JMX: Get the Most out of this Unsung Hero

Tom Lubinski

Chief Technical Officer
tlubinski@sl.com

SL Corporation
Corte Madera, CA

8 November, 2012

Agenda

Introduction to SL Corporation

JMX: A little background

JMX: How it can help

Q&A

Who is SL and what is RTView ?

Why should I listen to you about JMX ?

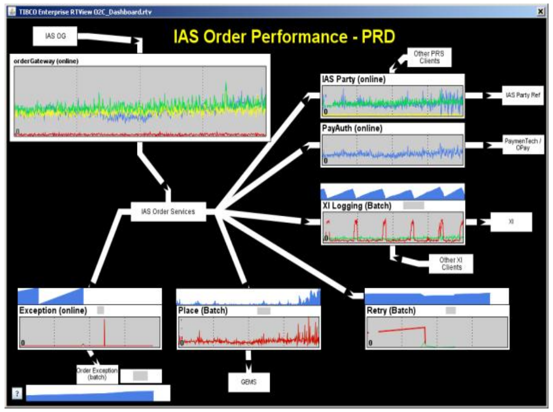
SL Corporation …

Extensive background in real-time application monitoring

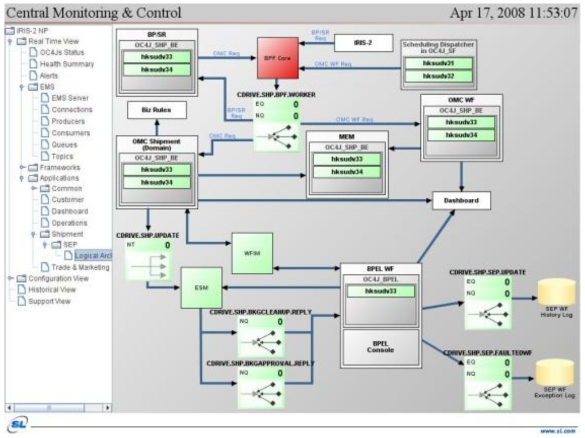
Large volumes of dynamic data

Visualization technologies

Specialists in Application / Middleware, esp. TIBCO



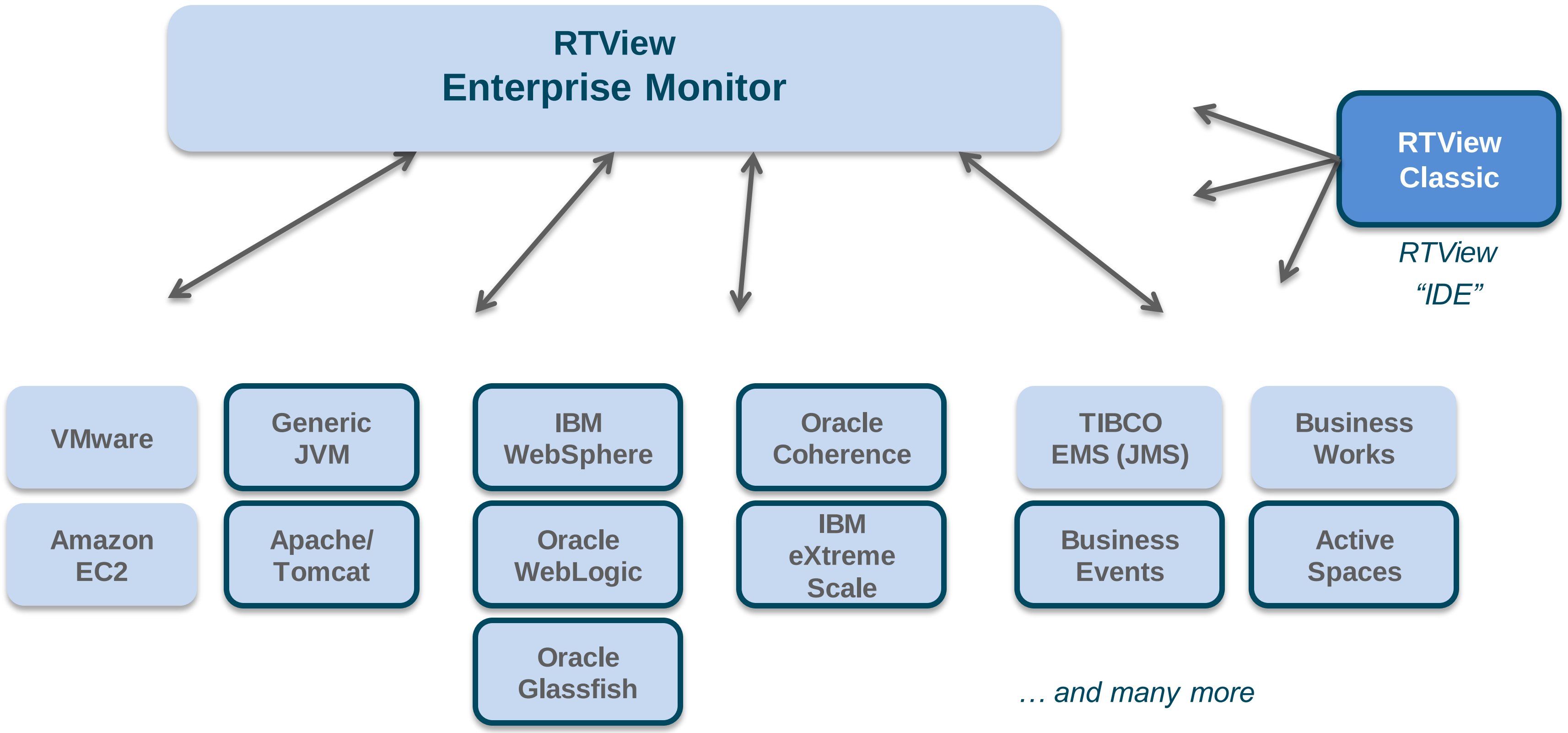
Critical Tax Season Applications at Intuit



