StringToDate(beginDate);

var beginDateStr = tmpBeginDate.get("yy")+"-"+tmpBeginDate.get("mm")+"-"+tmpBeginDate.get("dd")+" 08:30:00";

treelist增加记录

var node = document.all.tlFactoryDevice.createTreeNode();

node.factoryName = retObj.factoryName;

node.deviceModel = retObj.deviceModel;

node.factoryDeviceId = retObj.factoryDeviceId;

document.all.tlFactoryDevice.add(node);

js获取某年某月的天数

function getDays(year,month){

if(month == 12){

year = year+1;

var temp = new Date(year+"/1/0");

}else{

var temp = new Date(year+"/" +month +"/0");

}

return temp.getDate();

}

java String 转Date

DateFormat format = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");

Date date = null;

String str = null;

str = "2007-1-18";

date = format.parse(str);

js按钮可用性

document.getElementById('bt2').disabled=0 //可用

document.getElementById('bt2').disabled=true //不可用

js的eval()

它的功能是把对应的字符串解析成JS代码并运行

比如说你现在要运行一个可变的方法

function name1(){……}

function name2(){……}

var m="name1";

eval(m+'()');//运行name1();

m='name2';

eval(m+'()');//运行name2();

链接中运行javascript

<a href='javascript: window.open(

java的IO整理

String fileName="D:"+File.separator+"a.txt";

File f=new File（fileName）;

f.createNewFile();

f.listFiles();//列出文件夹下所有文件完整路径

//f.各种方法各种探索

OutputStream out =new FileOutputStream（f）;

String str="你好";

byte[] b=str.getBytes（）;

out.write（b）;//向文件写入字符串

out.close（）;

Writer out =new FileWriter（f,true）;//声明一个可追加内容的out

关于字节流和字符流的区别

实际上字节流在操作的时候本身是不会用到缓冲区的，是文件本身的直接操作的，但是字符流在操作的 时候下后是会用到缓冲区的，是通过缓冲区来操作文件的。

读者可以试着将上面的字节流和字符流的程序的最后一行关闭文件的代码注释掉，然后运行程序看看。你就会发现使用字节流的话，文件中已经存在内容，但是使用字符流的时候，文件中还是没有内容的，这个时候就要刷新缓冲区。

使用字节流好还是字符流好呢？FileOutputStream

答案是字节流。首先因为硬盘上的所有文件都是以字节的形式进行传输或者保存的，包括图片等内容。但是字符只是在内存中才会形成的，所以在开发中，字节流使用广泛。

js日期相差

var objDate1=new Date(2008,6,19,23,59,59);

var objDate2=new Date(2008,6,20,0,01,01);

var msSpan=objDate1.getTime()-objDate2.getTime();

alert("相差:"+msSpan/(24\*60\*60\*1000)+"天");

alert("相差:"+msSpan/(60\*60\*1000)+"小时");

alert("相差:"+msSpan/(60\*1000)+"分钟");

alert("相差:"+msSpan/(1000)+"秒");

对象数组的实例化

WorkOrderConfDto[] workOrderConfDtos = new WorkOrderConfDto[templist.size()];

workOrderConfDtos[0] = new WorkOrderConfDto();

workOrderConfDtos[0].setCreateDate(new Date());//没有第二句，这句会包nullpointer错误

dynamic方法的与或表示

ID:id,&&:STATE:state

中兴Date.js

var today = new Date();

today.add("dd", 7);//加7天，yy年，mm月，hh时，mi分，ss秒

中兴，不知道啥,随便可以运行一条sql

\_import("BSCommon"); //必须引入

function QueryConditionFun(){

\_extends(this, BSCommon, '"form1"'); //必须引入

this.clearParam();

this.addParam(2,"areaId", session1.area.areaId);//2为String,1为int

retObj = this.callRemoteFunctionQuery("canAreaSaveQueryCondition");

if(retObj && retObj.length>0 && retObj.length==1&&retObj[0].RET!=null){

QueryCondition = retObj[0].RET; //必须大写

}

}

select \* from SERVICE\_BEAN where SERVICE\_NAME='canAreaSaveQueryCondition'

select \* from QUERY\_BEAN where SERVICE\_ID in(320823)

js获取select的text

document.all.startProvince.options[document.all.startProvince.selectedIndex].text

js的Array在java里从map里取出装变成list

List list = (ArrayList) MapUtils.getObject(paramMap,"juxiangTree");

oracle字符串转时间比较

sqlBuf.append(" and wwoi.CREATE\_DATE <= to\_date('"+MapUtils.getString(quercondition, "createDate2")+"','yyyy-mm-dd hh24:mi:ss') ");

oracle的case的另一种方式

(CASE WHEN ojdo.RETURN\_DATE < SYSDATE THEN '是' else '否' END ) as a

js的String转Date再比较

var itemlimitDate = StringToDate(items[i].receiptDline);

var toDate = new Date();

if(itemlimitDate<toDate){

oracle的取第一条记录

select \* from (select \* from table order by a )C

where rownum=1

数据库sql查询，多行多列变成一行

MAX(case WHEN BUDGET\_SUBJECT\_ID= -1 then REIMBURSE\_MONEY end ) AS SUMMERY,TACHE\_NAME,

迭代出map的内容

public static void main(String[] args) {

Map<String, String> map = new HashMap<String, String>();

map.put("1", "a");

map.put("222", "aa");

map.put("33", "aaa");

Set<String> set = new HashSet<String>();

set= map.keySet();

for (String key : set) {

//循环取出了你map里面的值然后再调用你的sql方法想怎么存就怎么存

System.out.print(key+" = "+map.get(key));

}

}

xml节点存入map

SAXReader reader = new SAXReader();

Map map = new HashMap();

String name = null;

String value = null;

Element element = reader.read(new ByteArrayInputStream(xml.getBytes("GBK"))).getRootElement();

List<Element> elements = element.elements();

for(int i=0;i<elements.size();i++){

map.put(elements.get(i).getName(),elements.get(i).getText());

}

oracle日期相差的小时并四舍五入

floor((oo.LIMIT\_DATE-ofo.FIRST\_DEAL\_TIME)\*24)

創建子節點并賦值

public void createChildElemet(Element operElement, String elementName,

String elementValue) {

Element element = XMLDom4jUtils.appendChild(operElement, elementName);

if (elementValue != null && !"".equals(elementValue))

element.addText(elementValue);

}

从零创建xml

Document xmlDoc = XMLDom4jUtils.createDocument();

Element rootele = xmlDoc.addElement("Request");

xmlDoc.setXMLEncoding("GBK");

xmlDoc.setRootElement(rootele);

String转xml

Document docu = DocumentHelper.parseText(xml);

System.out.println("===================docu=========="+docu.asXML());

xml转map

private static HashMap transferXML2Map(Document doc)

{

HashMap map = new HashMap();

Node dataNode = doc.getRootElement().selectSingleNode("params");

List paramList = dataNode.selectNodes("param");

for(int i=0;i<paramList.size();i++)

{

Node param = (Node) paramList.get(i);

Node code = param.selectSingleNode("code");

Node value = param.selectSingleNode("value");

map.put(code.getStringValue(),value.getStringValue());

}

return map;

}

js的insertAdajcentHTML的用法

在指定的地方插入html标签语句

http://blog.csdn.net/zjlovety/article/details/6648716

java与存储过程的合作

java

Connection conn = null;

CallableStatement proc = null;

String callStr = "{call PROC\_COL\_CONFIG(?,?,?,?,?,?,?,?)}";

conn = getConnection();

proc = conn.prepareCall(callStr);

proc.registerOutParameter(8, OracleTypes.CURSOR);//设置第八个输出变量为游标

proc.execute();

rs = (ResultSet) proc.getObject(8);//取第八个变量转化成结果集

sql的存储过程

CREATE PROCEDURE IOMTEST.PROC\_COL\_CONFIG(P\_STAFF\_ID IN NUMBER,

P\_PAGE\_ID IN NUMBER,

P\_ACTION IN NUMBER,

P\_INDEX\_STR IN VARCHAR2,

P\_DISPLAY\_STR IN VARCHAR2,

P\_FIELD\_STR IN VARCHAR2,

P\_WIDTH\_STR IN VARCHAR2,

P\_RC OUT PKG\_FAULT\_REPORT\_QUERY.RESULT\_RC) --过程头的处理

OPEN P\_RC FOR SELECT \* FROM TEST --返回输出的游标结果集，就这么写了

java返回js的collection形式

java

-------------

Collection col = new ArrayList();

