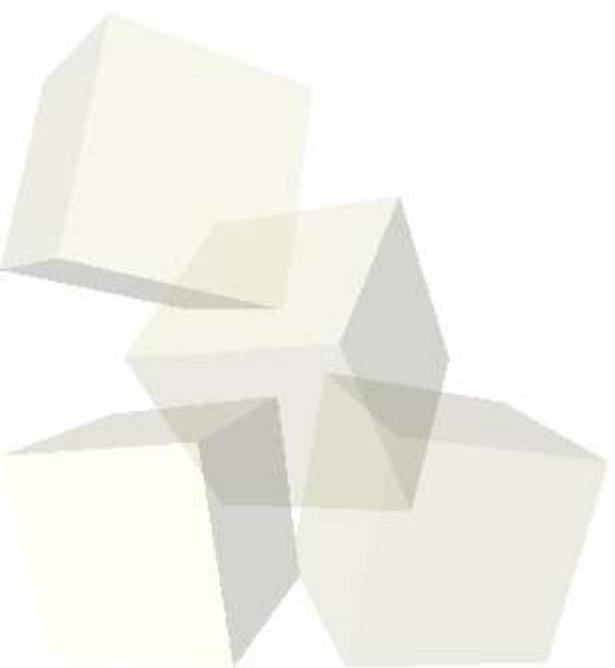




Java Management Extensions

Alexander Nägele
Benjamin Bratkus
Tobias Walter
Uwe Grözinger





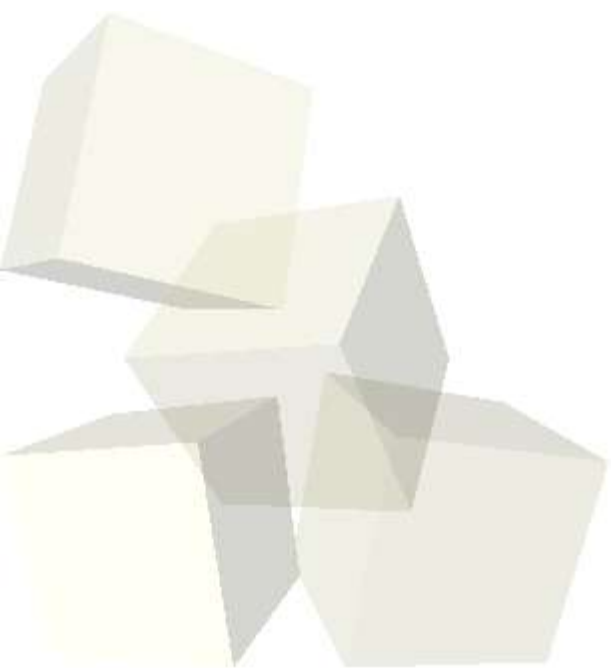
- Einführung
- Architektur
- Managed Beans
- MBean-Server
- Agent Services
- Fazit

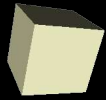




Was ist JMX ?

- Verwaltung von Java-Anwendungen und -Diensten
- Spezifikation durch den Java Communication Process
- Spezifikation beschreibt die Architektur sowie Verwaltungs- und Monitoring-Dienste für Java





Woraus entstand JMX ?

- Grosse Rolle von Java in der Entwicklung verteilter Systeme
- Durch fortschreitende Konnektivität wurde Standard benötigt, solche Gebilde zu managen
- Heutige Anwendungen sind nahezu unkalkulierbar
- Notwendigkeit eines dynamischen Managements von Java-Komponenten





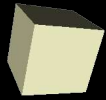
Woraus entstand JMX ?

- Beteiligung grosser Firmen wie IBM, Apache Group, BEA Systems und Borland
- Verwendung bei JBoss, Tomcat, IBM Websphere und BEA Weblogic



