BeanShell在网管开发中的应用

【摘要】

本文主要介绍了在网管开发中，如何利用 BeanShell 方便网管开发、问题定位和测试的一些方法和技巧。本文最后介绍如何把 BeanShell 集成到网管中。

【关键词】BeanShell　脚本

1. 引言

BeanShell 是非常精悍的脚本语言，最大的特点就是解析执行 Java 代码，而不是编译模式，对网管开发者而言，不需熟悉 Jython 等新的语法规则和 API。网管开发中经常需要获取程序运行中的状态，动态调试、以及生成测试数据等工作，我们可以有效的利用 BeanShell 帮助我们解决这些问题和提高开发效率。

2. BeanShell 应用

a) 做计算器

```
bsh % print(4*5-9*(3-2));
11
bsh % print(3|5);
7
bsh % print(Integer.toHexString(15));
f
bsh % print(Integer.toBinaryString(15));
1111
```

b) 测试 API

```
bsh % String a="abcdefg";
bsh % print(a.substring(0,4));
abcd
bsh % print(a.substring(1,4));
bcd
```

```
bsh % BigInteger big = new BigInteger("1010110101010110");
bsh % print(big.toString());
1010110101010110
bsh % long aLong = 1010110101010110;
// Error: Parser Error: Parse error at line 13, column 14 : Error or
big for integer type: 1010110101010110
bsh % BigInteger big2 = new BigInteger("1010110101010110");
bsh % big3 =big.add(big2)
;
bsh % print(big3.toString());
2020220202020220
bsh % print(big3.toString(2));
1110010110101100001001010111000001001101101011111100
bsh % print(big3.toString(16));
72d612b826d7c
```

我们还可以在真实的运行环境下，测试下 API 的正确性和性能

c) 系统管理功能

pwd() 显示当前所在的目录　mv　移动文件　cp　复制文件　rm 删除文件
通过这些命令，我们可以删除一些不用的配置。

d) 查看程序运行状态

e) 生成测试数据

LLDP 测试需要读取设备的 LLDPmib 数据。在性能测试中，不可能有很多设备，也不可能有可测试的试验环境。我们就写了 LLDP 的模拟器，所有设备的 mib 访问都会从设备中获取。但是每次都需要重新生成测试数据，都需要重新代码和编译，还有个缺点是测试数据有些是错误。测试数据起初使用 MBean 方式，重新生成，但是感觉还是不灵活，因为每次都需要在 MBean 添加新的方法，以便可以调用。

A. 生成测试数据

```
import  com.zte.ums.ip.commonip.lldp.emf.DataDb;
import  com.zte.ums.ip.commonip.lldp.emf.DeviceMib;
import  com.zte.ums.ip.commonip.lldp.common.LldpMibConstants;

DeviceMib[] mibs = devices.values().toArray(new DeviceMib[0]);
int size = mibs.length;
for(int i = 0; i < size; i++)
{
DeviceMib d = mibs[i];
// 初始化设备信息
d.putOidValue(LldpMibConstants.MIB_LLDP_LOC_CHASSISID_SUBTYPE, "4");
d.putOidValue(LldpMibConstants.MIB_LLDP_LOC_CHASSIS_ID, d.getMac());
d.putOidValue(LldpMibConstants.MIB_LLDP_LOC_SYSNAME, d.getIp());
d.putOidValue(LldpMibConstants.MIB_LLDP_LOC_SYSDESC, d.getIp());
}
// 生成链路
for(int i = 0; i < size - 1; i++)
{
DeviceMib.addLink(mibs[i], mibs[i + 1]);
}
DeviceMib.addLink(mibs[0], mibs[size - 1]);
```

B. 显示所有的测试数据

```
import  com.zte.ums.ip.commonip.lldp.emf.DataDb;
import  com.zte.ums.ip.commonip.lldp.emf.DeviceMib;
int size = DataDb.devices.size();
System.out.println(size);
//System.out.println(DataDb.devices.get("192.168.1.1").getIp());
Iterator iterator = DataDb.devices.values().iterator();
while(iterator.hasNext())
{
DeviceMib device = (DeviceMib)iterator.next();
System.out.println(device.getIp());
}
```

C. 查看测试数据的正确性

```
import  com.zte.ums.ip.commonip.lldp.emf.DeviceMib;
DeviceMib device = DataDb.devices.get("7.7.7.6");
System.out.println(device.getIp());
System.out.println(device.getMac());
```

f) 修改程序

通常可以使用 JMX 方式，启动服务、停止服务。或者其他的 api 接口，但是有个缺陷就是不是所有的 api 都支持这个方式，实际上只有很少的接口支持 jmx 方式。