EomsOrderExtendDto dto = new EomsOrderExtendDto();

dto.setId(new Long(rs.getLong("ID")));

dto.setEomsOrderId(new Long(rs.getLong("EOMS\_ORDER\_ID")));

dto.setAttrId(new Long(rs.getLong("ATTR\_ID")));

dto.setTacheCode(rs.getString("TACHE\_CODE"));

dto.setAttrValue(rs.getString("ATTR\_VALUE"));

dto.setValueName(rs.getString("VALUE\_NAME"));

dto.setState(rs.getString("STATE"));

dto.setStateDate(rs.getTimestamp("STATE\_DATE"));

dto.setCreateDate(rs.getTimestamp("CREATE\_DATE"));

col.add(dto);

return col;

js

-------------------

var eomsOrderAttrList=callRemoteFunctionNoTrans("com...

for(i=0;i<eomsOrderAttrList.length;i++){

newRowId=eomsOrderAttrList[i].rowId;

}

循环object的所有属性值

this.showObject = function(retObj){

//显示initObj所有属性值

var props = "" ;

for (var p in retObj){

if (typeof(retObj[p]) == "function"){

}else{

props += p + " = " + retObj[p] + " \t ";

}

}

alert ("retObj="+props) ;

}

button上的鼠标变成手型

style="cursor: hand;"

js获取元素所在的tr

var tr=obj.parentElement.parentElement;

js调用存储过程，传入orderid为参数，返回字符串类型

var getTabInfo = callRemoteFunctionNoTrans("com.ztesoft.iom.order.bl.OrderQuery","getTabDetail",orderId,"procEomsOrderExtendInsert");

js判断undefined

typeof(document.all.eoms\_10344)!="undefined"

oracle字段操作

添加字段的语法：alter table tablename add (column datatype [default value][null/not null],….);

修改字段的语法：alter table tablename modify (column datatype [default value][null/not null],….);

删除字段的语法：alter table tablename drop (column);

sqlplus中文乱码问题

登入oracle用户后运行sqlplus前运行一下

export NLS\_LANG="SIMPLIFIED CHINESE\_CHINA.ZHS16GBK"

ZTEsoftSelect:ZTEsoftSelect的添加以及选定

<td class="td\_blue\_txt" align="right">本地网：</td>

<td class="td\_grey" align="left">

<!-- <select class='select\_htc' id='faultArea' name="faultArea" ></select> -->

<span style="width:130px;">

<ZTEsoftSelect:ZTEsoftSelect id="faultArea" check="true" disable="false" showTreeImage="false" showImage="false" imgStyle="" checkBind="M" txtField="text" listWidth="100" listHeight="70">

<ZTEsoftSelect:property>

<ZTEsoftSelect:columns>

<ZTEsoftSelect:column width="100" display="true" displayText="text" propertyName="text"></ZTEsoftSelect:column>

<ZTEsoftSelect:column display="false" displayText="value" propertyName="value"></ZTEsoftSelect:column>

</ZTEsoftSelect:columns>

</ZTEsoftSelect:property>

</ZTEsoftSelect:ZTEsoftSelect>

</span>

</td>

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

this.initFaultArea = function(){

if(document.all.faultArea){

try{

var areaDtos = callRemoteFunctionNoTrans("com.ztesoft.sa.saorder.bl.FaultMonitorQuery","getAllFaultArea");

var xml='<property><items>';

if(areaDtos && areaDtos.length && areaDtos.length>0 ){

for(var i=0;i<areaDtos.length;i++){

xml = xml+'<item text="'+areaDtos[i].faultAreaName+'" value="'+areaDtos[i].faultAreaId+'"></item>';

}

}

xml = xml+'</items></property>';

document.all.faultArea.loadByXML(xml);

}catch(ex){

ErrorHandle(ex,1);

}

}

if(session1.area.areaId!=null&& session1.area.areaId!=1){

var itemsqueryRange=document.all.faultArea.items;

for(var i = 0; i < itemsqueryRange.length; i++){

if(itemsqueryRange[i].value==session1.area.areaId){

itemsqueryRange[i].setChecked(true);

}

}

document.all.faultArea.disable = true;

document.all.faultArea.displaySelectedInfo();

}

}

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

js设置title

document.title

js上的String转成XML

function createXml(str){

if(document.all){

var xmlDom=new ActiveXObject("Microsoft.XMLDOM");

xmlDom.loadXML(str);

return xmlDom;

}

else {

return new DOMParser().parseFromString(str, "text/xml");

}

}

dom4j读取xml信息

String contentXml = "<?xml version=\"1.0\" encoding=\"GBK\" ?><Request><Business>55101</Business><SERIALNO>2014070914572482194080</SERIALNO><SGHM>2014070914572482194080</SGHM><LINETYPE>1</LINETYPE><ZJHM>123</ZJHM><SGSJ>2012-07-05 10:50:12</SGSJ><DELXFS>111</DELXFS><ZAXX>505</ZAXX><SLR>193123</SLR><SLLXR>11</SLLXR><LXDH>111</LXDH><ZASX>100</ZASX><SGBZ>1</SGBZ><DQDM>025</DQDM><KHXM>1</KHXM><YHDZ>1</YHDZ><ZJRQ>2012-07-05 10:50:12</ZJRQ><SYZT>1</SYZT><USERID>9109110711082876</USERID></Request>";

Document document = null;

Element element = null;

document = XMLHelper.fromXML(contentXml, "UTF-8");

element = document.getRootElement();

if(XMLHelper.child(element, "Business",true)!=null&&!"".equals(XMLHelper.child(element, "Business").getText())){

param.setBusiness(XMLHelper.child(element, "Business").getText());

}

直接js运行sql返回list

大方法导入语句\_extends(this, BSCommon, '"form1"');

this.clearParam();

this.addParam(1, "INF\_CODE", faultArea);

var bureauList = this.callRemoteFunctionQuery("qryBureauByAreaId");

oracle小数与字符串结合是小数点的0消失解决办法

select to\_char(0.1,'fm9999990.9999') from dual

简化统计sql的方法

,SUM(case when oo.EXT\_STATE=7 and a.extflag =0 and frr.FAULTCAUSE is null then 1 else 0 end) as satisfy /\*满意\*/

//截取中文字符串

//截取字符串(包括中文）

this.SetString = function(str,len){

var strlen = 0;

var s = "";

for(var i = 0;i < str.length;i++)

{

if(str.charCodeAt(i) > 128){

strlen += 2;

}else{

strlen++;

}

s += str.charAt(i);

if(strlen >= len){

return s ;

}

}

return s;

}

oracle判断字符串是不是数字

select nvl2(translate( '12243112321', '\1234567890 ', '\'), 'is not number ', 'is number ') from dual

java 对Wed Sep 16 11:26:23 CST 2009日期格式字符串转成正常格式

String dateStr = "Wed Sep 16 11:26:23 CST 2009";

SimpleDateFormat sdf = new SimpleDateFormat("EEE MMM dd HH:mm:ss zzz yyyy", Locale.US);

//java.util.Date对象

Date date = (Date) sdf.parse(dateStr);

//2009-09-16

String formatStr = new SimpleDateFormat("yyyy-MM-dd").format(date);

System.out.println(formatStr);

//2009-09-16 11:26:23

String formatStr2 = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss").format(date);

System.out.println(formatStr2);

java调用action

Map ttm=new HashMap();

ttm.put("parameter\_1", tempMap);

tm2=(Map)ActionExecution.singleton().excuteNestedAction(

"/fault\_config\_timelimit", "CreateTacheTimeLimit",

ttm);

oracle递归函数

START WITH rlr.RELATED\_GEOGRAPHY\_LOC\_ID='133325' CONNECT BY PRIOR rlr.GEOGRAPHY\_LOC\_ID = rlr.RELATED\_GEOGRAPHY\_LOC\_ID

oracle分页查询

V\_BASE\_SQL\_STRING\_GD :='SELECT \* FROM (SELECT N.\*,ROWNUM AS ROW\_ID FROM (' || V\_BASE\_SQL\_STRING\_GD || ') N ) WHERE ROW\_ID >= ' || V\_ID1 || ' AND ROW\_ID <' || V\_ID2

查看存储过程

select text from all\_source where name = 'PROC\_DAILY\_FAULT\_INFO'

查找锁表sql

SELECT sess.sid,sess.serial#,lo.oracle\_username,lo.os\_user\_name,ao.object\_name,lo.locked\_mode

from v$locked\_object@dl\_iom lo, dba\_objects@dl\_iom ao,v$session@dl\_iom sess

where ao.object\_id = lo.object\_id and lo.session\_id = sess.sid ORDER BY sess.sid;

js分解object

function JsonToString(o) {

var arr = [];

var fmt = function(s) {

if (typeof s == 'object' && s != null) return JsonToStr(s);

return /^(string|number)$/.test(typeof s) ? "'" + s + "'" : s;