- Wird ab JDK 1.5 fest implementiert

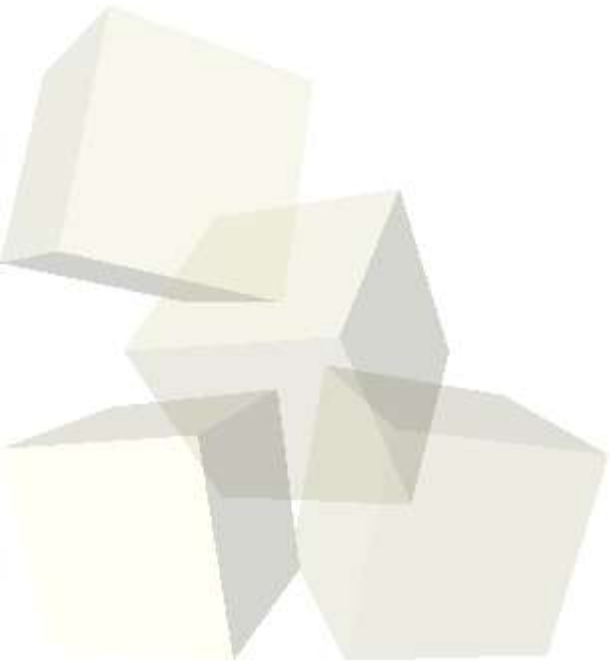


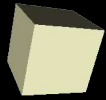


- JMX implementiert eine SNMP-Manager-API (javax.management.snmp)
- Einfacher Zugriff auf SNMP-Agents möglich
- Support von SNMP-Abfragen
- Kann mit SNMPv1 und SNMPv2 Traps umgehen
- Erlaubt Security und Verschlüsselung



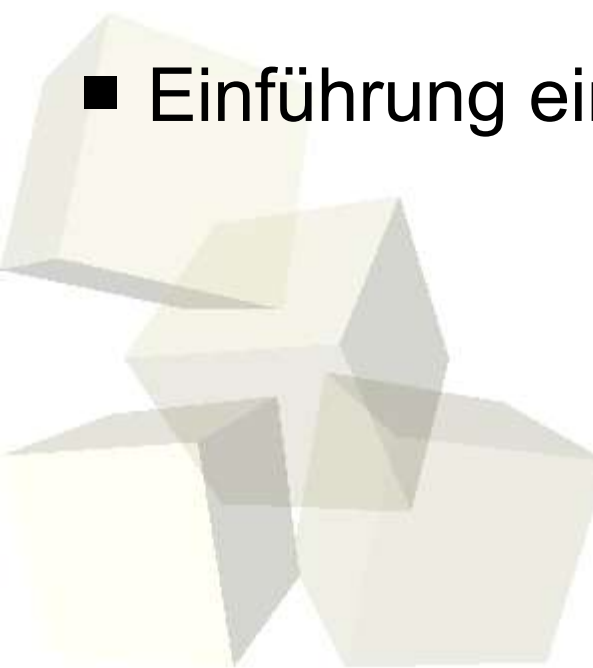
- Instrumentation Level
- Agent Level
- Distributed Services Level

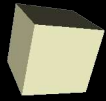




Instrumentation Level

- Beinhaltet zu überwachende Ressourcen, bspw. Anwendung, Dienst oder User
- Repräsentation dieser Ressourcen durch MBeans
- MBean kapselt Ressource und stellt Schnittstelle zur Verfügung
- Einführung eines Notifikationsmechanismus



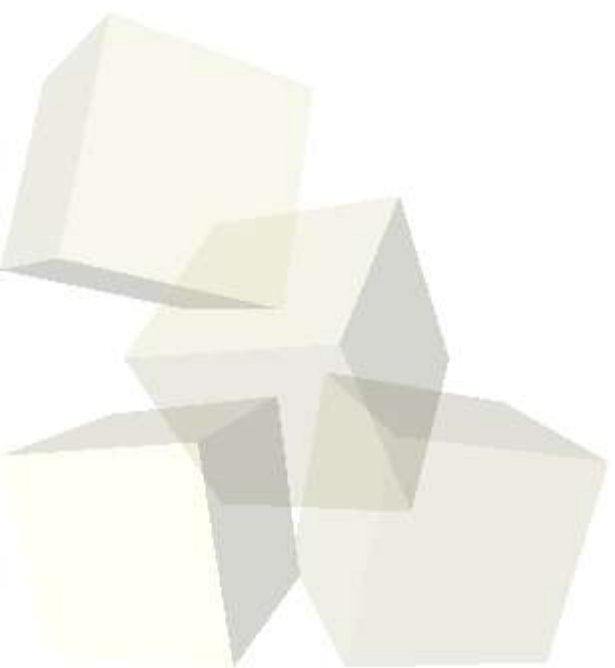


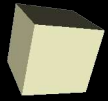
- Definiert Agenten
- Clientzugriff auf Ressourcen über die Agenten
- Agent besteht aus MBean Server und Agent Services
- Zugriff auf MBean Server über Kommunikationsadapter und Konnektoren





