Gerenciamento JMX

Ivan Francolin Martinez

MAC 5863
Sistemas de Middleware Avançados
Prof. Francisco Reverbel

30/11/2006

Objeto MBean

- Um objeto gerenciavel via JMX
- Representa um dispositivo, aplicação ou qualquer outro recurso
- Expõe informações sobre suas características (Meta Dados, `MBeanInfo`)
- Possui atributos e/ou operações
- Representado por um nome único

Atributos

- Permite alterar o estado interno do MBean
- somente leitura, somente escrita ou ambos
- Pode ser um tipo primitivo, ou objeto
- Recomendável `Serializable`

Operações

- Permite executar operações no objeto
- Pode receber parâmetros
- Pode ter valor de retorno

Notificações

- Permite que o gerenciador seja informado sobre mudanças no MBean
- Informações devem ser incrementais, sobre mudanças no estado

Standard MBean

- Definido facilmente a partir de uma interface
- Uma classe XXX deve estender uma interface XXXMBean que define os atributos e operações.
- Um atributo é definido a partir de métodos getYYY e setYYY
- Todos outros métodos são operações

Standard MBean

```
public interface HelloMBean {

    public void sayHello();
    public int add(int x, int y);

    public String getName();

    public int getCacheSize();
    public void setCacheSize(int size);
}
```

MXBean

- Versão melhorada do Standard MBean
- Uma classe XXX deve extender uma interface XXXMXBean que define os atributos e operações.
- Um Standard MBean que utiliza um conjunto limitado de tipos
- Deve utilizar tipos definidos no pacote `javax.management.openmbean`
- Tipos complexos são mapeados utilizando `CompositeDataSupport`
- Inicialmente utilizados no Java 1.5, mas somente na versão 1.6 foi incluído suporte para criação de novos MXBeans

Dynamic MBean

- Definido implementando a interface `javax.management.DynamicMBean`
- Todo tratamento deve ser implementado manualmente

interface DynamicMBean

```
package javax.management;

public interface DynamicMBean {

    public Object getAttribute(String attribute)
        throws AttributeNotFoundException,
        MBeanException, ReflectionException;

    public void setAttribute(Attribute attribute)
        throws AttributeNotFoundException,
        InvalidAttributeValueException, MBeanException,
        ReflectionException ;

    public AttributeList getAttributes(String[] attributes);

    public AttributeList setAttributes(AttributeList attributes);

    public Object invoke(String actionName, Object params[],
                         String signature[])
        throws MBeanException, ReflectionException ;

    public MBeanInfo getMBeanInfo();
}
```

Open MBean

- Dynamic MBean
- Segue padronização que permite maior flexibilidade
- Somente tipos primitivos, classes básicas e classes JMX
- Não necessita de ajustes no CLASSPATH

Model MBean

- Implementa a interface `javax.management.modelmbean.ModelMBean`
- Funciona como ponte para um objeto real
- Mapeamento para métodos de um objeto POJO

ObjectName

- Identifica de forma única cada MBean
- Permite selecionar objetos a partir de uma mascara
- formato :
  `domínio:pares chave/valor`
  `domínio:chave1=valor1,chave2=valor2,...`

Identificando um objeto

- `java.lang:type=Runtime`
- `java.lang:type=MemoryPool,name=Perm Gen`
- `Users:type=Role,rolename=tomcat,database=UserDatabase`

Selecionando objeto(s)

- `java.lang:*`
- `java.lang:type=MemoryPool,*`
- `*:type=ClassLoading`
- também existe a possibilidade de utilizar `QueryExp` para seleção

MBeanServer

- Mantém referencias para MBeans
- intermediário entre o sistema gerenciador e os objetos reais
- pode ser acessado localmente ou remotamente utilizando Conectores
- criado a partir de `MBeanServerFactory`