}

for (var i in o)

arr.push("'" + i + "':" + fmt(o[i]));

return '{' + arr.join(',') + '}';

}

mySQL日期时间比较

UNIX\_TIMESTAMP(DISPATCH\_DATE) >UNIX\_TIMESTAMP('1999-01-13 10:28:00')

jQuery easyUI dataGrid getSelections用法

var arr = $('#grid1').datagrid('getSelections');

alert(arr[0].number);

jQuery easyUI dataGrid appendRow用法

var arr = $('#grid1').datagrid('getSelections');

for(var i=0;i<arr.length;i++){

$('#grid2').datagrid('appendRow',arr[i]);

}

mysql中文建表

ENGINE=InnoDB DEFAULT CHARSET=utf8

js字符串转日期及日期比较

if(new Date(Date.parse($("input[name='requiredFinishTime']").val().replace("-", "/")))<new Date()){

js 拼日期

var d = new Date();

var dateStr = d.getFullYear()+"-"+d.getMonth()+"-"+d.getDate()+" "+d.getHours()+":"+d.getMinutes()+":"+d.getSeconds();

extjs DateField 控件取值格式化

Ext.getCmp('requiredFinishTime').getValue().format('Y-m-d h:i:s')

随机数java

java.util.Random rd;

rd.nextInt(240);

js获取url参数

var str=window.location.href;

//alert(str);

var es=/urlLoginId=/;

es.exec(str);

urlLoginId=RegExp.rightContext;

extjs radiogroup的change listeners写法

{

xtype:'radiogroup',

fieldLabel : "审核结果",

id:'auditResult',

items : [{

boxLabel : '通过',

inputValue : "1",

name : "rg",

checked : true

}, {

boxLabel : '不通过',

name : "rg",

inputValue : "0"

}],

listeners : {

change: function(radiofield,oldvalue) {

if(radiofield.getValue().inputValue == 0){

Ext.getCmp("approveOpinionEdit").setValue("");

}else if(radiofield.getValue().inputValue == 1){

Ext.getCmp("approveOpinionEdit").setValue("同意");

}

}

},

anchor:'95%'

}

js日期加一天

new Date(new Date().getTime() + 24\*60\*60\*1000)

extjs的Textfield的非中文输入验证

editor: new Ext.form.TextField({

regex :/^[A-Za-z0-9]+$/,

regexText : "只能输入数字和字母!",

allowBlank: false

})

extjs的grid的行remove后new Ext.grid.RowNumberer({header:'序号',width:40})序号列的数值刷新

grid.getView().refresh()

extjs的TextField对日期的验证

new Ext.form.TextField({//new Ext.form.DateField({//new Ext.ux.form.DateTimeField({

regex :/^(?:(?:1[6-9]|[2-9][0-9])[0-9]{2}([-/.]?)(?:(?:0?[1-9]|1[0-2])\1(?:0?[1-9]|1[0-9]|2[0-8])|(?:0?[13-9]|1[0-2])\1(?:29|30)|(?:0?[13578]|1[02])\1(?:31))|(?:(?:1[6-9]|[2-9][0-9])(?:0[48]|[2468][048]|[13579][26])|(?:16|[2468][048]|[3579][26])00)([-/.]?)0?2\2(?:29))/,

allowBlank: false//,

// format: 'Y-m-d h:i:s'

})

用ExtJs实现列表中点击一个按钮可以动态的给列表增加一行

function addColumn(){

var grid = Ext.getCmp('gridPanelID');

var stores = grid.getStore();

var recordType = grid.getStore().recordType;

var data = new recordType();

grid.stopEditing();

stores.insert(0, data);

stores.sort();

grid.getView().refresh();

grid.startEditing(0, 0);

};

剔除重复项向grid内加记录extjs

var st = Ext.getCmp('modifyResourceList').getStore();

var fla = 0;

for(var i=0;i<resData.length;i++){

for(var j=0;j<st.data.items.length;j++){

if(resData[i].PHYSICAL\_DEVICE\_ID==st.data.items[j].data.resourceId

&&resData[i].RES\_CLASS\_ID==st.data.items[j].data.resourceClassId){

fla = 1;

break;

}

}

if(fla==0){

var resDataTmp = new Object();

resDataTmp.resourceId = resData[i].PHYSICAL\_DEVICE\_ID;

resDataTmp.resourceClassId = resData[i].RES\_CLASS\_ID;

resDataTmp.resourceName = resData[i].PHYSICAL\_DEVICE\_NAME;

resDataTmp.specialtyDomain = resData[i].SPECIALITY\_TYPE\_ENUM\_ID;

Ext.getCmp('modifyResourceList').getStore().add(new Ext.data.Record(resDataTmp));

}

fla = 0;

}

extjs获取grid的某行某列

Ext.getCmp('teleEditGrid').getStore().data.items[0].data.attributionCityName

Extjs中EditorGridPanel获取某一单元格内的控件的相关信息

1. 获取列号为colIndex行号为rowIndex的单元格内的控件的方法为：

var compo= grid.getColumnModel().getCellEditor(colIndex, rowIndex);

2. 获取该单元格内的控件的值的方法为：

var value = compo.getValue();

3. 假设compo为combobox控件，则获取控件的数据源的方法为：

var compoStore = compo.field.getStore();

4. 在3的基础上获取选中的combobox的Index的方法：

var index = compoStore .find('列名',value);

extjs循环grid的一列找出重复值

this.enumCheckUnique = function(dataArray){

var flag = true;

for(var i=0;i<dataArray.length;i++){

var enumValue = dataArray[i].data.enumValue;

var enumValueMeaning = dataArray[i].data.enumValueMeaning;

for(var j=i+1;j<dataArray.length;j++){

if(enumValue == dataArray[j].data.enumValue){

Ext.Msg.alert('操作提示','第'+(i+1)+'行枚举值和第'+(j+1)+'行相同!');

return false;

}

if(enumValueMeaning == dataArray[j].data.enumValueMeaning){

Ext.Msg.alert('操作提示','第'+(i+1)+'行含义和第'+(j+1)+'行相同!');

return false;

}

}

}

return flag;

}

java读取excel单元值时候，1变成了1.0的解决 办法

new DecimalFormat("0").format(cell.getNumericCellValue());

extjs的combox赋值，用getValue()不会取值

Ext.getCmp("testProvinceId").setRawValue(session.logonAccount.provinceCompanyName);

extjs grid new Ext.grid.CheckboxSelectionModel()默认勾选

var aa = new Array();

if((response.rows[0].cardOperationTypeEnumId==3)&&(flag=="6"||flag=="0")){//借用 （申请人确认环节或者详情） 勾选已接收的卡

var st = Ext.getCmp('testCardLendList').getStore();

for(var i=0;i<st.getCount();i++){

var record = st.getAt(i);

if(record.data.isAccepted=="accepted"){

aa.push(record);

}

}

Ext.getCmp('testCardLendList').getSelectionModel().selectRecords(aa);

}

extjs 让一个grid结束编辑

Ext.getCmp('userInfoGrid').stopEditing()

extjs 修改控件的html属性

Ext.getCmp('id').update(html)

js 全部替换字符串

retVal.msg.replace(/<BR>/g,"\n");// 前面是正则表达式，/g表示全部

extjs 下拉框 鼠标停留显示浮动框 属性tpl

items : {

xtype : 'ZTESOFT.combofield',

hideLabel : true,

name : 'cities',

id : 'cities',

valueField : 'id',

displayField : 'text',

mode : 'local',

triggerAction : 'all',

forceSelect : true,

editable : false,

value : '',

tpl: '<tpl for="."><div ext:qtip="{text}" class="x-combo-list-item">{text}</div></tpl>',

store : new Ext.data.JsonStore({

remoteSort: true,

fields: ['id', 'text']

}),

anchor : '100%'

}

JBI

企业服务总线，用Java实现面向服务的架构，JBI目的主要是在于创建一个可以集成各种组件服务的运行环境，当然这也是一种服务总线思想的体现

在JBI容器内部，只有一种标准的规格化消息（Normalized Message）。任务服务组件进入JBI环境之前，通过BC转换为规格消息NM。在JBI环境里，所有的服务都不能相互调用，不论是请求还是回答消息，都要先转给NMR，再由NMR分发。JBI运行环境里面的组件（SE、BC）和NMR都是通过NM来进行信息交换的。

