
ParnassusData™

# TomCat 迁移步骤简述以及案例

吕晶

# 目录

# 背景

- Z电信是省内最大的电信业务处理平台
  - 7×24工作，平均在线600左右，峰值超过1000

- 通过60人天的工作，完成了所有8个应用的迁移工作
  - 完成了功能和压力测试
  - 通过迁移，基本消除了连接/操作失败问题，并使效率提高30%。

# 目录

# 准备工作 – 安装

- 安装Tomcat
  - 版本选择
- 测试环境安装与配置
  - 数据库
  - jms
  - etc...

# 准备——开发工具

- 使用基于Eclipse的开发工具
  - 集成Tomcat测试环境
- 集成测试工具
- 性能测试工具

# 准备 - 测试机器拓扑

- 单机测试环境
- 集群测试与负载平衡

# 迁移准备

- 设置环境
- 在开发工具中导入war文件
  - 选择可以展开的项目
  - 如有需要，加载java源代码

# 修复J2EE /J2SE资源

- 如果需要，必须先做J2EE/J2SE版本之间的升级
- 例如，Java EE 5+ 的标准规定，带资源annotation的setter方法，必须有一个对应的变量存在
  - Or an error will occur - JSR250: CommonAnnotations for Java, 2.3: @Resource methods must be setters that follow the standard JavaBeans convention. i.e. void "setProperty(<Type> value)" for "<Type> property". DaoHelper.java /exchange/src/gnnt/MEBS/base/dao/jdbc line 37 Annotation Problem
  - 相对应修改并通过编译

# 编译JSPs

- JSP页面可能会存在大量的错误，预编译能够帮助找到静态错误，减少出错几率，提高效率
  - 使用eclipse或其他JSP编译工具
  - 右键点击项目，选择JAVA EE > Compile JSPs
  - 可以加自定义的参数

- JSP碎片问题
  - 如果JSP文件是一个JSP碎片，则它不能单独编译
  - 建议重命名为“*.jspf” (使用“重构”-> 重命名，或 F2

# 定义数据源

- 数据源可能分散在不同的文件里
  - 检查数据源定义
- 不同web.xml版本的数据源可能会有不一致的情况出现

# 部署并运行

- 做简单的测试，找到更多的问题
- 通过查看应用提供的log查找问题所在

# 案例：Cookie问题

- 首先，页面中出现了cookie功能失效的问题
  - 问题在于，javascript函数实现有问题
  - 该函数会在第一个cookie中查找以JSESSION的key。由于Tomcat行为的不一致，它出现在第二个cookie中，导致该函数判断出错
  - 该函数的实现会因不同服务器，不同浏览器版本等失效，必须修改
- 以下是标准的查看cookie功能是否开启的函数

```
function CookieEnable()
 {
    // DHV - added this code for checking if cookies are enabled
    var cookieEnabled = (navigator.cookieEnabled) ? true : false;
    if (typeof navigator.cookieEnabled == "undefined" && !cookieEnabled)
  {
   document.cookie="testcookie";
   cookieEnabled = (document.cookie.indexOf("testcookie") != -1) ? true : false;
  }
    return (cookieEnabled);
 }
```

# 案例： FLEX问题

- **客户端无法注册到subtopic上**
- **解决方案: 对于同一个**destination ， 只需要创建channel一次。

- `common.service.ConsumerService.as`
  - Defines a global variable to hold the static ChannelSets
    - `private var _channelSets:ChannelSet = new ChannelSet();`
  - Populate the _channelSets in the constructor of the ConsumerService (This only runs once, since ConsumerService is static)
    - `var consumerAddr:String = CacheDatas.serviceConfig.consumerAddr.toString()`
    - `_channelSets.addChannel(new StreamingAMFChannel("my-streaming-amf",consumerAddr));`
  - Remove the creation of a new Channel in the subscribe() method and simply pass the _channelSets to the Consumer
    - `cs.channelSet = _channelSets;`

# 案例：线程问题

- 多线程问题必须注意
  - 在J2EE中，避免使用额外的线程
  - 如果有必要，则尽可能少使用线程
  - 避免在start()等函数中重新开线程（可能造成极大的泄漏）
  - 避免主线程等待在子线程上——这将造成死锁

- 现有修改：

- Step 1 StatusMsgSendThread 查找DB中的信息
- Step 2 如果有，则发信息到jms/mrTopic
- Step 3 DevMessageThread 在 jms/mrTopic,上监听，如果有信息，则加到 DevMessageDataCache
- Step 4 DevMessagePushMainThread每隔一秒钟查看DevMessageDataCache
- Step 5 如果有信息，DevMessagePushThread移除信息并向客户端发消息

- 未来改进：
  - 移除线程
  - 使用线程池

# 其他问题

- 注意每一个线程需要去读正确的cache

```
@Override
public void task() throws Exception {
  IMessageProducer messageProducer =
(IMessageProducer)ApplicationContextUtil.getContext().getBean("messageProducerImpl");
  IMessageCreator messageCreator = null;
  Map map = DevMessageDataCache.remove();
```