OOCL World Wide Shipment Tracking

RTView Enterprise Monitor

Collects, Analyzes, and Visualizes data from different sources, many using JMX

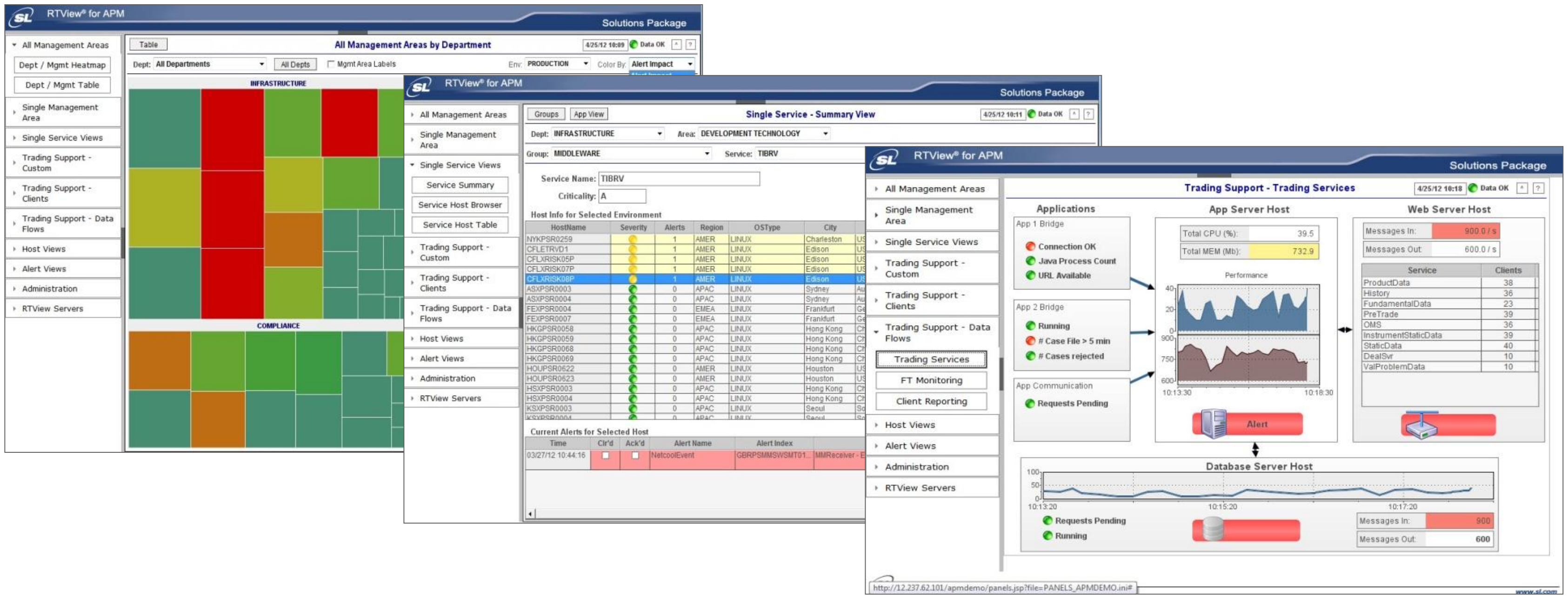


… and many more

RTView Enterprise Monitor

High-Level Application Summary Views

Drilldown to individual components to investigate problems



Service Name: TIBRV

Criticality: A

Host Info for Selected Environment

HostName	Severity	Alerts	Region	OSType	City
NYKPSR0259	●	1	AMER	LINUX	Charleston
CFLETRVD1	●	1	AMER	LINUX	Edison
CFLXRISK05P	●	1	AMER	LINUX	Edison
CFLXRISK07P	●	1	AMER	LINUX	Edison
CFLXRISK08P	●	1	AMER	LINUX	Edison
ASXPSR0003	●	0	APAC	LINUX	Sydney
ASXPSR0004	●	0	APAC	LINUX	Sydney
FEXPSR0004	●	0	EMEA	LINUX	Frankfurt
FEXPSR0007	●	0	EMEA	LINUX	Frankfurt
HKGPSR0058	●	0	APAC	LINUX	Hong Kong
HKGPSR0059	●	0	APAC	LINUX	Hong Kong
HKGPSR0068	●	0	APAC	LINUX	Hong Kong
HKGPSR0069	●	0	APAC	LINUX	Hong Kong
HOUPSR0622	●	0	AMER	LINUX	Houston
HOUPSR0623	●	0	AMER	LINUX	Houston
HSXPSR0003	●	0	APAC	LINUX	Hong Kong
HSXPSR0004	●	0	APAC	LINUX	Hong Kong
KSXPSR0003	●	0	APAC	LINUX	Seoul
KSXPSR0004	●	0	APAC	LINUX	Seoul

Current Alerts for Selected Host

Time	Clr'd	Ack'd	Alert Name	Alert Index	