JTA

事务管理

JMS

消息队列，一对一，订阅

SCA

实现了两个功能：一是组件和传输协议的分离，二是接口和实现语言的分离

jquery

易扩展，ui一般，轻量级，面向过程

extjs

组件化，富客户端，加载速度慢，ui效果好，面向对象

binary

主键不要设置为binary的，否则索引无效，如果真的希望主键是大小写敏感的，可能有别的办法

mysql中的to\_char和to\_date

date\_format(date,'%Y-%m-%d %T') -------------->oracle中的to\_char();

str\_to\_date(date,'%Y-%m-%d %T') -------------->oracle中的to\_date();

mysql表的自关联

Mysql不支持表自关联update/delete

替代方案：根据原表建立一个要更新数据的临时表，关联该临时表

mysql日期运算

DATE\_ADD(now(),INTERVAL -48 HOUR)

mysql的getOrgChildList存过写法，重点是group\_concat、FIND\_IN\_SET函数的运用，rootId为传入参数

begin

#递归包含自身，rootId为开始递归的节点

DECLARE sTemp VARCHAR(21845);

DECLARE sTempChd VARCHAR(21845);

SET sTemp = '';

SET sTempChd =cast(rootId as CHAR);

WHILE sTempChd is not null DO

SET sTemp = concat(sTemp,',',sTempChd);

SELECT group\_concat(org\_id) INTO sTempChd FROM uos\_org where FIND\_IN\_SET(parent\_id,sTempChd)>0;

END WHILE;

RETURN sTemp;

END

mysql两个日期相差的时间

select timestampdiff(MINUTE,now(),effect\_date)

Apache JMeter是Apache组织开发的基于Java的压力测试工具。

apache http server性能调优

总Apache进程数 = KeepAliveTimeout \* 每秒种HTTP请求数 / 平均KeepAlive请求

Apache占用内存 = 总Apache进程数 \* 平均每进程占用内存数

基于上面的公式，我们就可以推算出当 平均KeepAlive请求 <= KeepAliveTimeout 时，关闭 KeepAlive 选项是划算的，否则就可以考虑打开。

Redis

值得一说的是它的独家功能：存储列表和集合，这是它与mc之流相比更有竞争力的地方

Redis 是一个高性能的key-value数据库

机构在选择和使用开源技术时，你有什么建议？

曾宪杰：我个人觉得需要考虑这几个方面：

这个软件目前的功能和它的RoadMap

软件本身的架构

该软件开发的活跃度

该开源软件是否是遵守该领域内的国际规范的

在同类产品中，要挑选有比较优势的。并且要考虑可能存在的移植代价。这个移植指的是采用了这款开源软件后现有系统的移植，或者是从这个开源软件到其他软件的移植。

一个大型的高并发高性能网站架构需要从哪些层面去考虑？技术堆栈的选择又应该注意什么？

曾宪杰：对于大型的高并发高性能网站的架构，我觉得主要考虑的方面包括水平扩展性、稳定性、架构相关基础设施的易用性、容错性、异地数据中心支持等方面。

而技术方面的选择，如果不考虑现有技术人员的情况，那么建议选择的是有成功案例并且社区比较活跃的技术。

hadoop

checkSum secondaryNameNode(冷备份，帮助NN合并eidts log，减少NN启动时间) fsimage edits

HDFS nameNode dataNode secondaryNameNode

mapreduce jobTracker taskTracker

hadoop job scheduler：FIFO调度器 Fair Scheduler公平调度器 Capacity scheduler计算能力调度器

Android的广播接收者BroadCastReceiver

两种接收者receiver的注册方式：建议在onResume中注册接收者，在onPuase中注销接收者

1.系统注册：在AndroidManifest.xml中注册

2.代码注册：在activity中注册

三种发广播方式：

1.普通：sendBroadcast(),不能截断广播

2.有序：sendOrderedBroadcast(),优先的接收者可以将广播截断abortBroadcast(),后面的接收者就不能接收该广播

3.异步：sendStickyBroadcast(),需要配置权限BROADCAST\_STICKY

云的真相

基于虚拟化的硬体资源伸缩性

基于虚拟化的服务可扩展性

提供了按需付费，俺是付费的弹性服务

云的选择

openstack（云上的linux，虚拟化各种系统，新） vagrant（虚拟机） docker（操作系统和应用程序一起打包，避免环境不同造成的运行差异）

掌握bash、sed、awk三剑客编程

Android网络下载省电技巧：

预取数据

批量传输与连接

减少连接次数：重用连接，网络连接不要立即关闭，可以用一次get获取就不要用两次get

DDMS监控工具：可监控网络传输情况，制定合理的网络连接代码

Cloud to Device Messaging(C2DM):当有更新的时候通知对应的app，代替由app定时向server询问是否有更新

Android获取缓存目录

获取外部缓存的目录Context.getExternalCacheDir();

获取内部缓存的方法Context.getCache();

使用反射机制开启HTTP response cache

private void enableHttpResponseCache() {

try {

long httpCacheSize = 10 \* 1024 \* 1024; // 10 MiB

File httpCacheDir = new File(getCacheDir(), "http");

Class.forName("android.net.http.HttpResponseCache")

.getMethod("install", File.class, long.class)

.invoke(null, httpCacheDir, httpCacheSize);

} catch (Exception httpResponseCacheNotAvailable) {

Log.d(TAG, "HTTP response cache is unavailable.");

}

}

所有cache中的HTTP请求都可以直接在本地存储中进行响应，并不需要开启一个新的网络连接

被cache起来的response可以被server所确保没有过期

WIFI耗电最小，使用手机网络的情况下，使用带宽越宽，预取数据应该尽量多，尽量拉长网络激活时间

Android解决同一用户在多台设备上存储云数据的一致性问题（云内存只有128k）

解决一：只记录总分，不完善

解决二：记录总分和增量，add（total，add），不完善

解决三：记录每台设备上的总分，X:x,A:a,B:b，x为初始值，总分为x+a+b

Android创建异步设配器传输数据，服务器到设备之间，需要：

1.Authenticator：存储用户信息 注册到xml文件夹下

2.bound service：拥有Authenticator，提供Authenticator 注册到manifest.xml里

3.content provider：提供数据 注册到application下

4.sync Adapter：实现AbstractThreadedSyncAdapter 其中onPerformSync为实际的传输数据操作，开发者需要做的步骤：连接网络；上传或下载数据；处理数据差异；关闭连接和清除临时文件

5.又一个bound service：拥有sync Adapter（static类型），其中onBind方法返回Adapter的binder给系统，以便系统能够调用Adapter的onPerformSync方法

6.权限：INTERNET、READ\_SYNC\_SETTINGS、WRITE\_SYNC\_SETTINGS、AUTHENTICATE\_ACCOUNTS

Android执行同步操作的5中方式：

1.Google Cloud message：用BroadcastReceiver 来接收执行命令，ContentResolver.requestSync(ACCOUNT, AUTHORITY, null);

2.content provider数据变更触发：对provider绑定一个观察者contentObserver，ContentResolver.requestSync(ACCOUNT, AUTHORITY, null);

3.网络连接命令：当设备连接上网络后，会定期发送一个tcp/ip命令，用contentReceiver捕获并触发同步，mResolver = getContentResolver();mResolver.setSyncAutomatically(ACCOUNT, AUTHORITY, true);

4.定时执行：设定一个时间间隔，定时执行同步，ContentResolver.addPeriodicSync(

ACCOUNT,

AUTHORITY,

null,

SYNC\_INTERVAL);//SYNC\_INTERVAL的单位为毫秒

5.用户触发：由用户手动执行，Bundle settingsBundle = new Bundle();

settingsBundle.putBoolean(

ContentResolver.SYNC\_EXTRAS\_MANUAL, true);

settingsBundle.putBoolean(

ContentResolver.SYNC\_EXTRAS\_EXPEDITED, true);

ContentResolver.requestSync(mAccount, AUTHORITY, settingsBundle);

Android处理Request：

1.创建RequestQueue：Volley.newRequestQueue(this)//主线程

2.创建Request：StringRequest stringRequest = new StringRequest(Request.Method.GET, url,

new Response.Listener() {

@Override

public void onResponse(String response) {

// Display the first 500 characters of the response string.

mTextView.setText("Response is: "+ response.substring(0,500));

}

}, new Response.ErrorListener() {

@Override

public void onErrorResponse(VolleyError error) {

mTextView.setText("That didn't work!");

}

});

3.把Request加入Queue：add

4.可取消Request：先加标识：stringRequest.setTag(TAG);后取消：mRequestQueue.cancelAll(TAG);

