1. **怎么解决session一致性缓存的问题**

博客中第0组的第7篇里面有详解。

1. **Redis的优势和特点**

博客中第8组的第1、2篇里面有详解。

1. **一千万用户并发抢购，怎么设计**

博客中第0组的第3-5篇里面有详解。

1. **如果成功的用户有10万，redis存不下怎么处理**

博客中第8组的第5-9篇里面有详解。

1. **JVM垃圾回收，新生代和老年代的垃圾回收，可达性分析**

博客中第5组的第8篇里面有详解。

1. **HashMap和Hashtable区别，HashMap添加的对象为什么要重写equals和hashcode**

博客中第3组的第4-8篇里面有详解。

1. **数据库了解多少，事务acid，事务操作，怎么加锁**

博客中第7组的里面有详解。

1. **微服务了解什么，微服务治理，微服务之间通信**

博客中第6组的第29-35篇里面有详解。

1. **介绍spring中的熟悉的注解**

博客中第6组的第16篇里面有详解。

1. **Redis和mysql的区别**

博客中第7组的第1篇里面有详解。

1. **Redis的持久化有哪些方式，具体原理**

博客中第8组的第4篇里面有详解。

1. **自己写一个tomcat服务器，你会怎么写**

博客中第11组的第3篇里面有详解。

1. **让你实现autowire注解的功能你会如何实现**

博客中第6组的第6篇里面有详解。

1. **tomcat修改jvm内存大小？**

Java虚拟机；**tomcat不是直接运行在物理操作系统上，而是运行在Java虚拟机上**，通常说的配置jvm就是配置分配给Java虚拟机的内存大小；

如果使用vmware或virtualbox那么给虚拟机分配内存那是很好理解的，可以理解成jvm这个虚拟机就运行tomcat一个进程，所有jvm的内存全给tomcat使用。

**JVM就是运行java的，tomcat就是你的java程序，**servlet就是你写在容器（如tomcat）里的动态资源，远程客户端点击或者一个url请求会封装成一个http请求，到你的服务端后由容器解析http请求，并封装为java servlet请求和响应，最后放进servlet的service方法根据请求作出服务。所以最最简单的话tomcat有类似netty的处理网络的网络功能，和处理servlet的功能。

情况一:解压版本的Tomcat, 要通过startup.bat启动tomcat才能加载配置

要添加在tomcat 的bin 下catalina.bat 里

rem Guess CATALINA\_HOME if not defined  
set CURRENT\_DIR=%cd%后面添加,红色的为新添加的.

set JAVA\_OPTS=-Xms256m -Xmx512m -XX:PermSize=128M -XX:MaxNewSize=256m -XX:MaxPermSize=256m -Djava.awt.headless=true

情况二:安装版的Tomcat下没有catalina.bat

windows服务执行的是bin/tomcat.exe.他读取注册表中的值,而不是catalina.bat的设置.

修改注册表HKEY\_LOCAL\_MACHINE/SOFTWARE/Apache Software Foundation/Tomcat Service Manager/Tomcat5/Parameters/JavaOptions  
原值为  
-Dcatalina.home="C:/ApacheGroup/Tomcat 5.0"  
-Djava.endorsed.dirs="C:/ApacheGroup/Tomcat 5.0/common/endorsed"  
-Xrs

加入 -Xms300m -Xmx350m   
重起tomcat服务,设置生效

///////////////////////////////////////////////////////////////////////////////////////////////

Windows下，在文件/bin/catalina.bat，Unix下，在文件/bin/catalina.sh的前面，增加如下设置：  
JAVA\_OPTS='-Xms【初始化内存大小】 -Xmx【可以使用的最大内存】'  
需要把这个两个参数值调大。例如：  
JAVA\_OPTS='-Xms256m -Xmx512m'

1. **servlet如何获取web.xml中配置的参数？**

**步骤：1.配置参数到Web.xml中。**

**2.在各自的servlet中调用api进行参数的获取。**

  在项目中，**有一些经常使用的代码，我们需要保存到配置文件里**，在Web.xml中提供了良好的方法进行参数的读取

一、Web.xml中参数的设定

(1)设置全局参数（编码），该参数所有的Servlet都可以访问

**[html]** [view plain](https://blog.csdn.net/suwu150/article/details/51535353) [copy](https://blog.csdn.net/suwu150/article/details/51535353)

1. <!-- 设置全局参数 -->
2. **<context-param>**
3. **<param-name>**encoding**</param-name>**
4. **<param-value>**utf-8**</param-value>**
5. **</context-param>**

(2)设置局部参数，该参数只有相应的Servlet能够访问

**[html]** [view plain](https://blog.csdn.net/suwu150/article/details/51535353) [copy](https://blog.csdn.net/suwu150/article/details/51535353)