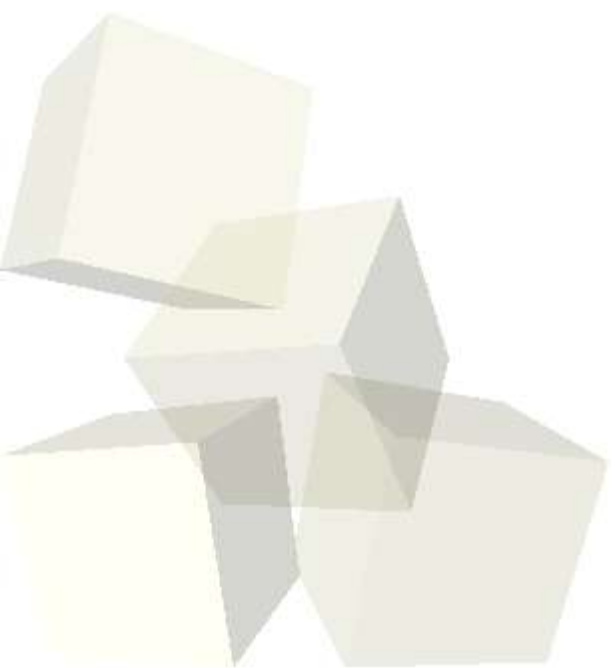
- MBean Server ist zentrale Komponente des Agenten
- MBeans werden beim MBean Server registriert
- Agent als Brücke zwischen MBeans und Management-Anwendung





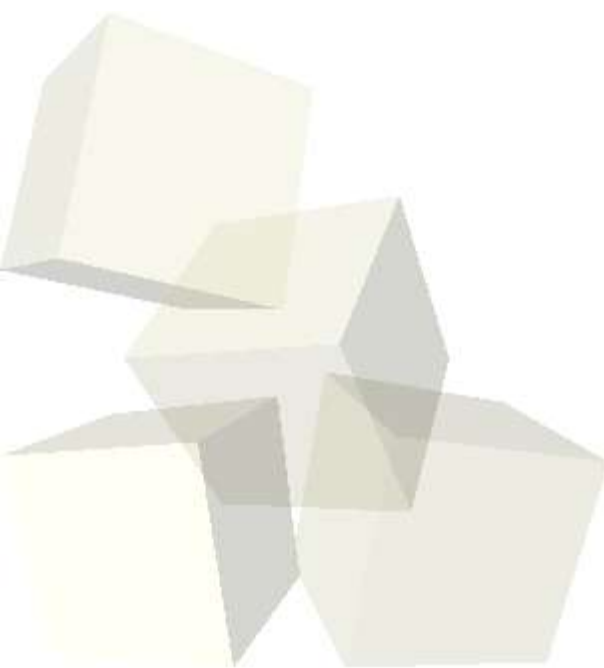
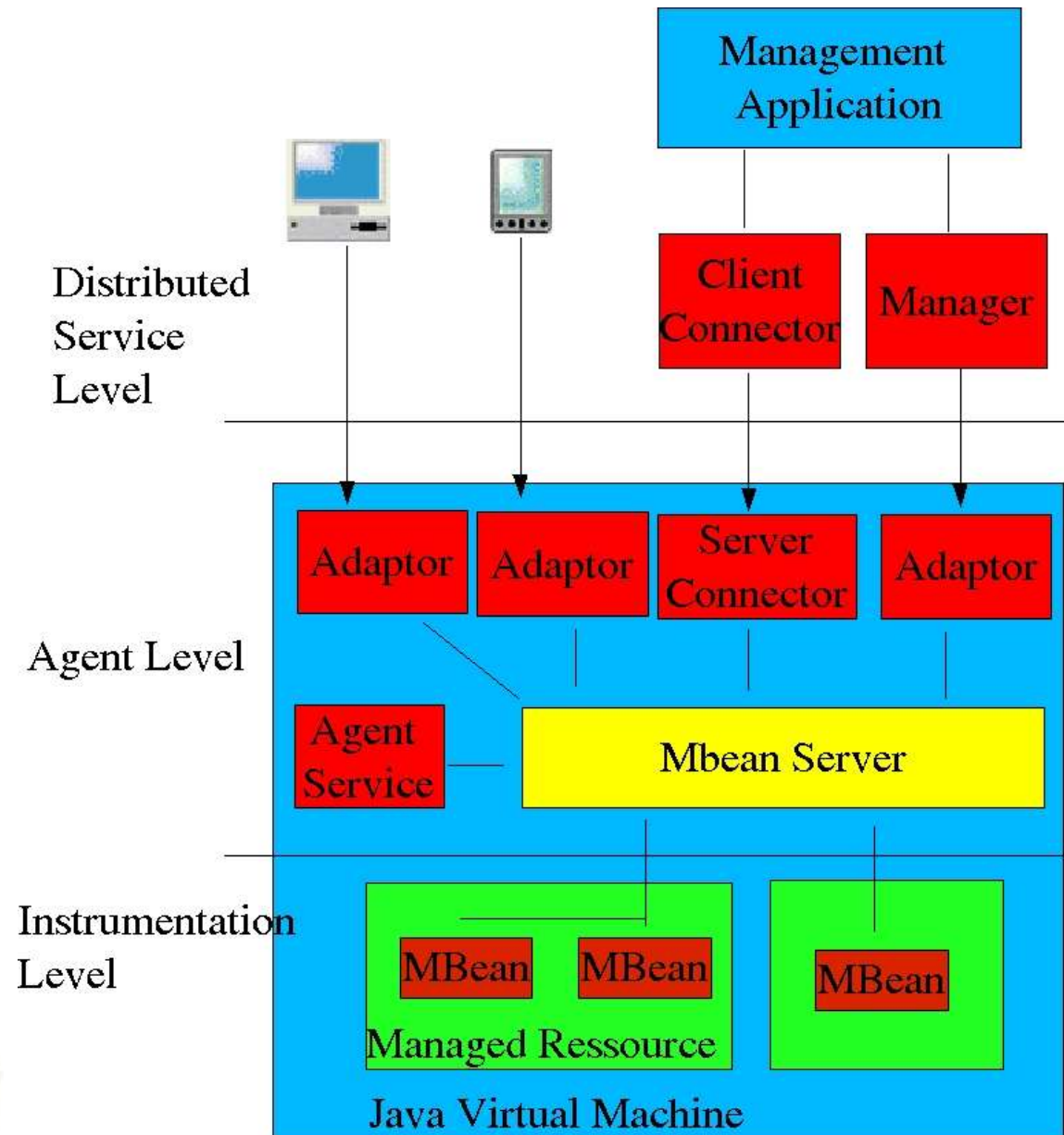
Distributed Services Level