Android的两种HTTP连接方式：

1.AndroidHttpClient：bug相对较少，需要的代码量大，已停止维护

2.HttpURLConnection：轻量，API Level 9之前不适用，即姜饼

Android配置RequestQueue

Cache cache = new DiskBasedCache(getCacheDir(), 1024 \* 1024);

Network network = new BasicNetwork(new HurlStack());

mRequestQueue = new RequestQueue(cache, network);

mRequestQueue.start();

Android把RequestQueue设为Singleton，在整个application的生命周期可用

Volley.newRequestQueue(mCtx.getApplicationContext());//初始化时传入application对象

Android的ImageLoader类

帮助类，主要用于缓存图片，最佳用法是设为singleton，存在于app的整个生命周期，而不是在activity里面创建

mImageLoader = new ImageLoader(mRequestQueue,  
                new ImageLoader.ImageCache() {  
            private final LruCache<String, Bitmap>  
                    cache = new LruCache<String, Bitmap>(20);  
  
            @Override  
            public Bitmap getBitmap(String url) {  
                return cache.get(url);  
            }  
  
            @Override  
            public void putBitmap(String url, Bitmap bitmap) {  
                cache.put(url, bitmap);  
            }  
        });

Android获取屏幕显示的字节数，一像素是4字节，可用于算图片缓存的大小

final DisplayMetrics displayMetrics = ctx.getResources().  
                getDisplayMetrics();  
        final int screenWidth = displayMetrics.widthPixels;  
        final int screenHeight = displayMetrics.heightPixels;  
        // 4 bytes per pixel  
        final int screenBytes = screenWidth \* screenHeight \* 4;

Android的4种请求：

1.ImageRequest

2.JsonArrayRequest

3.JsonObjectRequest

4.StringRequest

Android根据查询条件获取通讯录结果步骤：

1.申请权限：android.permission.READ\_CONTACTS

2.定义ListView和item layout

3.定义Fragment

4.设定各种全局变量

5.初始化Fragment

6.为ListView设置CursorAdapter

mCursorAdapter = new SimpleCursorAdapter(  
                getActivity(),  
                R.layout.contact\_list\_item,  
                null,  
                FROM\_COLUMNS, TO\_IDS,  
                0);

mContactsList.setAdapter(mCursorAdapter);

7.为ListView设置 [AdapterView.OnItemClickListener](http://developer.android.com/reference/android/widget/AdapterView.OnItemClickListener.html)

8.定义返回的列

@SuppressLint("InlinedApi")

private static final String[] PROJECTION =  
        {  
            Contacts.\_ID,  
            Contacts.LOOKUP\_KEY,  
            Build.VERSION.SDK\_INT  
                    >= Build.VERSION\_CODES.HONEYCOMB ?  
                    Contacts.DISPLAY\_NAME\_PRIMARY :  
                    Contacts.DISPLAY\_NAME  
  
        };

9.定义内容的游标列号

private static final int CONTACT\_ID\_INDEX = 0;

private static final int LOOKUP\_KEY\_INDEX = 1;

10.明确查询条件

@SuppressLint("InlinedApi")  
    private static final String SELECTION =  
            Build.VERSION.SDK\_INT >= Build.VERSION\_CODES.HONEYCOMB ?  
            Contacts.DISPLAY\_NAME\_PRIMARY + " LIKE ?" :  
            Contacts.DISPLAY\_NAME + " LIKE ?";

private String mSearchString;

private String[] mSelectionArgs = { mSearchString };

如果指定查询某种类型，如下

private static final String SELECTION =  
            /\*  
             \* Searches for an email address  
             \* that matches the search string  
             \*/  
            Email.ADDRESS + " LIKE ? " + "AND " +  
            /\*  
             \* Searches for a MIME type that matches  
             \* the value of the constant  
             \* Email.CONTENT\_ITEM\_TYPE. Note the  
             \* single quotes surrounding Email.CONTENT\_ITEM\_TYPE.  
             \*/  
            Data.MIMETYPE + " = '" + Email.CONTENT\_ITEM\_TYPE + "'";

11.定义onItemClick()方法

12.初始化loader，在Fragment的onActivityCreated内

13.实现onCreateLoader()，返回一个CursorLoader

return new CursorLoader(  
                getActivity(),  
                Contacts.CONTENT\_URI,  
                PROJECTION,  
                SELECTION,  
                mSelectionArgs,  
                null  
        );

如果一个值要查所有类型，如下：

Uri contentUri = Uri.withAppendedPath(  
                Contacts.CONTENT\_FILTER\_URI,  
                Uri.encode(mSearchString));  
        // Starts the query  
        return new CursorLoader(  
                getActivity(),  
                contentUri,  
                PROJECTION,  
                null,  
                null,  
                null  
        );

14.实现onLoadFinished() 和 onLoaderReset()

public void onLoadFinished(Loader<Cursor> loader, Cursor cursor) {  
        // Put the result Cursor in the adapter for the ListView  
        mCursorAdapter.swapCursor(cursor);  
    }

public void onLoaderReset(Loader<Cursor> loader) {  
        // Delete the reference to the existing Cursor  
        mCursorAdapter.swapCursor(null);  
  
    }

Android用intent修改通讯录

Intent intent = new Intent(Intents.Insert.ACTION);// Intent.ACTION\_EDIT

或者Intent.ACTION\_INSERT\_OR\_EDIT

intent.putExtra(Intents.Insert.EMAIL, mEmailAddress.getText());//添加信息

editIntent.setDataAndType(mSelectedContactUri,Contacts.CONTENT\_ITEM\_TYPE);//修改时指定修改哪一条通讯录

intentInsertEdit.setType(Contacts.CONTENT\_ITEM\_TYPE);//用于既添加又修改的情况

startActivity(intentInsertEdit);

Android注意android:layout\_toRightOf属性

<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"  
                android:layout\_width="match\_parent"  
                android:layout\_height="wrap\_content">  
    <QuickContactBadge  
        android:id="@+id/quickcontact"  
        android:layout\_height="wrap\_content"  
        android:layout\_width="wrap\_content"  
        android:scaleType="centerCrop"/>  
    <TextView android:id="@+id/displayname"  
              android:layout\_width="match\_parent"  
              android:layout\_height="wrap\_content"  
              android:layout\_toRightOf="@+id/quickcontact"  
              android:gravity="center\_vertical"  
              android:layout\_alignParentRight="true"  
              android:layout\_alignParentTop="true"/>  
</RelativeLayout>

Android获取位置信息

引入google service jar包

权限ACCESS\_COARSE\_LOCATION(模糊位置)  ACCESS\_FINE\_LOCATION(精确位置)

检查是否安装google service APK：int resultCode = GooglePlayServicesUtil.isGooglePlayServicesAvailable(this);// ConnectionResult.SUCCESS为成功

实现GooglePlayServicesClient.ConnectionCallbacks, GooglePlayServicesClient.OnConnectionFailedListener监听

其中public void onConnectionFailed(ConnectionResult connectionResult)方法里可以尝试解决

if (connectionResult.hasResolution()) {  
            try {  
                // Start an Activity that tries to resolve the error  
                connectionResult.startResolutionForResult(  
                        this,  
                        CONNECTION\_FAILURE\_RESOLUTION\_REQUEST);  
                /\*  
                 \* Thrown if Google Play services canceled the original  
                 \* PendingIntent  
                 \*/  
            } catch (IntentSender.SendIntentException e) {  
                // Log the error  
                e.printStackTrace();  
            }  
        }

在onCreate里new LocationClient(this, this, this);

在onStart里mLocationClient.connect();

在onStop里mLocationClient.disconnect();//不然会浪费电

获取位置的方法mLocationClient.getLastLocation();

Android定时更新位置信息

onCreate：初始化两个对象，Request设置setPriority(优先级)，setInterval(更新间隔)，setFastestInterval(最快更新间隔)

new LocationClient(this, this, this);

LocationRequest.create();

onStart:client连接上

mLocationClient.connect();

onResume：可以读取用户设置属性，设置全局变量控制是否持续更新位置

GooglePlayServicesClient.ConnectionCallbacks的onConnect:如果用户没有关闭更新设置，把client和Request绑定

mLocationClient.requestLocationUpdates(mLocationRequest, this);

onPause：保存用户的设置属性（是否持续更新）

onStop：client断开连接，去掉持续更新请求

removeLocationUpdates(this);

mLocationClient.disconnect();