03/27/12 10:44:16	☐	☐	NetcoolEvent	GBRPSMMSWSMT01...	MMReceiver - E

Total CPU (%):	39.5
Total MEM (Mb):	732.9

Messages In:	900.0 / s
Messages Out:	600.0 / s

Service	Clients
ProductData	38
History	36
FundamentalData	23
PreTrade	39
OMS	36
InstrumentStaticData	39
StaticData	40
DealSvr	10
ValProblemData	10

Messages In:	900
Messages Out:	600

Why should I care about JMX ?

(it's kind of boring, actually …)

Monitoring is critical for complex apps …

Do you want to re-invent the wheel ?

SL has found JMX to be excellent model …
as we hope you see in this presentation

Whence JMX …

Need for collecting monitoring information

Roots in agent technology, like TIBCO Hawk

JMX 1.1 then 1.2

External in Java 1.4, but automatic in 1.5+

Why JMX …

“Standards”

Standardize monitoring and management

Standard system-independent data types

Standard naming / access mechanism

Typical Monitoring Approaches …

None at all … very common

Output log files

Write to Database

Send JMS Message

Custom TCP or Web Service transport

JMX Makes it easy … and standard !

Confession …

Selfish Motives

Unclean Thoughts

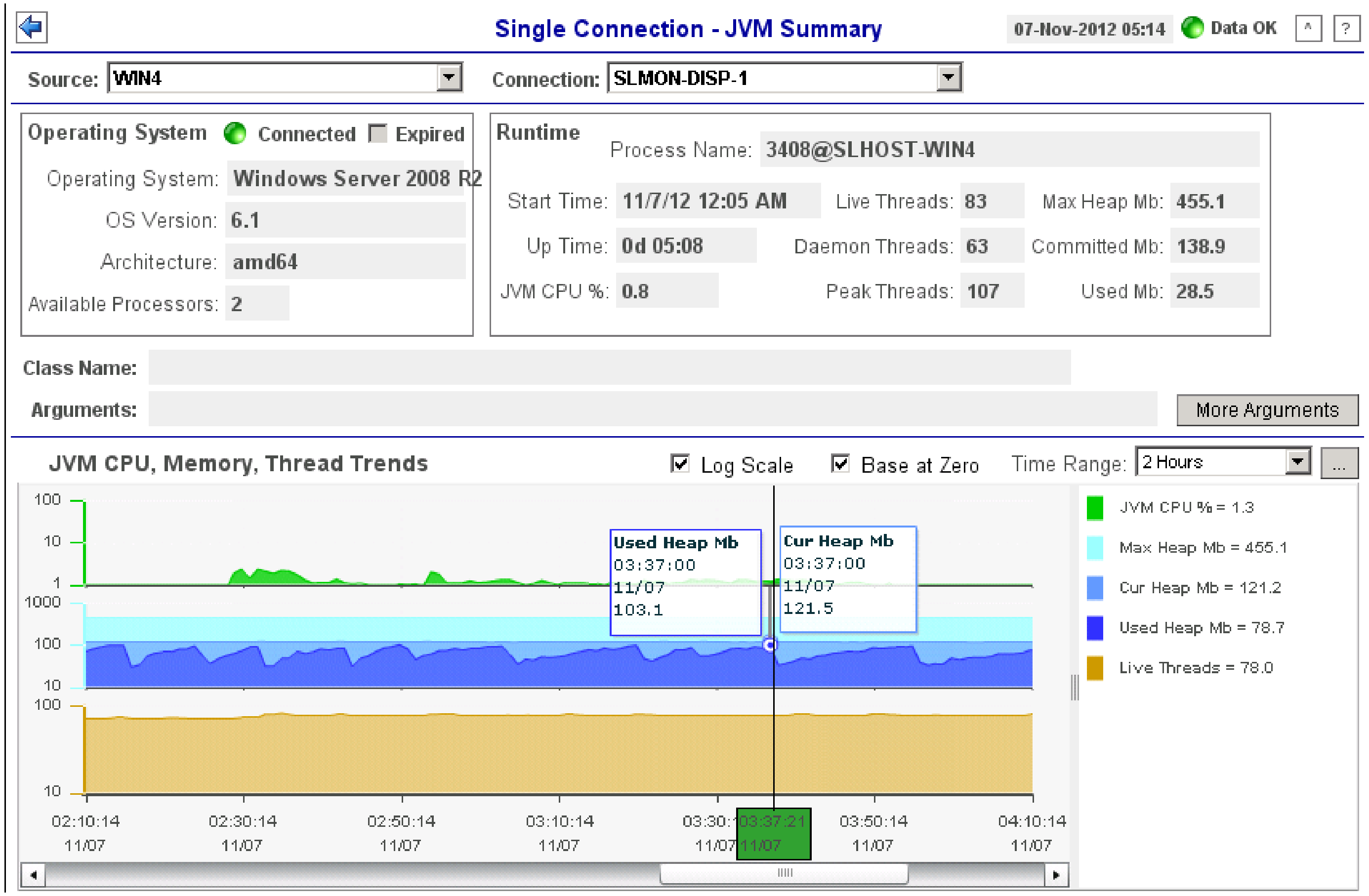
We lust for you to produce more monitoring data - easily !