Principais métodos do MBeanServer

```
ObjectInstance registerMBean(Object object,
     ObjectName name)
ObjectInstance createMBean(String className,
   ObjectName name)
void unregisterMBean(ObjectName name)

MBeanInfo getMBeanInfo(ObjectName name)

Set queryMBeans(ObjectName name, QueryExp query)
Set queryNames(ObjectName name, QueryExp query)
String[] getDomains()

Object invoke(ObjectName name, String operationName,
              Object[] params, String[] signature)
```

Principais métodos do MBeanServer

```
Object getAttribute(ObjectName name, String attribute)
AttributeList getAttributes(ObjectName name,
                            String[] attributes)

void setAttribute(ObjectName name, Attribute attribute)
AttributeList setAttributes(ObjectName name,
                            AttributeList attributes)

void addNotificationListener(ObjectName name,
                             NotificationListener listener,
                             NotificationFilter filter,
                             Object handback)
void removeNotificationListener(ObjectName name,
                                ObjectName listener)
```

Conectores

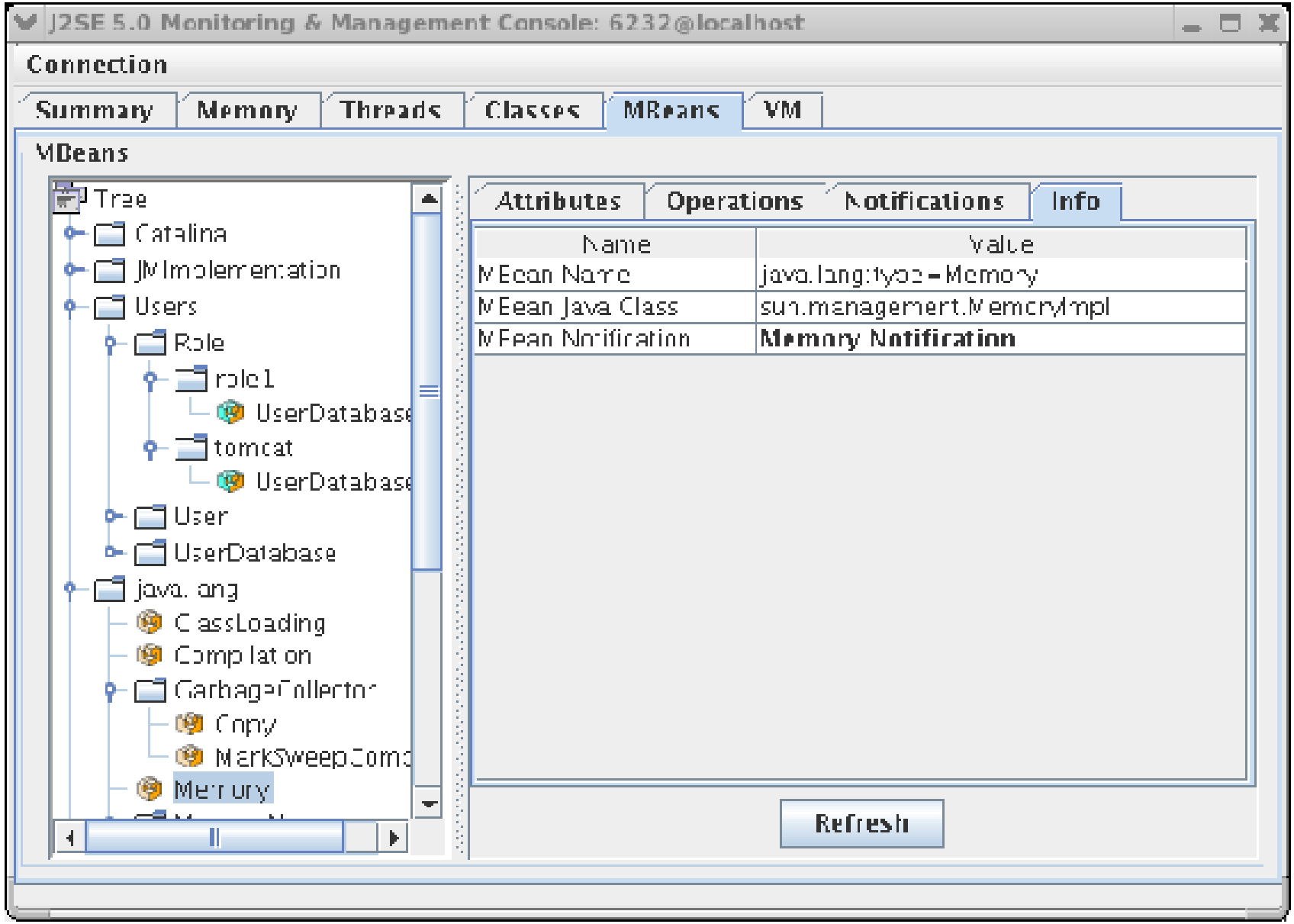
- Permitem acesso remoto ao `MBeanServer`
- JMX Remote API (JSR-160)
- RMI
- JMXMP
- *WebServices* (JSR-262 em desenvolvimento)