MEAN架构

M：mongo数据库 nosql

E：express 基于Node的web开发框架

A：angular 前端js开发框架

N：Node 基于V8运行时环境

Android的Geofences

Android的识别用户活动

ActivityRecognitionClient：主类

mActivityRecognitionClient =

new ActivityRecognitionClient(mContext, this, this);

Intent intent = new Intent(

mContext, ActivityRecognitionIntentService.class);// ActivityRecognitionIntentService 继承IntentService

mActivityRecognitionPendingIntent =

PendingIntent.getService(mContext, 0, intent,

PendingIntent.FLAG\_UPDATE\_CURRENT);// mActivityRecognitionPendingIntent用于接收ActivityRecognitionClient的对用户活动的更新信息，在ActivityRecognitionIntentService里获取用户活动信息，并处理

mActivityRecognitionClient.requestActivityUpdates(

DETECTION\_INTERVAL\_MILLISECONDS,

mActivityRecognitionPendingIntent);//为mActivityRecognitionClient注册接收信息的intent（mActivityRecognitionPendingIntent）

mActivityRecognitionClient.removeActivityUpdates(

mActivityRecognitionPendingIntent);//这是取消监听

Android的后台任务类IntentService

1.开始一个IntentService

mServiceIntent = new Intent(getActivity(), RSSPullService.class);// RSSPullService类继承IntentService， onHandleIntent()方法里实现后台要处理的代码  
mServiceIntent.setData(Uri.parse(dataUrl));

getActivity().startService(mServiceIntent);

2. RSSPullService类内发送广播告知处理结果

Intent localIntent =  
            new Intent(Constants.BROADCAST\_ACTION)  
            // Puts the status into the Intent  
            .putExtra(Constants.EXTENDED\_DATA\_STATUS, status);  
    // Broadcasts the Intent to receivers in this app.  
    LocalBroadcastManager.getInstance(this).sendBroadcast(localIntent);

注册RSSPullService类，在<application>内加以下

<service  
            android:name=".RSSPullService"  
            android:exported="false"/>

3.创建继承BroadcastReceiver类的ResponseReceiver 类，用于接收处理结果的广播，onReceive方法里为处理结果的代码

4.在activity里注册BroadcastReceiver类，接收包含处理结果的广播

// The filter's action is BROADCAST\_ACTION  
        IntentFilter mStatusIntentFilter = new IntentFilter(  
                Constants.BROADCAST\_ACTION);  
        // Adds a data filter for the HTTP scheme  
        mStatusIntentFilter.addDataScheme("http");

// Instantiates a new DownloadStateReceiver  
       ResponseReceiver mDownloadStateReceiver =  
                new ResponseReceiver ();  
        // Registers the DownloadStateReceiver and its intent filters  
        LocalBroadcastManager.getInstance(this).registerReceiver(  
                mDownloadStateReceiver,  
                mStatusIntentFilter);

Android后台用cursorLoader处理数据

1.activity或FragmentActivity类实现LoaderManager.LoaderCallbacks<Cursor>接口

在onCreate或onCreateView方法里初始化loader

getLoaderManager().initLoader(URL\_LOADER, null, this);// FragmentActivity里用getSupportLoaderManager，URL\_LOADER类似于id，用于区别于其他loader

2.实现onCreateLoader方法，查询数据，返回带数据的cursor

@Override  
public Loader<Cursor> onCreateLoader(int loaderID, Bundle bundle)  
{  
    /\*  
     \* Takes action based on the ID of the Loader that's being created  
     \*/  
    switch (loaderID) {  
        case URL\_LOADER:  
            // Returns a new CursorLoader  
            return new CursorLoader(  
                        getActivity(),   // Parent activity context  
                        mDataUrl,        // Table to query  
                        mProjection,     // Projection to return  
                        null,            // No selection clause  
                        null,            // No selection arguments  
                        null             // Default sort order  
        );  
        default:  
            // An invalid id was passed in  
            return null;  
    }  
}

3.实现onLoadFinished方法，当数据查询完成后自动调用，实现处理数据的代码

@Override  
public void onLoadFinished(Loader<Cursor> loader, Cursor cursor) {  
    ...  
    /\*  
     \* Moves the query results into the adapter, causing the  
     \* ListView fronting this adapter to re-display  
     \*/  
    mAdapter.changeCursor(cursor);  
}

4.解释3里面的处理代码，一个view设置了一个adapter，adapter设置cursor的时候就会自动展示cursor内变更的数据，十分方便，设置方法如下：

public String[] mFromColumns = {  
    DataProviderContract.IMAGE\_PICTURENAME\_COLUMN  
};  
public int[] mToFields = {  
    R.id.PictureName  
};  
// Gets a handle to a List View  
ListView mListView = (ListView) findViewById(R.id.dataList);  
/\*  
 \* Defines a SimpleCursorAdapter for the ListView  
 \*  
 \*/  
SimpleCursorAdapter mAdapter =  
    new SimpleCursorAdapter(  
            this,                // Current context  
            R.layout.list\_item,  // Layout for a single row  
            null,                // No Cursor yet  
            mFromColumns,        // Cursor columns to use  
            mToFields,           // Layout fields to use  
            0                    // No flags  
    );  
// Sets the adapter for the view  
mListView.setAdapter(mAdapter);

5.最后实现onLoaderReset方法，当数据改变，重新查询之前会调用，需清除cursor，避免缓存

@Override  
public void onLoaderReset(Loader<Cursor> loader) {  
      
    /\*  
     \* Clears out the adapter's reference to the Cursor.  
     \* This prevents memory leaks.  
     \*/  
    mAdapter.changeCursor(null);  
}

Android保持屏幕常亮两方法：

1.

**getWindow().addFlags(WindowManager.LayoutParams.FLAG\_KEEP\_SCREEN\_ON);**

2.

<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"  
    android:layout\_width="match\_parent"  
    android:layout\_height="match\_parent"  
    **android:keepScreenOn="true">**  
    ...  
</RelativeLayout>

取消设置屏幕常亮：

getWindow().clearFlags(WindowManager.LayoutParams.FLAG\_KEEP\_SCREEN\_ON)

Android保持cpu清醒

1.权限

<uses-permission android:name="android.permission.WAKE\_LOCK" />

2.代码

PowerManager powerManager = (PowerManager) getSystemService(POWER\_SERVICE);  
Wakelock wakeLock = powerManager.newWakeLock(PowerManager.PARTIAL\_WAKE\_LOCK,  
        "MyWakelockTag");  
wakeLock.acquire();

Android使用WakefulBroadcastReceiver保持cpu清醒：

当运行后台任务时，假如期间系统进入睡眠，则此后台任务将在往后的不确定时间才会完成，因此必须保持cpu清醒

1.注册一个继承WakefulBroadcastReceiver 的receiver

<receiver android:name=".MyWakefulReceiver"></receiver>

2.实现onReceive方法，MyIntentService为后台任务类

public class MyWakefulReceiver extends WakefulBroadcastReceiver {  
  
    @Override  
    public void onReceive(Context context, Intent intent) {  
  
        // Start the service, keeping the device awake while the service is  
        // launching. This is the Intent to deliver to the service.  
        Intent service = new Intent(context, MyIntentService.class);  
        startWakefulService(context, service);  
    }  
}

3. MyIntentService类处理完成后告知receiver取消cpu清醒状态

public class MyIntentService extends IntentService {  
    public static final int NOTIFICATION\_ID = 1;  
    private NotificationManager mNotificationManager;  
    NotificationCompat.Builder builder;  
    public MyIntentService() {  
        super("MyIntentService");  
    }  
    @Override  
    protected void onHandleIntent(Intent intent) {  
        Bundle extras = intent.getExtras();  
        // Do the work that requires your app to keep the CPU running.  
        // ...  
        // Release the wake lock provided by the WakefulBroadcastReceiver.  
        MyWakefulReceiver.completeWakefulIntent(intent);  
    }  
}

Android的定时任务，使用AlarmManager和PendingIntent 类

Android开机后立即运行任务

1.获取接收开机情况的权限

<uses-permission android:name="android.permission.RECEIVE\_BOOT\_COMPLETED"/>

2.用receiver接收此信息

public class SampleBootReceiver extends BroadcastReceiver {  
  
    @Override  
    public void onReceive(Context context, Intent intent) {  
        if (intent.getAction().equals("android.intent.action.BOOT\_COMPLETED")) {  
            // Set the alarm here.  
        }  
    }  
}

注册receiver，其中android:enabled="false"表明该receiver不会自动生效，除非app代码设定

<receiver android:name=".SampleBootReceiver"  
        android:enabled="false">  
    <intent-filter>  
        <action android:name="android.intent.action.BOOT\_COMPLETED"></action>  
    </intent-filter>  
</receiver>