- Schnittstelle für den Zugriff von Mgmt.-Applikationen auf den Agent-Level
- Zugriff bspw. per HTTP oder SNMP
- Abbildung der MBeans auf eine GUI





Architektur Übersicht





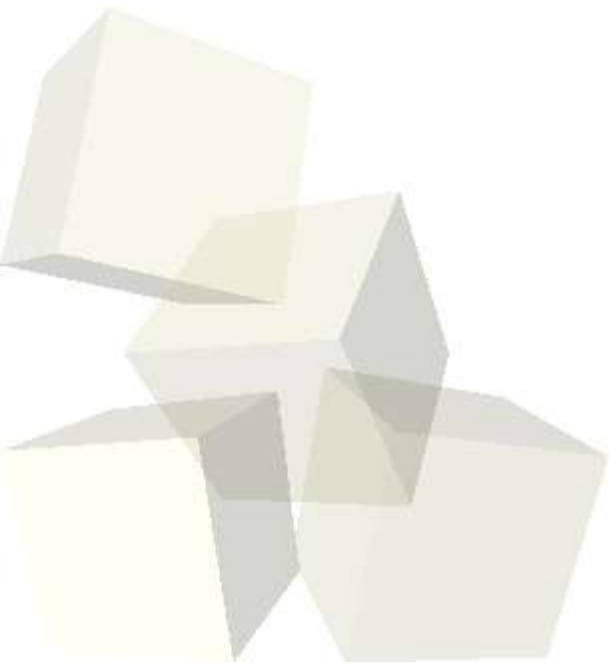
- 4 Arten von Managed Beans
 - ◆ Standard MBean
 - ◆ Dynamic MBean
 - ◆ Model MBean
 - ◆ Open MBean

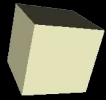




Managed Beans

- stellt zu überwachende Ressource dar
- jede MBean benötigt Mgmt.-Schnittstelle
- die vier MBeans unterscheiden sich nur in der Definition ihrer Schnittstelle





- 4 'goldene' Regeln zur Implementierung
 - ◆ Attributzugriff über getter- und setter-Methoden (Java Bean Spezifikation)
 - ◆ Codierung der Ressource entsprechend eines JMX MBean Type
 - ◆ mindestens ein öffentlicher Konstruktor
 - ◆ keine abstract Klassendefinition