在调试状态下，通过 hotcode switch 方式修改。这种需要打开调试模式，一般只在调试模式下运行。

通过配置文件方式，修改业务流程。这个需要很多支持才行。每次都要重新获取，这个开销比较大。

本文使用 BeanShell。

a. 修改日志级别

```
bsh % print(org.apache.log4j.Logger.getRoot().getLevel());
DEBUG
bsh % org.apache.log4j.Logger.getRoot().setLevel(org.apache.log4j.Level.INFO);
bsh % print(org.apache.log4j.Logger.getRoot().getLevel());
INFO
```

b. Release 和 Debug 模式切换

```
bsh % print(com.zte.ums.ip.api.commonip.common.linkmng.IpDebug.IS_TEST);
true
bsh % com.zte.ums.ip.api.commonip.common.linkmng.IpDebug.IS_TEST=false;
bsh % print(com.zte.ums.ip.api.commonip.common.linkmng.IpDebug.IS_TEST);
false
```

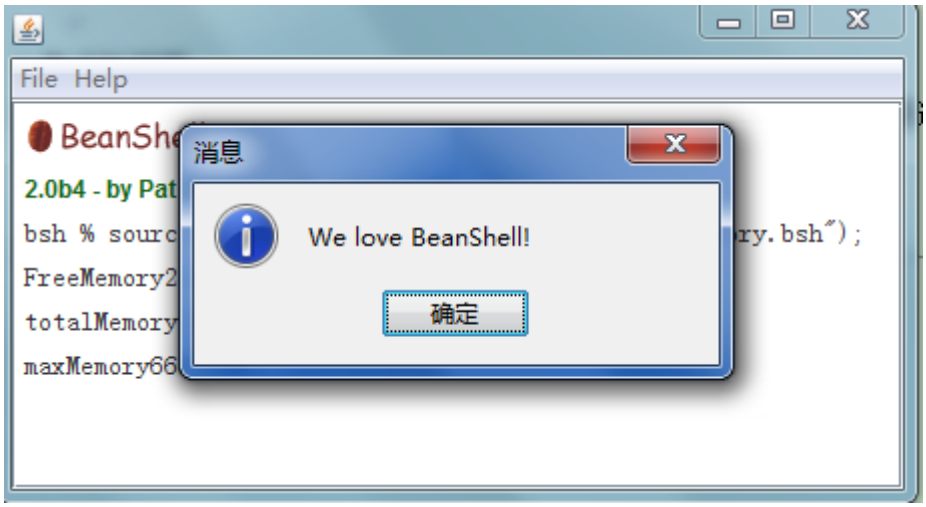
g) 其他应用

你还可以 集成 ant 脚本中，生成可以 Realease 和 Debug 不通模式的代码。
自动下载项目的代码，构建开发环境。

3. 集成到网管

beanshell 提供了两种集成方式，可以本地集成到 Swing，或者远程通过 telnet 或者 web 访问。

使用 bsh.util.JConsole 集成到 Swing 中，JConsole 是个 Swing 组件，需要在 JDialog 或者 JFrame 中。

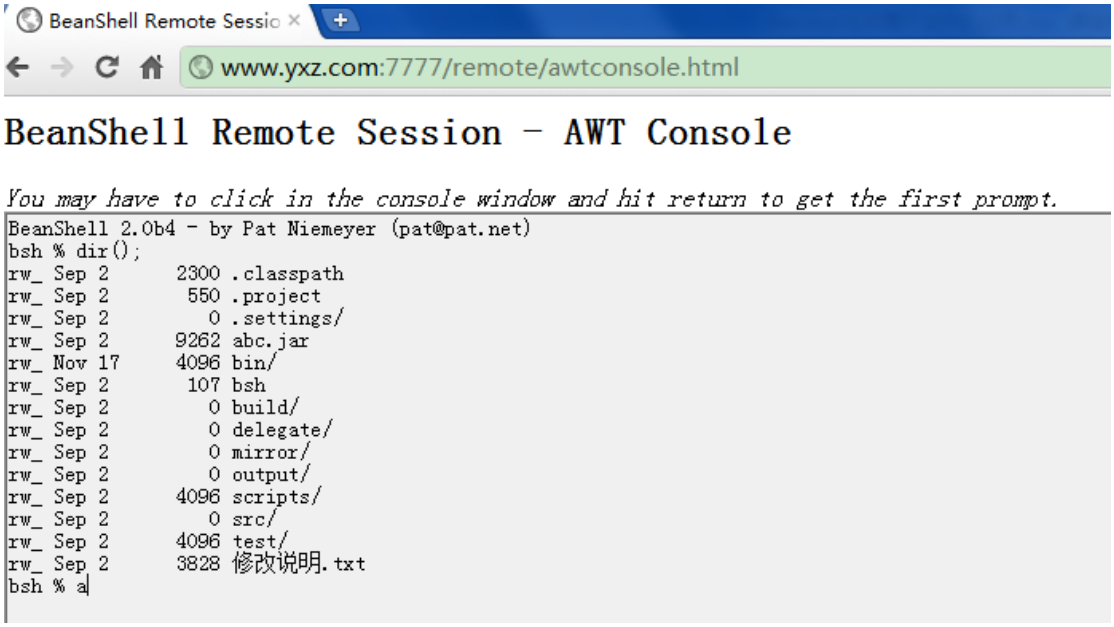


本地集成，可以充分扩展 BeanShell。比如打开本机脚本等。
由于网管的服务端没有加载 Swing，不支持本地集成 BeanShell 的功能，可以通过 Web 或者 Telnet 方式来访问。