代码设定：

ComponentName receiver = new ComponentName(context, SampleBootReceiver.class);  
PackageManager pm = context.getPackageManager();  
  
pm.setComponentEnabledSetting(receiver,  
        PackageManager.COMPONENT\_ENABLED\_STATE\_ENABLED,  
        PackageManager.DONT\_KILL\_APP);

代码取消：

ComponentName receiver = new ComponentName(context, SampleBootReceiver.class);  
PackageManager pm = context.getPackageManager();  
  
pm.setComponentEnabledSetting(receiver,  
        PackageManager.COMPONENT\_ENABLED\_STATE\_DISABLED,  
        PackageManager.DONT\_KILL\_APP);

Android进程优先级（当内存不足，先去掉哪些？）：

1. Foreground process

。掌管了正与用户交互的，调用了onResume方法的Activity

。掌管了正与用户交互的Activity绑定的service

。掌管了调用了startForeground()方法的service

。掌管了调用了onCreate、onStart、onDestroy方法的service

。掌管了正在执行onReceive的BroadcastReceiver

2. Visible process

。掌管了不在前台但是用户可见的Activity，调用了onPause方法

。掌管了用户可见的Activity绑定的service

3. Service process

不是1和2的service，例如音乐播放、网络下载

4. Background process

调用onStop方法的Activity，被LRU (least recently used) list管理

5. Empty process

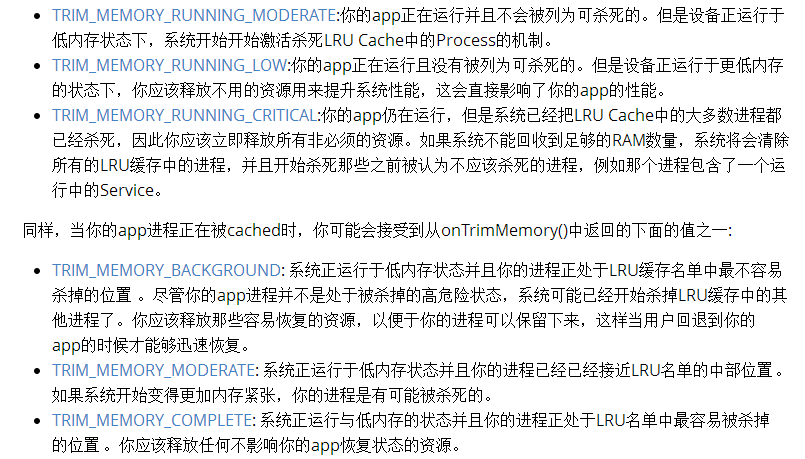
用于缓存目的的

Android应用如何管理内存：

1.尽量用IntentService处理后台任务，因为它会在完成任务后结束自己

2.实现Activity里的onTrimMemory()方法，当这个方法的参数为TRIM\_MEMORY\_UI\_HIDDEN时，说明用户已经离开这个应用，可在方法里释放UI资源，onStop方法是当用户离开当前Activity时调用的，可在此释放网络连接，广播接受者等资源

3.根据onTrimMemory方法的各种参数，判断应该释放多少资源：



4.通过getMemoryClass()获取可用heap大小，通过在application标签内加属性largeHeap=true，并用getLargeMemoryClass()方法获取更大的heap，虽然不一定成功

5.使用优化的数据容器：SparseArray, SparseBooleanArray, 与 LongSparseArray

6. 为序列化的数据使用Protocol buffers

7. 使用ProGuard(java 混淆)来精简编译后的字节码，Android项目发布时自动启用

8. 对最终的APK使用zipalign

9.使用多进程，例如音乐播放应用，一个进程控制UI，一个进程控制后台service，这样可以在适当的时候单独释放前台UI资源，实现方法<service android:name=".PlaybackService"

android:process=":background" />，冒号表示该进程对改应用私有

Android使用sdk\tools下的hierarchyviewer工具可以调试layout性能

Android的include标签，在layout中引入已有layout的示例：

<include android:id="@+id/news\_title"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

layout="@layout/title"/>

只能重写被引入layout的根元素的属性，如果要重写属性，必须连width和height都一并重写了

Android的merge标签

<merge xmlns:android="http://schemas.android.com/apk/res/android">

<Button

android:layout\_width="fill\_parent"

android:layout\_height="wrap\_content"

android:text="@string/add"/>

<Button

android:layout\_width="fill\_parent"

android:layout\_height="wrap\_content"

android:text="@string/delete"/>

</merge>

merge可以代替原本应该是根节点的LinearLayout等，这样在别的layout里include这个layou时就不会发生LinearLayout嵌套LinearLayout的情况，优化了性能

Android的ViewStub标签

ViewStub标签所包含的layout是那种在特定时刻才需要渲染的layout，例如进度条

<ViewStub

android:id="@+id/stub\_import"

android:inflatedId="@+id/panel\_import"

android:layout="@layout/progress\_overlay"

android:layout\_width="fill\_parent"

android:layout\_height="wrap\_content"

android:layout\_gravity="bottom" />

以下方法代码可以把ViewStub包含的layout显示

((ViewStub) findViewById(R.id.stub\_import)).setVisibility(View.VISIBLE);

// or

View importPanel = ((ViewStub) findViewById(R.id.stub\_import)).inflate();

一旦显示之后，id属性就无效了，取而代之的是用inflatedId来获取控件

ViewStub标签不支持用merge标签包含的layout

Android启动一个线程来执行任务

public class PhotoDecodeRunnable implements Runnable {

...

@Override

public void run() {

// 把当前的线程变成后台执行的线程

android.os.Process.setThreadPriority(android.os.Process.THREAD\_PRIORITY\_BACKGROUND);

...

/\*

\* 在PhotoTask实例中存储当前线程，以至于这个实例能中断这个线程

\*/

mPhotoTask.setImageDecodeThread(Thread.currentThread());

}

...

}

Android创建一个线程池

BlockingQueue<Runnable> mDecodeWorkQueue = new LinkedBlockingQueue<Runnable>();

mDecodeThreadPool = new ThreadPoolExecutor(

NUMBER\_OF\_CORES, // Initial pool size

NUMBER\_OF\_CORES, // Max pool size

KEEP\_ALIVE\_TIME,

KEEP\_ALIVE\_TIME\_UNIT,

mDecodeWorkQueue//管理线程的队列，类型为BlockingQueue<Runnable>

);

Android往线程池添加任务及中断线程

mDecodeThreadPool.execute(photoTask.getPhotoDecodeRunnable());//参数是Runnable类型

synchronized (sInstance) {//避免系统正在调用该线程，用变量做同步锁，因为只有等待状态的线程才能中断

new Thread(runnable). interrupt();//只有Thread才有中断方法

}

if (Thread.interrupted()) {//判断是否已中断

return;

}

Java中Runnable和Thread的区别

Runnable是接口，能实现资源共享

Thread是类

1. **class** MyThread **extends** Thread{
2. **private** **int** ticket=10;
3. **public** **void** run(){
4. **for**(**int** i=0;i<20;i++){
5. **if**(**this**.ticket>0){
6. System.out.println("卖票：ticket"+**this**.ticket--);
7. }
8. }
9. }
10. };
11. MyThread mt1=**new** MyThread();
12. MyThread mt2=**new** MyThread();
13. MyThread mt3=**new** MyThread();
14. mt1.start();//每个线程都各卖了10张，共卖了30张票
15. mt2.start();//但实际只有10张票，每个线程都卖自己的票
16. mt3.start();//没有达到资源共享
17. **class** MyThread **implements** Runnable{
18. **private** **int** ticket=10;
19. **public** **void** run(){
20. **for**(**int** i=0;i<20;i++){
21. **if**(**this**.ticket>0){
22. System.out.println("卖票：ticket"+**this**.ticket--);
23. }
24. }
25. }
26. }
27. MyThread mt=**new** MyThread();
28. **new** Thread(mt).start();//同一个mt，但是在Thread中就不可以，如果用同一
29. **new** Thread(mt).start();//个实例化对象mt，就会出现异常
30. **new** Thread(mt).start();

Android跨线程操作UI控件

Activity里面:

looper = Looper.myLooper();

handler = new MyHandler(looper,t);//t为UI控件，MyHandler继承Handler

点击事件处理：

m = new MyRunable(handler,message);// MyRunable继承Runable，执行后台任务的线程类，message为Message类型

new Thread(m).start();//启动线程

MyRunable的run方法：

m.sendMessage(message);//m为m = new MyRunable(handler,message);里的handler，message为message