We want you to use standards …
So RTView can collect data easily, analyze it, and visualize it

… and provide useful, actionable information for you

Examples of Standard JMX Data

Every JVM produces CPU, Memory, Thread, GC information



Single Connection - JVM Summary 07-Nov-2012 05:14 Data OK

Source: WIN4 Connection: SLMON-DISP-1

Operating System ● Connected ☐ Expired

Operating System:	Windows Server 2008 R2
OS Version:	6.1
Architecture:	amd64
Available Processors:	2

Runtime

Process Name: 3408@SLHOST-WIN4

Start Time:	11/7/12 12:05 AM	Live Threads:	83	Max Heap Mb:	455.1
Up Time:	0d 05:08	Daemon Threads:	63	Committed Mb:	138.9
JVM CPU %:	0.8	Peak Threads:	107	Used Mb:	28.5

Class Name:

Arguments:

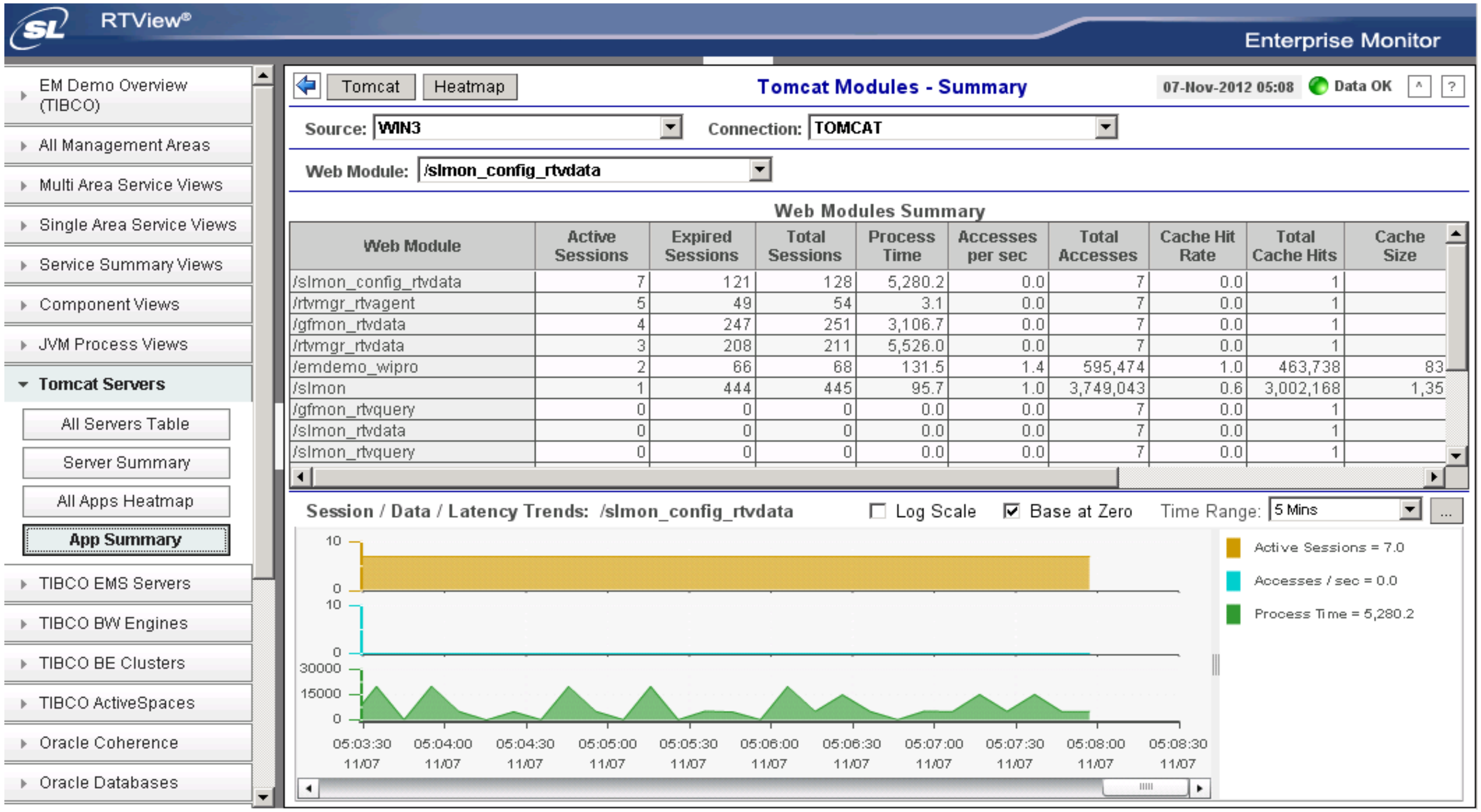
More Arguments

JVM CPU, Memory, Thread Trends ☑ Log Scale ☑ Base at Zero Time Range: 2 Hours

🟩	JVM CPU % = 1.3
🟦	Max Heap Mb = 455.1
🟦	Cur Heap Mb = 121.2