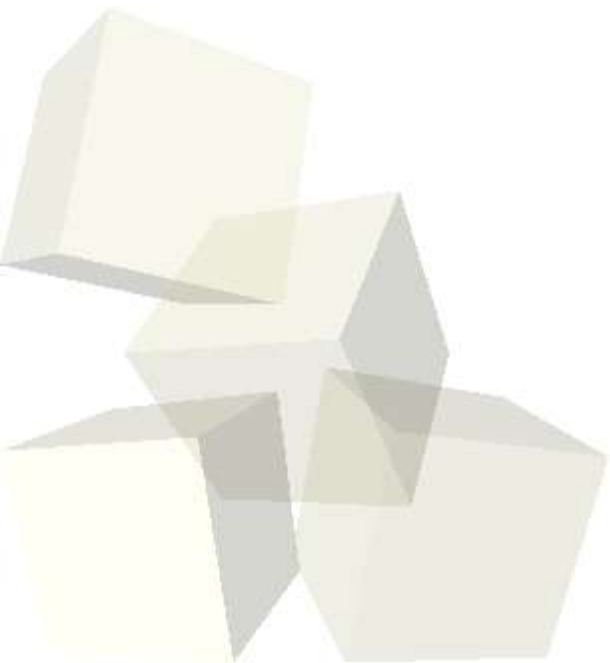


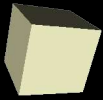
- Einfachste Form der MBean
- Statisches Mgmt.-Interface
- Attributzugriff über getter- und setter-Methoden
- Einzuhaltende Namenskonvention für die Introspection des MBean Servers
- Beispiel

StudentMBean
int getMatrikelnummer()
int getSemester()
setSemester(int sem)



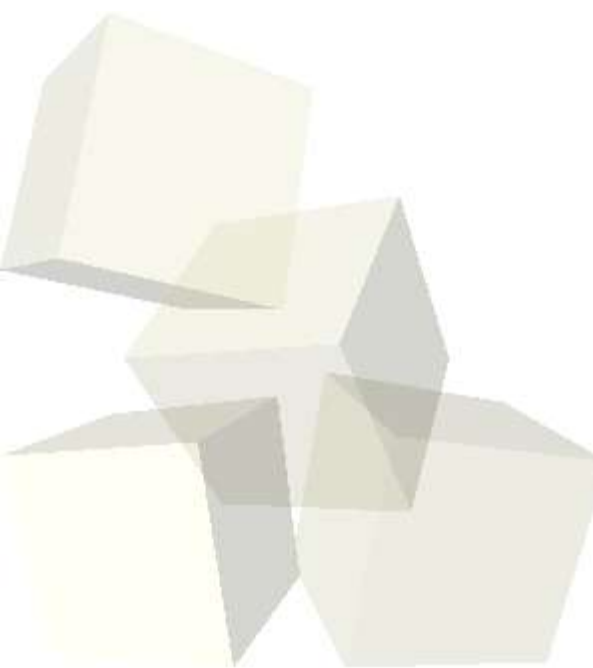
- Dynamischere Mgmt.-Schnittstelle als Standard MBean
- Implementiert javax.management.DynamicMBean-Interface
- Interface stellt Methoden zur Verfügung, die Mgmt.-Schnittstelle zu ermitteln