Android的Application Not Responding

导致ANR的原因：

。输入事件5秒内没有响应

。BroadReceiver不能再10秒内结束任务

Android进度条显示百分比

继承AsyncTask的后台任务类内调用publishProgress方法传入百分比会触发onProgressUpdate方法

publishProgress((int) ((i / (float) count) \* 100));

protected void onProgressUpdate(Integer... progress) {

setProgressPercent(progress[0]);

}

Android的系统架构

1.Linux内核层：硬件驱动

2.运行库和androidruntion：即C/C++函数库部分，例如负责Android网页浏览器运行的webkit；支持游戏开发的2dsgl和3dopengles，Androidruntion负责解释和执行生成的dalvik格式的字节码

3.应用软件架构：java封装的api，提供开发人员调用

4.应用程序层：app

Android的onSaveInstanceState() 和 onRestoreInstanceState()方法何时调用

遇到意外情况，例如内存不足、用户直接按Home键，非用户意愿销毁Activity时候，屏幕横向竖向切换

onSaveInstanceState() 和 onRestoreInstanceState()方法保存临时数据

onPause方法保存持久化数据

HAProxy负载均衡的安装与配置

http://network.51cto.com/art/201103/249586\_1.htm

软件负载均衡一般通过两种方式来实现：基于操作系统的软负载实现和基于第三方应用的软负载实现。LVS就是基于Linux操作系统实现的一种软负载，HAProxy就是开源的并且基于第三应用实现的软负载。

Android的4大组件

http://www.07net01.com/zhishi/316962.html

JNI：Java Native Interface 用java运行本地已编译的其他语言文件，例如C/C++

步骤：http://www.cnblogs.com/youxilua/archive/2011/09/16/2178554.html

1.编写调用其他语言文件的java文件：

2.用java命令编译生成.class文件 javac +文件

3.用java命令编译生成.h文件 javah +文件

4.根据.h文件里的方法名编写c语言文件

5.用C语言的编译器（例如mingw、VC6、GNUStep）把c语言文件编译成so文件或dll文件，window为dll，Android等Linux为so

gcc -Wall -D\_JNI\_IMPLEMENTATION\_ -Wl,--kill-at   -Id:/java/include –Id:/java/include/win32   -shared -o sum.dll java\_sum\_native.c

Android的View和ViewGroup的重写

Linux用nohup命令后台运行java

http://it.100xuexi.com/view/otdetail/20120322/95a89a07-fe4d-410e-b9d7-3e0216fc93ac.html

MySQL优化经验http://www.jb51.net/article/24392.htm

1.优化你的查询语句，使得查询缓存可以用上（即不用curdate(),now()等动态值函数）

2.用explain查询运行规则，分析查询语句

3.当确定查询结果只有1条时，使用limit 1

4.为查询字段添加索引，不只是主键

5. 在Join表的时候使用相同类型的列，并将其索引

6.当需要随机选一条数据时

// 千万不要这样做：   
$r = mysql\_query("SELECT username FROM user ORDER BY RAND() LIMIT 1");   
  
// 这要会更好：   
$r = mysql\_query("SELECT count(\*) FROM user");   
$d = mysql\_fetch\_row($r);   
$rand = mt\_rand(0,$d[0] - 1);   
$r = mysql\_query("SELECT username FROM user LIMIT $rand, 1");

7.避免select \*

8.为每张表设置自增id，关联表除外

9. 使用 ENUM 而不是 VARCHAR

10. 从 PROCEDURE ANALYSE() 取得建议，表的数据量大建议才会准确

11.字段尽可能使用not null，因为null字段需要额外的空间

12. 使用Prepared Statements

13.无缓存查询

14. 把IP地址字段设成UN SIGNED INT类型，使用INET\_ATON()和INET\_NTOA()进行字符串和整形的转换

15.使表的每个字段都是固定长度，非VARCHAR，TEXT，BLOB类型

16.将表里面不常用到的字段拆分出来独立一个表

17.当有大量delete或insert操作时，拆分成小部分分别运行

18.更小的列类型，TINYINT代替INT，DATE代替DATETIME

19.选择正确的存储引擎，MyISAM表锁，适合读操作较多；InnoDB行锁，适合写操作较多

20.使用对象关系映射器ORM

21.小心永久链接，即永久的线程用于数据连接，即使没有数据操作

XSS跨站脚本攻击

http://www.cnblogs.com/TankXiao/archive/2012/03/21/2337194.html

Linux的shell script入门

http://www.cnblogs.com/suyang/archive/2008/05/18/1201990.html

Linux的shell script实际应用

http://justcoding.iteye.com/blog/1943504

Linux的for循环变量自增

#!/bin/bash  
for j in $(seq 1 5)  
do  
  echo $j  
done

其他自增

1. i=`expr $i + 1`;  
2. let i+=1;  
3. ((i++));  
4. i=$[$i+1];  
5. i=$(( $i + 1 ))

Linux grep cut sort sed awk用法

http://www.cnblogs.com/dong008259/category/340620.html

grep：用正则表达式匹配字符串，返回匹配的整行内容

cut：按字节、字符截取每行内容，也可自定义分隔符取制定域内容，按行查找

sed：对行进行新增删除替换甚至行替换，甚至可以直接修改文件 sed -n '/ruby/p' ab | sed 's/ruby/bird/g'

awk：每行设置匹配条件，以及相应的操作行为，最后还有一个终结行为，还有内部变量，分割设定 awk 'BEGIN{total = 0;len = 0} {if($1~/^[0-9]+\.[0-9]\*/){total += $1; len++}} END{print total/len}' a

Linux判断是tab还是多个空格

[rocrocket@rocrocket programming]$ cat tab\_space.txt  
this is tab finish.  
this is several space      finish.  
[rocrocket@rocrocket programming]$ sed -n l tab\_space.txt  
this is tab\tfinish.$  
this is several space      finish.$  
看到了吧，如果是制表符（TAB），那么会显示为\t符号，如果是空格，就会原样显示。  
通过此方法即可以判断制表符和空格了。  
注意，上面sed -n后面的字符是L的小写字母哦，不要看错。

Linux输出重定向>和>>

>:会清空目标文件的内容，文件不存在则创建

>>:会追加内容到目标文件末尾，文件不存在则创建

Linux的tar命令

http://www.cnblogs.com/li-hao/archive/2011/10/03/2198480.html

Linux的定时任务crontab

http://blog.csdn.net/ethanzhao/article/details/4406017

http://hougbin.iteye.com/blog/799335

Oracle的硬解析和软解析

http://blog.csdn.net/tianlesoftware/article/details/5458896

1.语法检查：有没有错别字

2.语义检查：所查的表是否存在，是否有权限

3.解析：1）语法分析 2）权限与对象检查 3）在共享池中检查是否有完全相同的之前完全解析好的 4）选择执行计划 5）产生执行计划 //硬解析会全部执行，软解析执行1）-3）

其中3）验证语句是否一致；验证游标是否一致；验证执行环境是否一致（/\*+ first\_rows \*/的HINT）

4.执行sql

redis的实际应用场景

http://kenny7.com/2012/09/redis-usage-scenario.html

Oracle函数和存储过程的区别



shell的uniq：去除上下重复的行，可统计重复次数

http://os.51cto.com/art/201107/274991.htm

shell的tr：遍历文档内容，可以替换非匹配的内容，压缩相邻的重复字符，大小写转换，删除字符

http://www.2cto.com/os/201109/104590.html

oracle各种游标：

http://www.blogjava.net/cheneyfree/archive/2008/07/31/219138.html

1.显示游标

cursor get\_gsmno\_cur (p\_nettype in varchar2) is   
             select gsmno from gsm\_resource where nettype=p\_nettype and status='0';

open get\_gsmno\_cur('138');

2.隐式游标

 for rec in (select gsmno,status from gsm\_resource)   
 loop  
              dbms\_output.put\_line(rec.gsmno||'--'||rec.status);  
        end loop;

3.REF cursor 用于返回结果集

type rc is ref cursor;

l\_cursor rc;

open l\_cursor for 'select \* from emp';

以上为弱类型，以下为强类型

 type gsm\_rec is record(  
                gsmno varchar2(11),  
                status varchar2(1),  
                price number(8,2));  
           my\_rec gsm\_rec;   
             
    type app\_ref\_cur\_type is ref cursor /\*return gsm\_rec可加可不加，不影响执行结果\*/;  
           my\_cur app\_ref\_cur\_type;

open my\_cur for select gsmno,status,price from gsm\_resource where store\_id='SD.JN.01';  
          fetch my\_cur into my\_rec;