🟦	Used Heap Mb = 78.7
🟨	Live Threads = 78.0

Examples of Standard JMX Data

Tomcat produces Session data, Request Counts, and Response Time for entire server and every servlet



Tomcat Modules - Summary

07-Nov-2012 05:08

Source: WIN3

Connection: TOMCAT

Web Module: /slmon_config_rtvdata

Web Modules Summary

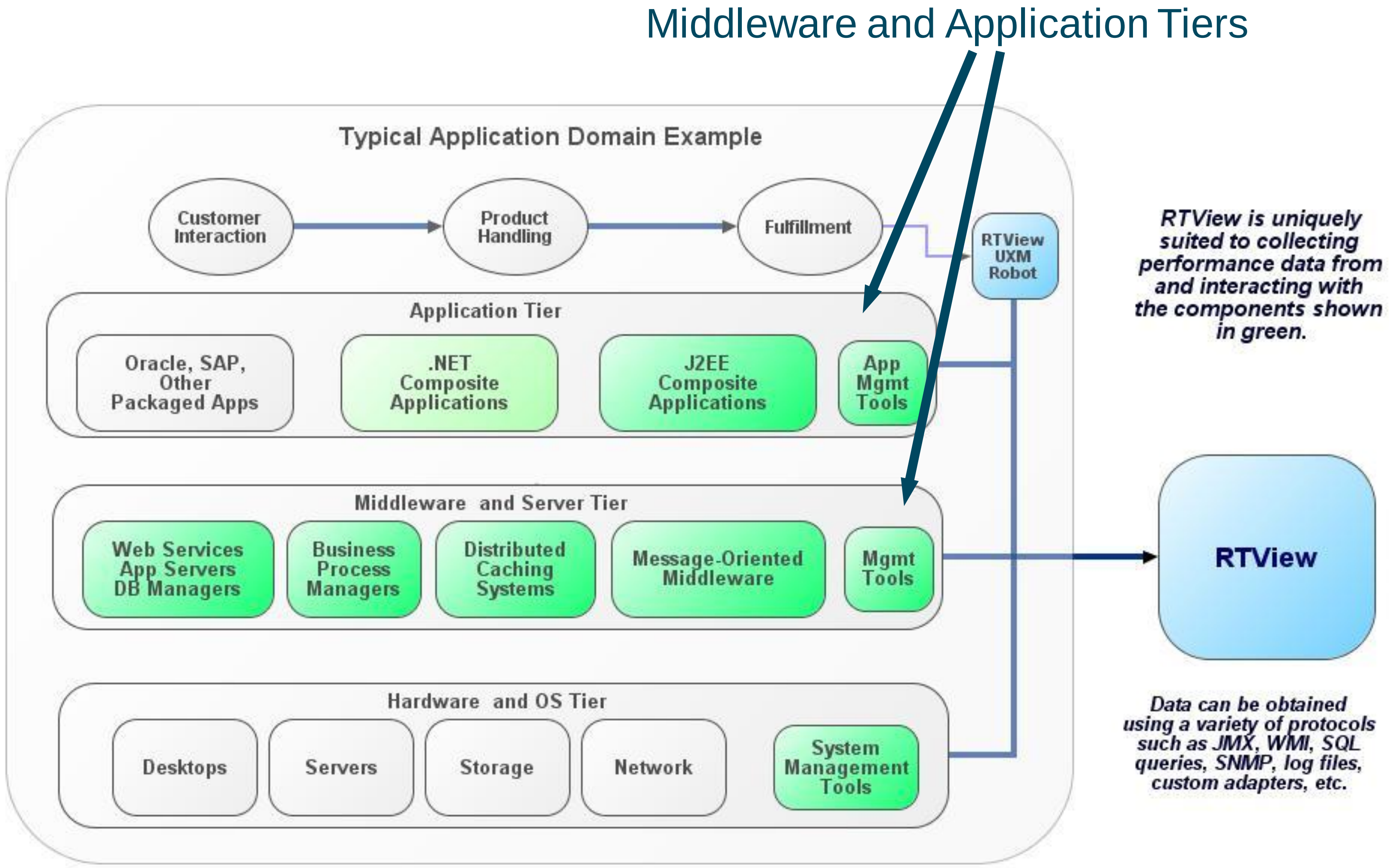
Web Module	Active Sessions	Expired Sessions	Total Sessions	Process Time	Accesses per sec	Total Accesses	Cache Hit Rate	Total Cache Hits	Cache Size
/slmon_config_rtvdata	7	121	128	5,280.2	0.0	7	0.0	1	
/rtvmgr_rtvagent	5	49	54	3.1	0.0	7	0.0	1	
/gfmon_rtvdata	4	247	251	3,106.7	0.0	7	0.0	1	
/rtvmgr_rtvdata	3	208	211	5,526.0	0.0	7	0.0	1	
/emdemo_wipro	2	66	68	131.5	1.4	595,474	1.0	463,738	83
/slmon	1	444	445	95.7	1.0	3,749,043	0.6	3,002,168	1,35
/gfmon_rtvquery	0	0	0	0.0	0.0	7	0.0	1	
/slmon_rtvdata	0	0	0	0.0	0.0	7	0.0	1	
/slmon_rtvquery	0	0	0	0.0	0.0	7	0.0	1	