在实际应用中，我们需要给每个进程都分配不通的端口号。我们通过 MBean 调用这个服务，运行 BeanShell 解析器，然后运行 BeanShell 内置的 server(listenerPort)方法进行端口监听。

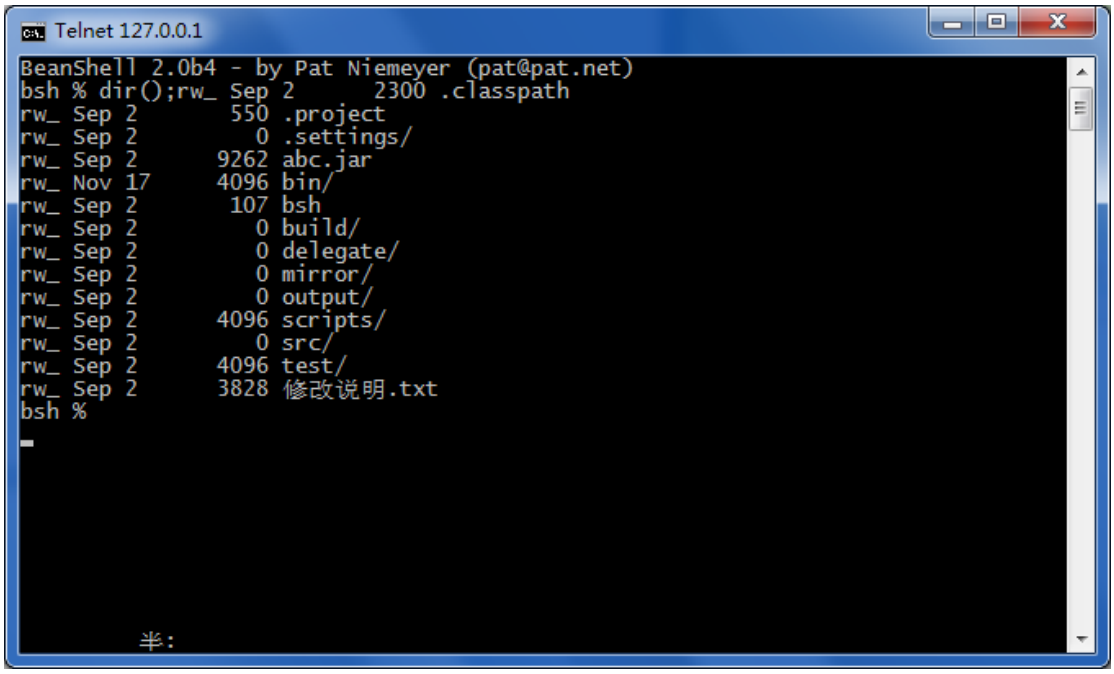
```
public class BeanShellWeb
{
   public static Interpreter interpreter  = new Interpreter();
   public static int         port         = 7777;
   public static int         retryCount   = 0;
   public static int         maxRetryCount = 10;

   // 多进程调用，连续地址
   public static void start()
   {
      while(true && retryCount < maxRetryCount)
      {
         try
         {
            interpreter.eval("server(" + (port + retryCount) + ")");
            DebugUtil.info(port + retryCount + " allocated successfully");
            // 成功分配
            break;
         }
         catch(EvalError e)
         {
            e.printStackTrace();
            DebugUtil.info(port + retryCount + "has allocated");
            retryCount++;
         }
      }
   }
```

WEB 方式访问



远程到这个地址　telnet　127.0.0.1:7778 比 web 的端口号多一个



4. 参考文献

http://www.beanshell.org