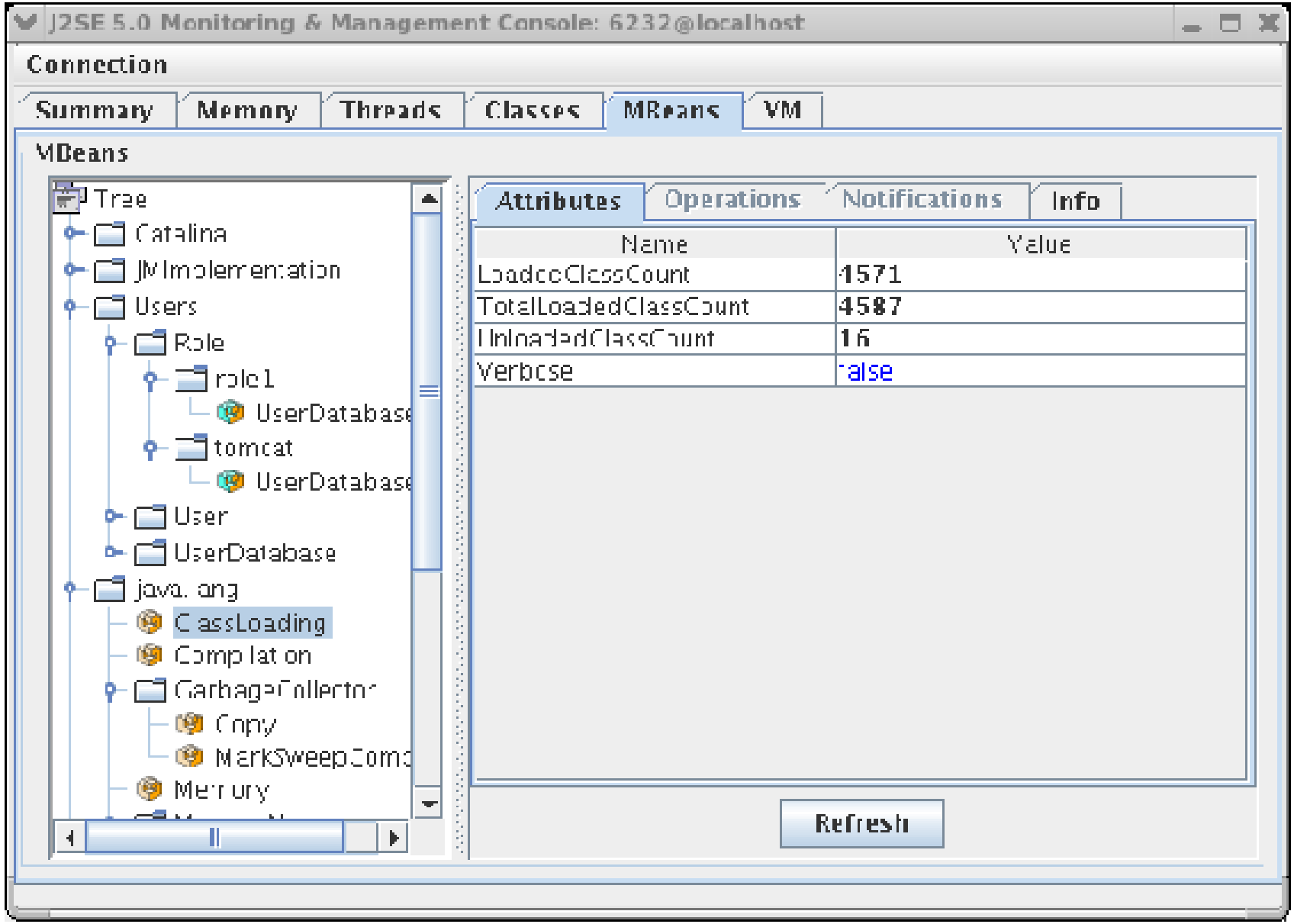
Adaptadores

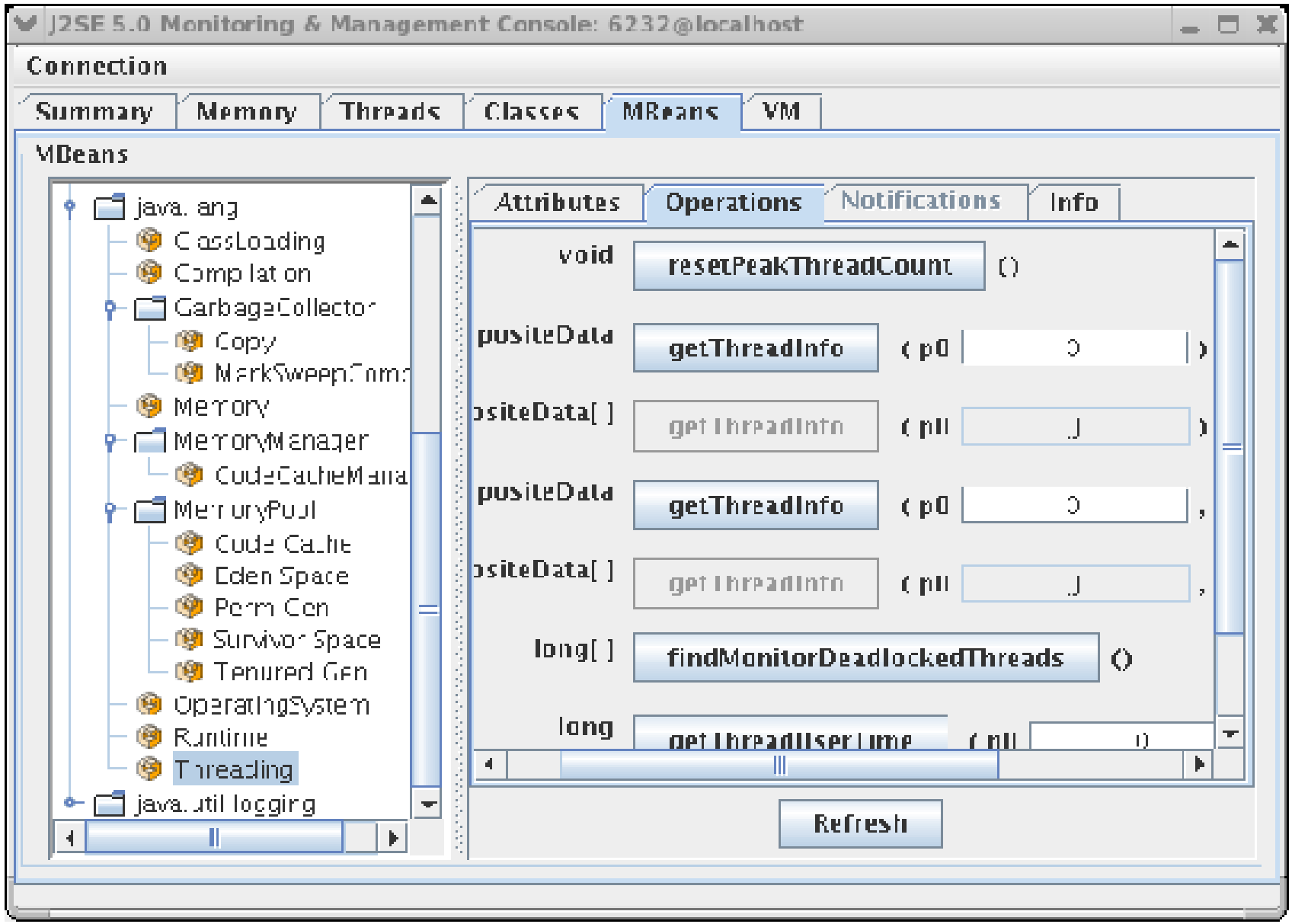
- Permitem o acesso a partir de sistemas/clientes que não suportam diretamente JMX.
- SNMP
- HTTP