Session / Data / Latency Trends: /slmon_config_rtvdata

☐ Log Scale ☑ Base at Zero Time Range: 5 Mins

🟧 Active Sessions = 7.0
🟦 Accesses / sec = 0.0
🟩 Process Time = 5,280.2

Where JMX Data can be Collected


Middleware and Application Tiers
Typical Application Domain Example
Customer Interaction
Product Handling
Fulfillment
RTView UXM Robot
Application Tier
Oracle, SAP, Other Packaged Apps
.NET Composite Applications
J2EE Composite Applications
App Mgmt Tools
Middleware and Server Tier
Web Services App Servers DB Managers
Business Process Managers
Distributed Caching Systems
Message-Oriented Middleware
Mgmt Tools
RTView
Hardware and OS Tier
Desktops
Servers
Storage
Network
System Management Tools

RTView is uniquely suited to collecting performance data from and interacting with the components shown in green.

Data can be obtained using a variety of protocols such as JMX, WMI, SQL queries, SNMP, log files, custom adapters, etc.

How can I get my Apps to produce data like Tomcat or other middleware ?

Learn to use JMX …

… The ***right*** way !

What is the so-called ***right*** way to use JMX ?

1) Abstract Monitoring Model
2) Simple Data Model
3) Pluggable Transport Mechanism

1) Abstract Monitoring Model

Monitoring is observing !

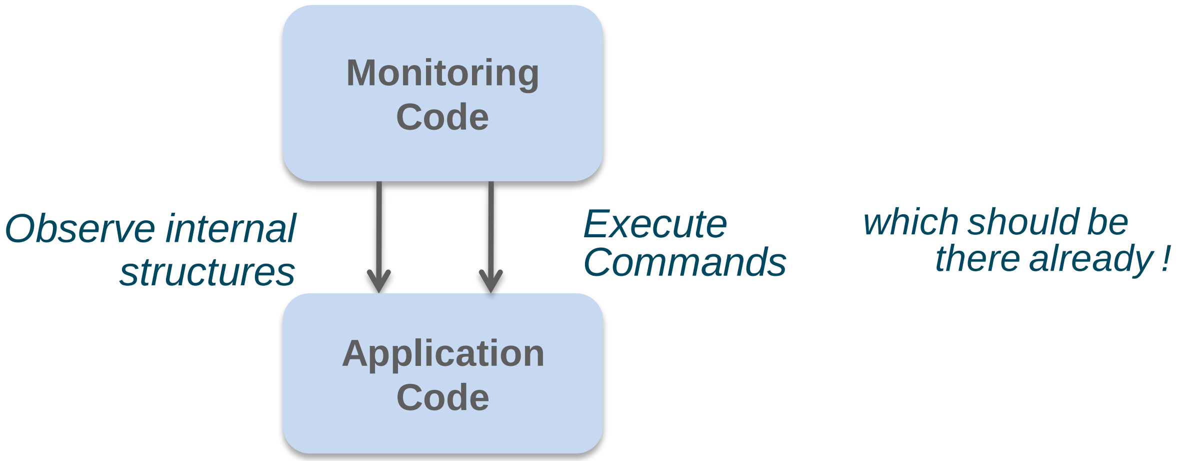
Management is commanding

Isolate monitoring / management code from application code

1) Abstract Monitoring Model

Communication should be one-way !