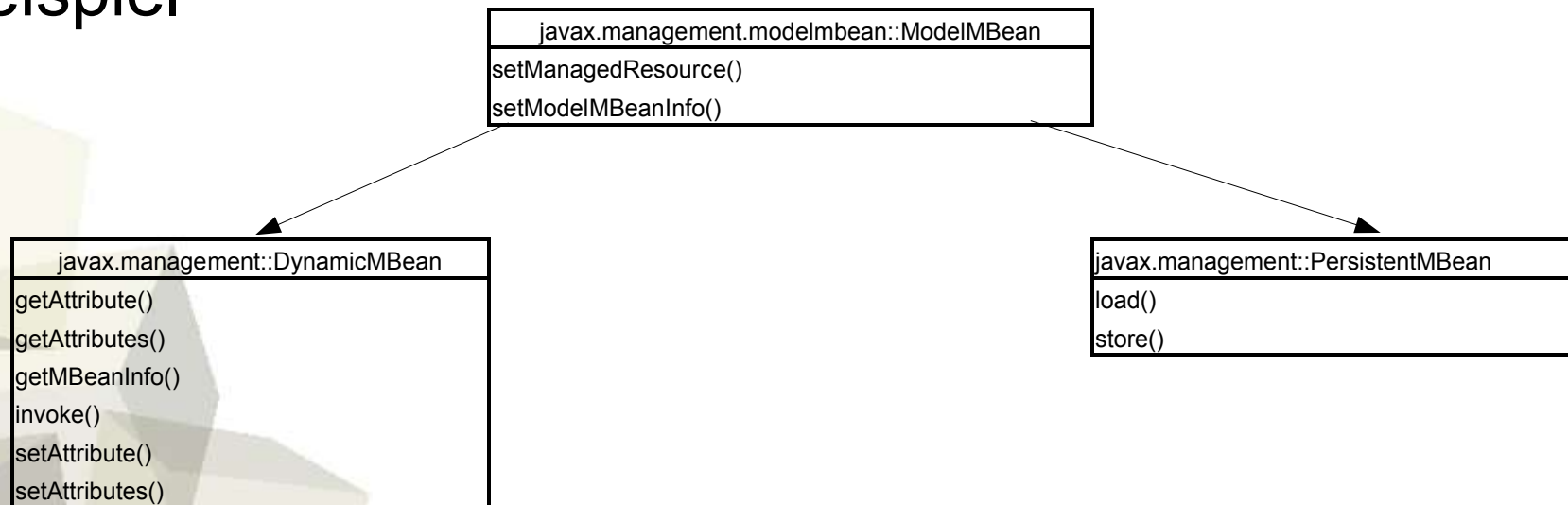
- Methode `getMBeanInfo()` liefert `MetaDataKlasse` zurück
- Beispiel

```
javax.management::DynamicMBean  
getAttribute()  
getAttributes()  
getMBeanInfo()  
invoke()  
setAttribute()  
setAttributes()
```





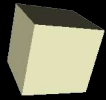
- Erweiterung der Dynamic MBean
- Management von so gut wie jeder Resource
- Implementiert PersistentMBeanInterface
- Beispiel





- Keine genaue Angabe zur Implementierung des Persistenzmechanismus
- Einführung des Descriptor-Konzepts
- Descriptoren enthalten Standard-Metadaten, eingeteilt in verschiedene Kategorien, bspw. zur automatischen Benachrichtigung bei Attributänderungen



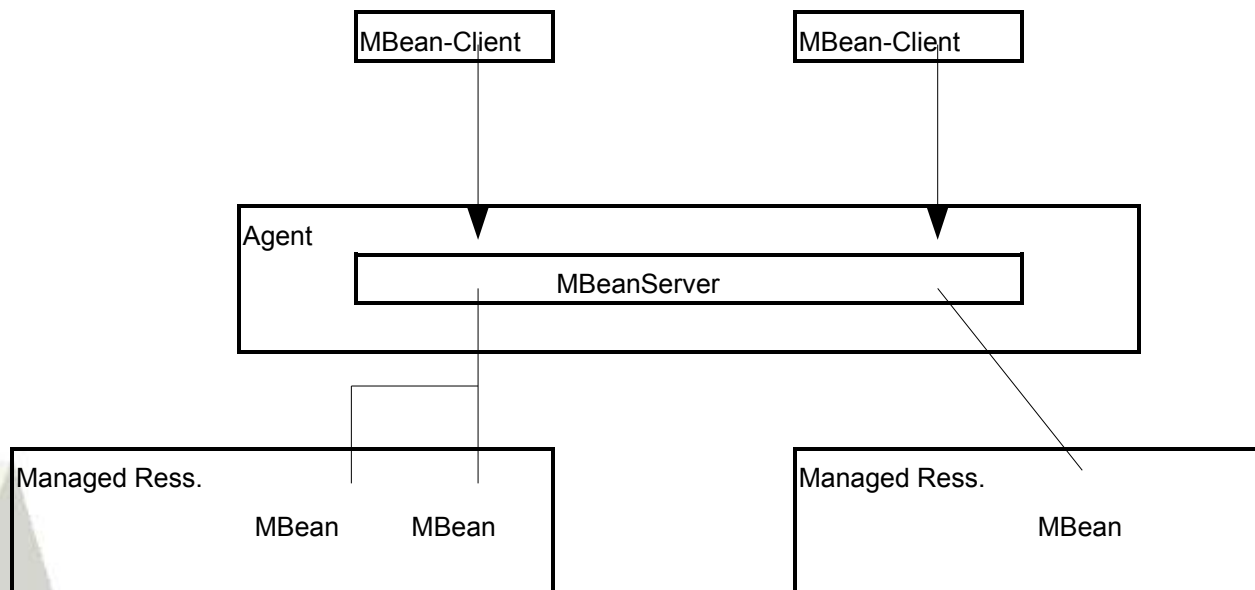


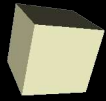
- Neue Objekte sollen zur Laufzeit erkannt und verwendet werden können
- Es existiert kein OpenMBean-Interface, das DynamicMBean-Interface wird implementiert
- Status der MBean muss durch BasicDataTypes festgelegt werden, bspw. Short, Integer, etc.