JConsole

- Cliente padrão disponível com o Java 1.5
- GUI


J2SE 5.0 Monitoring & Management Console: 6232@localhost
Connection
Summary
Memory
Threads
Classes
MBeans
VM
MBeans
Tree
Catalina
JMImplementation
Users
Role
role1
UserDatabase
tomcat
UserDatabase
User
UserDatabase
java.lang
ClassLoading
Compilation
Memory
Attributes
Operations
Notifications
Info
Name
Value
MBean Name
java.lang:type=Memory
MBean Java Class
sun.management.MemoryImpl
MBean Notification
Memory Notification
Refresh


J2SE 5.0 Monitoring & Management Console: 6232@localhost
Connection
Summary
Memory
Threads
Classes
MBeans
VM
MBeans
Tree
Catalina
JMImplementation
Users
Role
role1
UserDatabase
tomcat
UserDatabase
User
UserDatabase
java.lang
ClassLoading
Compilation
Copy
Mercury
Attributes
Operations
Notifications
Info
Name
Value
LoadedClassCount
1571
TotalLoadedClassCount
4587
UnloadedClassCount
16
Verbose
false
Refresh


J2SE 5.0 Monitoring & Management Console: 6232@localhost
Connection
Summary
Memory
Threads
Classes
MBeans
VM
MBeans
java.lang
ClassLoading
Compilation
GarbageCollector
Copy
MarkSweepCompact
Memory
MemoryManager
CodeCacheManager
MemoryPool
Code Cache
Eden Space
Perm Gen
Survivor Space
Tenured Gen
OperatingSystem
Runtime
Threading
java.util.logging
Attributes
Operations
Notifications
Info
void
resetPeakThreadCount
()
getThreadInfo
getThreadInfo
getThreadInfo
getThreadInfo
long[]
findMonitorDeadlockedThreads
()
long
Refresh

MX4J

- Implementação de JMX (JSR-3)
- Implementação de JMX Remote API (JSR-160)
- Open Source, pode ser utilizado em Java 1.4
- Possui Adaptador HTTP

Tomcat Ant Tasks

- Permite automatizar acesso JMX
- Sem necessidade de código Java
- Utilizando arquivos XML ant

Java 1.6

- JMX 1.4
- *Generics*
- suporte para "chave=*" em `ObjectName`

Java 1.7

- JMX 2.0
- Utilização de anotações
- Suporte para *Cascading*

- http://java.sun.com/javase/technologies/core/mntr-mgmt/javamanagement/
- http://java.sun.com/docs/books/tutorial/jmx/index.html
- http://mx4j.sourceforge.net/

Perguntas ?

- Dúvidas

Gerenciamento JMX

Ivan Francolin Martinez

MAC 5863
Sistemas de Middleware Avançados
Prof. Francisco Reverbel

30/11/2006