Application shouldn't know it is being monitored …



1) Abstract Monitoring Model

Only one requirement to use JMX:

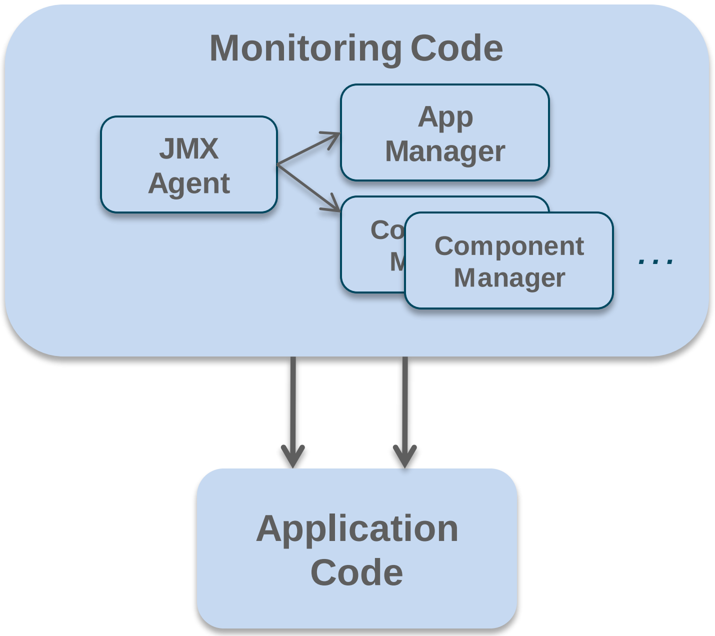
Application must indicate it is observable and set up the observer

Modularity is important ! Organize well from start !

More data structures = more complexity

1) Abstract Monitoring Model

Application = Global App Data + Component Data



Modular organization

1) Abstract Monitoring Model Application Class

```
public class MyApplication {

        // Global Application data …

        // Component data …

public MyApplication ()
{
        // Create a JMX Agent to manage this App
        SampleJmxAgent agent = new SampleJmxAgent(this);
}

{
```

1) Abstract Monitoring Model Sample JMX Agent Class

```
import javax.management.*;
import java.lang.management.*;

public class SampleJmxAgent {

public SampleJmxAgent (MyApplication myApp)
{
        // Get the platform MBeanServer
        MBeanServer mBeanServer = ManagementFactory.getPlatformMBeanServer();

        // Create App Manager instance
        SampleAppManager managerBean = new SampleAppManager(mBeanServer, myApp);

        // Multiple Component Manager instances ..
}
```

1) Abstract Monitoring Model Sample App Manager Class

```
public class SampleAppManager implements SampleAppManagerMBean
{
      private MyApplication myApp;

public SampleAppManager (MBeanServer mBeanServer, MyApplication myApp)
{
      // Save reference to App
      this.myApp = myApp;

      // Uniquely identify this MBean instance and register with the MBeanServer
      try {
        ObjectName managerName = new ObjectName("MyApplication:name=AppManager");
        mBeanServer.registerMBean(this, managerName);

      } catch(Exception e) {
        e.printStackTrace();
      }
}
        // data access method definitions …
}
```

1) Abstract Monitoring Model
Sample App Manager MBean Class

```
public interface SampleAppManagerMBean
{
    // data access method declarations …
}
```

2) Simple Data Model

Make monitoring data easy to consume !

Avoid complex data structures that must be parsed

Make data “self-contained” – include indexes

2) Simple Data Model

7 Basic Data Types:

- int, long
- double, float
- boolean
- String
- Date

2) Simple Data Model Application Class

```
public class MyApplication {

     // Global Application variables
    int intVar = 123;
    long longVar = 12345678900L;
    float floatVar = 12.34f;
    double doubleVar = 567.899999;
    boolean booleanVar = true;
    String stringVar ="TestString";
    java.util.Date dateVar = new java.util.Date();

public MyApplication ()
{
    // Create a JMX Agent to manage this App
    agent = new SampleJmxAgent(this);
}

{
```