1. <!-- 设置局部参数 -->
2. **<servlet>**
3. **<servlet-name>**InitServlet**</servlet-name>**
4. **<servlet-class>**servlets.InitServlet**</servlet-class>**
5. **<init-param>**
6. **<param-name>**driverClassName**</param-name>**
7. **<param-value>**sun.jdbc.odbc.JdbcOdbcDriver**</param-value>**
8. **</init-param>**
9. **</servlet>**

此时设置的参数仅在该Servlet中有效，其他Servlet不能访问

**二、获取参数**

获取全局参数的方法如下所示：

**ServletContext application**=this.getServletContext();

**application.getInitParameter("参数名称");**

获取局部参数的方法如下所示：

**this.getInitParameter("参数名称");//此处this指Servlet本身**

用下面的Servlet来接受Web.xml中设置的参数：代码如下所示：

**[html]** [view plain](https://blog.csdn.net/suwu150/article/details/51535353) [copy](https://blog.csdn.net/suwu150/article/details/51535353)

1. package servlets;
2. import java.io.IOException;
3. import java.io.PrintWriter;
4. import javax.servlet.ServletContext;
5. import javax.servlet.ServletException;
6. import javax.servlet.http.HttpServlet;
7. import javax.servlet.http.HttpServletRequest;
8. import javax.servlet.http.HttpServletResponse;
9. public class InitServlet extends HttpServlet {
10. public void doGet(HttpServletRequest request, HttpServletResponse response)
11. throws ServletException, IOException {
12. **/////全局参数调用**
13. ServletContext application = this.getServletContext();
14. String encoding = application.getInitParameter("encoding");
15. System.out.println("encoding参数是：" + encoding);
16. **/////局部参数调用**
17. String driverClassName = this.getInitParameter("driverClassName");
18. System.out.println("driverClassName参数是：" + driverClassName);
19. PrintWriter out = response.getWriter();
20. out.println("driverClassName:"+driverClassName);
21. **//在这儿测试使用获取全局变量的方式获取局部参数**
22. String driverClassName\_test=application.getInitParameter("driverClassName");
23. out.println("driverClassName\_test:"+driverClassName\_test);
24. System.out.println("driverClassName\_test参数是：" + driverClassName\_test);
26. }
27. }

下面就是web.xml配置的全部代码：

**[html]** [view plain](https://blog.csdn.net/suwu150/article/details/51535353) [copy](https://blog.csdn.net/suwu150/article/details/51535353)

1. **<?xml** version="1.0" encoding="UTF-8"**?>**
2. **<web-app** version="2.5" xmlns="http://java.sun.com/xml/ns/javaee"
3. xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4. xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
5. http://java.sun.com/xml/ns/javaee/web-app\_2\_5.xsd"**>**
6. <!-- 设置全局参数 -->
7. **<context-param>**
8. **<param-name>**encoding**</param-name>**
9. **<param-value>**utf-8**</param-value>**
10. **</context-param>**
11. <!-- 设置局部参数 -->
12. **<servlet>**
13. **<servlet-name>**InitServlet**</servlet-name>**
14. **<servlet-class>**servlets.InitServlet**</servlet-class>**
15. **<init-param>**
16. **<param-name>**driverClassName**</param-name>**
17. **<param-value>**sun.jdbc.odbc.JdbcOdbcDriver**</param-value>**
18. **</init-param>**
19. **</servlet>**
20. **<servlet-mapping>**
21. **<servlet-name>**InitServlet**</servlet-name>**
22. **<url-pattern>**/InitServlet**</url-pattern>**
23. **</servlet-mapping>**
24. **<session-config>**
25. **<session-timeout>**
26. 30
27. **</session-timeout>**
28. **</session-config>**
29. **</web-app>**

下面我们进行测试：

由于全局代码设置的encoding为utf-8，故显示结果为utf-8，而局部代码设置的是sun.jdbc.odbc.JdbcOdbcDriver，照样显示，**而当使用全局代码的方式去查看局部代码是，取到的值为null**,具体如下图所示：

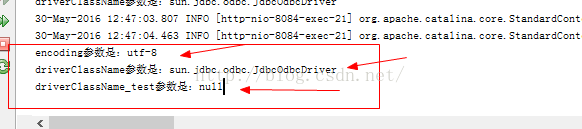


图1 显示结果

以上就是使用Servlet获取web.xml中的参数。

1. **分布式服务器会出现哪些问题**

**1.分布式session共享问题；**

**（引入项目中单点登录例子解决）**博客中第0组的第7篇里面有详解。

**2.分布式事务的问题。**

**CAP等内容，**博客中第7组的第24篇里面有详解。