- Herzstück des JMX-Agenten
- Softwaremodul zur Registrierung von MBeans





- Einführung des ObjectName-Konzepts zur Identifizierung der MBean-Instanz
- ObjectName besteht aus einer Domain und einem Satz von Key-Properties
- z.B. MyPrinters:name=papyrus,type=laser,vendor=HP,location=foo-pool





- Ein Agent muss den Clients obligatorische Services zur Verfügung stellen

- 4 Services
 - ◆ Dynamic Classloading Service

 - ◆ Timer Service

 - ◆ Relation Service

 - ◆ Monitoring Service



Dynamic Classloading Service

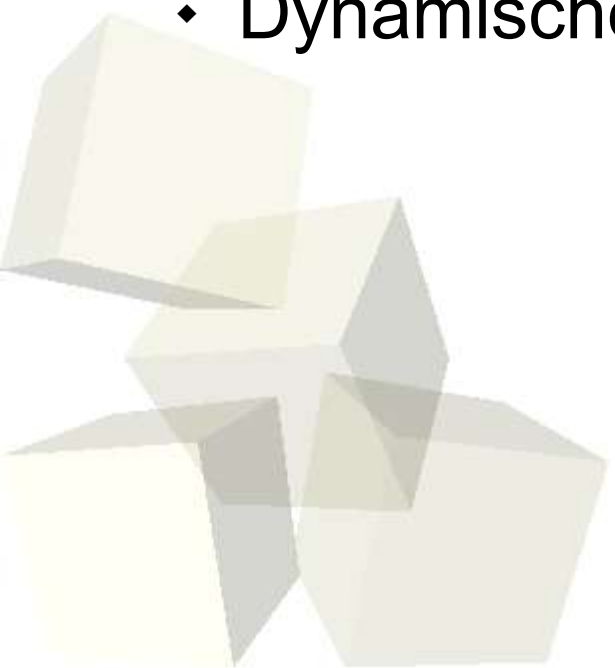
- Nutzt Management-Applet (sog. MLet)
- Aufruf von entfernten MBeans über URLs
- MLet Service verwaltet class- und jar-Cache
- Einlesen von xml-ähnlichen Dateien mit MLet-Tags
- MLet-Datei ist nicht zwingend notwendig





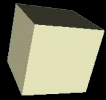
Dynamic Classloading Service

- Nutzung des Service ...
 - ◆ Remote Loading von MBeans
 - ◆ Zentralisierung der Konfiguration einer Application Ressource
 - ◆ Dynamisches Laden von MBeans zur Laufzeit

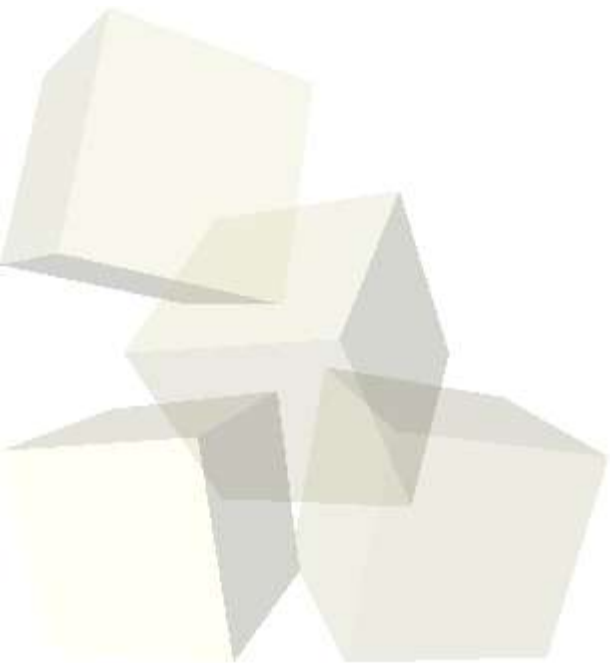




- Scheduling-Mechanismus für Ereignisbenachrichtigungen
- Verwaltung von verschiedenen Notifikationen
- Implementierung des Notification Broadcaster
- Registrierte Ereignisse werden an alle registrierten Listener versendet
- Der Timer Service arbeitet quasi als programmierbares Postfach