Should be

```
  Map map = PassengerMessageDataCache.remove();
```

# 案例：第三方类库DWR配置

- 发现DWR动态JS生成工具无法正常工作
  - 错误为，无法找到DWR生成的JS
  - 问题在于，在配置文件中，Bean ID和所对应的类未能正确定义
- 使用org.springframework.web.context.ContextLoaderListener方式
  - 有新的mapping方式
- 首先查看 WEB-INF/web.xml文件

```
<context-param>
<param-name>contextConfigLocation</param-name>
<param-value>classpath:somexml.xml</param-value>
</context-param>
<listener>
<listener-class>
org.springframework.web.context.ContextLoaderListener
</listener-class>
</listener>
```

# 定义DWR

- 查找DWR定义

```
<!--dwr servlet-->
<servlet>
<servlet-name>dwr-invoker</servlet-name>
<servlet-class> org.directwebremoting.servlet.DwrServlet
</servlet-class>
<init-param>
    <param-name>debug</param-name>
    <param-value>true</param-value>
</init-param>
</servlet>
<servlet-mapping>
    <servlet-name>dwr-invoker</servlet-name>
    <url-pattern>/dwr/*</url-pattern>
</servlet-mapping>
```

- 查看在dwr.xml中需要哪些bean
  - 对exchange来说，需要有五个
  - *checkIdAction， customerAdd， memberInfoTreeDwr，*
  - *noticePromptAction， settleStatusAction*

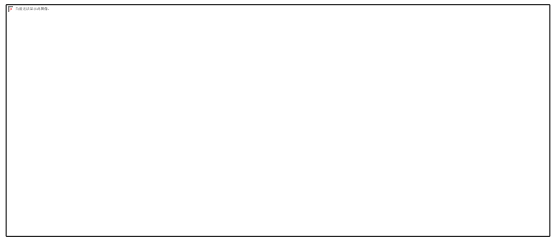
# 增加bean映射关系

- In account_default.xml
  - `<bean id="checkIdAction" class="account.action.CheckIdAction" autowire="byName"></bean>`
  - `<bean id="customerAdd" class="gnnt.MEBS.account.dwr.CustomerAdd" autowire="byName"></bean>`
- In broke_default.xml
  - `<bean id="memberInfoTreeDwr" class="broke.action.dwr.MemberInfoTreeDwr" autowire="byName"></bean>`
- In announcement_default.xml
  - `<bean id="noticePromptAction" class="announcement.dwr.NoticePromptAction" autowire="byName"></bean>`
- In settlement_default.xml
  - `<bean id="settleStatusAction" class="settlement.dwr.SettleStatusAction" autowire="byName"></bean>`

# 案例：Ecside框架的兼容修改

- ECSide是有一个基于jsp tag的开源列表组件. 简单的说,它就是一组可以帮助快速实现强大的列表的jsp标签

- 有许些兼容问题
  - 包括中文乱码，cvs/pdf等导出问题



- 官方的升级导致代码无法重用
  - 无法升级到最新版本
  - 只能通过打补丁的方式，改写其中的类
  - 必须部署到四个后台应用上

# 修改其他的配置信息

- database.xml
  - 该文件比较特殊，必须修改其中的数据库信息
  - 并修改其中的JNDI 名字

- 请注意! 不要使用非标准的JNDI名字
  - 存在: “java:/comp/env/mgr”
  - 应该使用: “java:comp/env/mgr”

# 本地化问题

- 有三个方面的中文乱码问题困扰过我们
  - 普通页面（如jsp/js）的乱码问题
  - 提交请求时中文乱码问题
  - Ecside等框架的乱码问题

- 解决方案有：
  - 保持环境-虚拟机-页面-配置文件的encoding一致
  - 在WebSphere的虚拟机启动参数
    - Java和进程管理-进程定义-Java虚拟机--通用JVM参数中加入：
      -Ddefault.client.encoding=GBK -Dfile.encoding=GBK -Duser.language=Zh -Duser.region=CN
  - 在web.xml中设置通用和框架的filter，并使之encoding保持一致
  - 使用GBK来编译JSPs文件
    - -war.path "${workspace_loc}/exchange/WebContent"
    - -javaEncoding GBK
    - -compileToDir "${workspace_loc}/WebContent/WEB-INF/classes/"

# 总结

- 经过近两个月的工作，我们迁移了所有8个程序
- 所有的程序已经部署在测试平台之上，并经过了功能测试
- 一共发现了36个bug，已经全部修复.

# 目录

- 背景
- Tomcat迁移工作
  - 准备工作
  - 迁移流程
  - 本地化问题
  - 总结
- 迁移工作后的建议
- 附录：迁移技术细节

# 迁移后的建议

- 展开项目，完整编译
  - 版本控制
  - **保证所有的代码编译通过**

- **依赖库分享**
  - **在迁移工作中，发现应用多依赖于各种开发库（DWR，ecside等）**
  - **在不同的应用中，包含了同样的开发库，为一个开发库打补丁，必须放到其他应用程序中**
    - **很可能被忘记**
    - **容易出错**

# 目录

专注于数据

PARNASSUSDATA

软件，方案，服务供应商


ParnassusData™