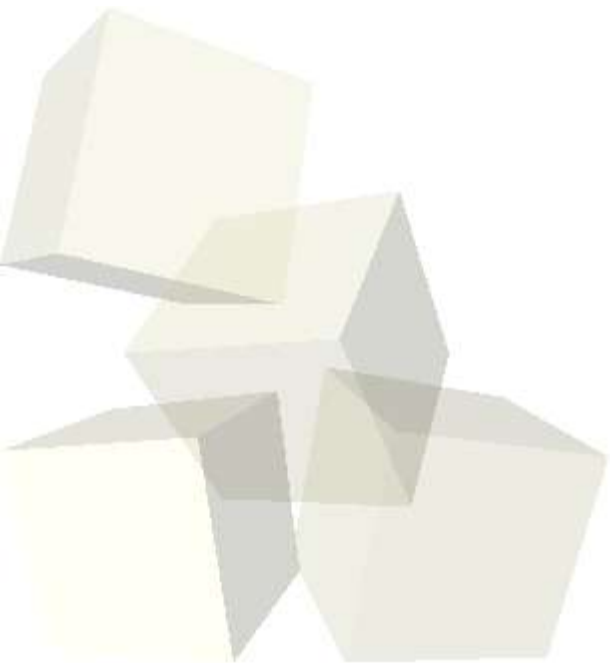


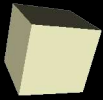
- Ermöglicht einfache Beziehungen zwischen MBeans
- Überprüfung dieser Beziehungen durch den Agenten
- Bei ungültigem Zustand der Beziehung wird die MBean entfernt





- Überwachung anderer MBeans
- Überwachen der Attributwerte einer MBean in zyklischen Abständen
- Benachrichtigung anderer Objekte, falls Grenzwerte erfüllt wurden (Auslösen von entsprechenden Events)





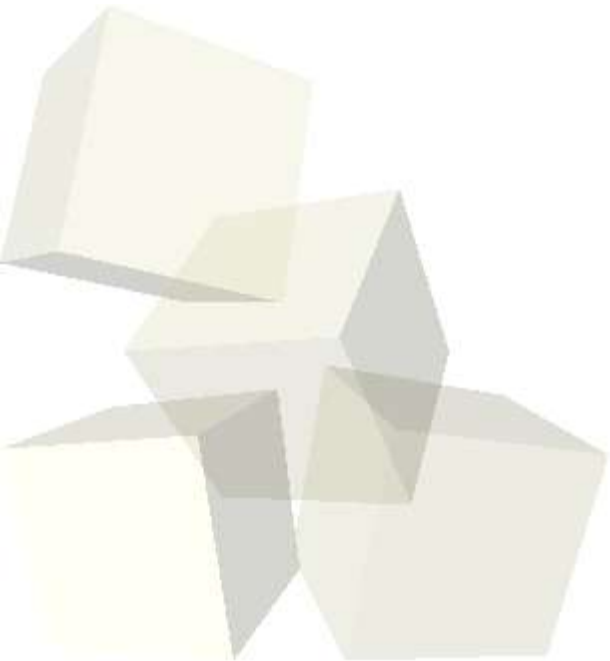
- Counter Monitor
 - ◆ ganzzahlige Attribute (Byte, Int, Short, Long)
 - ◆ auslösen von Treshold-Events

- Gauge Monitor
 - ◆ reelle Attribute (Float, Double)
 - ◆ auslösen von High- und Low-Events

- String Monitor
 - ◆ String-Attribute
 - ◆ löst Match- und Differs-Events aus



Ein Beispiel ...





- Schlanke Mgmt.-Schnittstelle
- Geringer Implementierungsaufwand auch bei bestehenden Systemen
- Durch Adaptern und Konnektoren lässt es sich in nahezu jede bestehende Lösung integrieren
- Könnte zum Defacto-Standard für Application Management werden





- <http://java.sun.com/products/JavaManagement/>
- <http://www.zdnet.de>
- <http://www.oio.de/public/java/jmx/jmx.htm>
- <http://www.jfs2003.de>
- http://www.st.informatik.tu-darmstadt.de/database/seminars/data/JMX_Seminar.pdf

Fragen ?