2) Simple Data Model
Sample App Manager Class

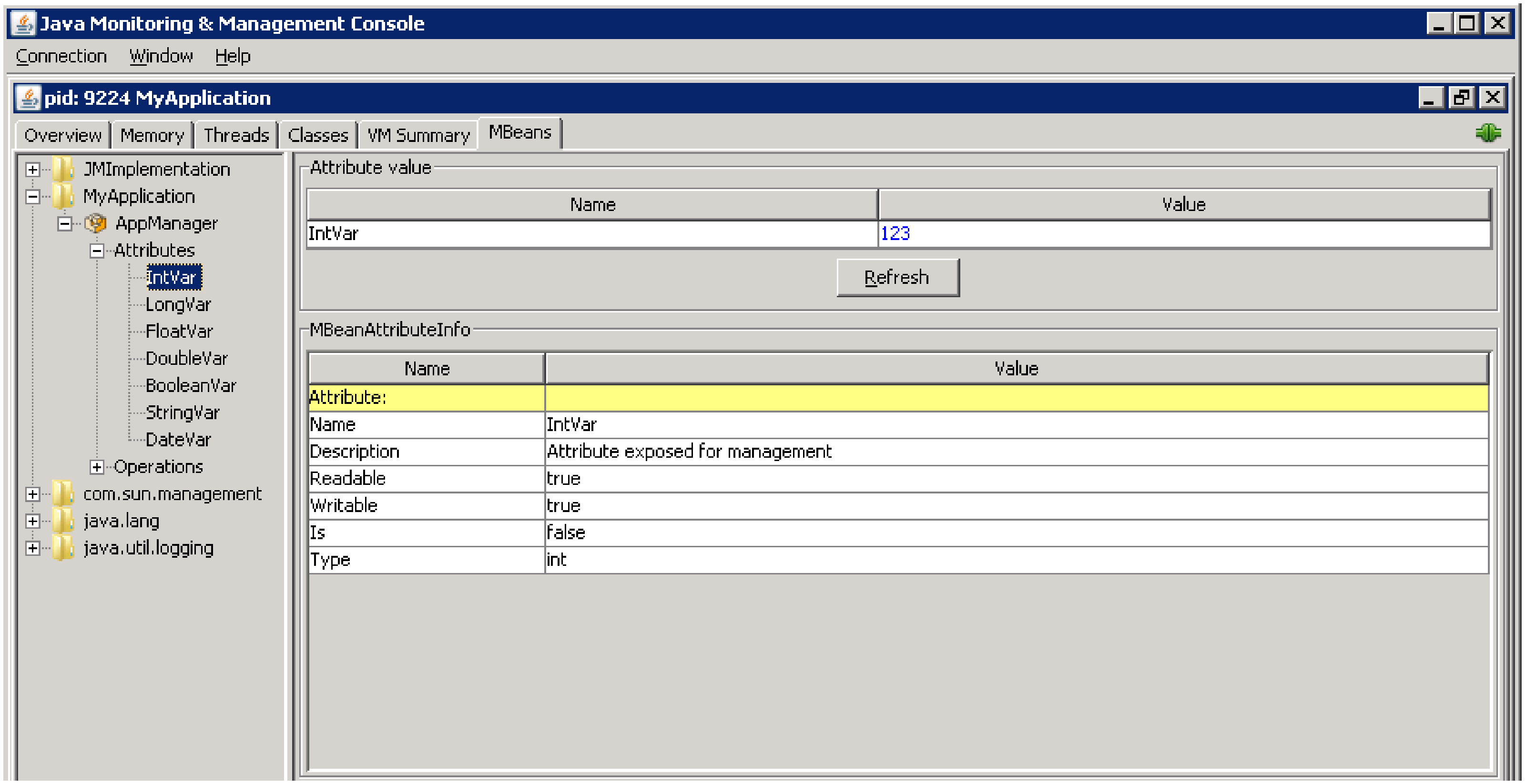
```
public class SampleAppManager implements SampleAppManagerMBean
{
        // data access method definitions …

        public int getIntVar () { return myApp.intVar; }
        public void setIntVar (int i) { }

        public long getLongVar () { return myApp.longVar; }
        public void setLongVar (long l) { }

        public float getFloatVar () { return myApp.floatVar; }
        public void setFloatVar (float f) { }

        public double getDoubleVar () { return myApp.doubleVar; }
        public void setDoubleVar (double d) { }

        public boolean getBooleanVar () { return myApp.booleanVar; }
        public void setBooleanVar (boolean b) { }

        public String getStringVar () { return myApp.stringVar; }
        public void setStringVar (String s) { }

        public java.util.Date getDateVar () { return myApp.dateVar; }
        public void setDateVar (java.util.Date date) { }
}
```

2) Simple Data Model
Sample App Manager MBean Class

```
public interface SampleAppManagerMBean
{
	// data access method declarations …
	public int getIntVar ();
	public void setIntVar (int i);

	public long getLongVar ();
	public void setLongVar (long l);

	public float getFloatVar ();
	public void setFloatVar (float f);

	public double getDoubleVar ();
	public void setDoubleVar (double d);

	public boolean getBooleanVar ();
	public void setBooleanVar (boolean b);

	public String getStringVar ();
	public void setStringVar (String s);

	public java.util.Date getDateVar ();
	public void setDateVar (java.util.Date date);
}
```

2) Simple Data Model

Viewed in jconsole …



Attribute value

Name	Value
IntVar	123

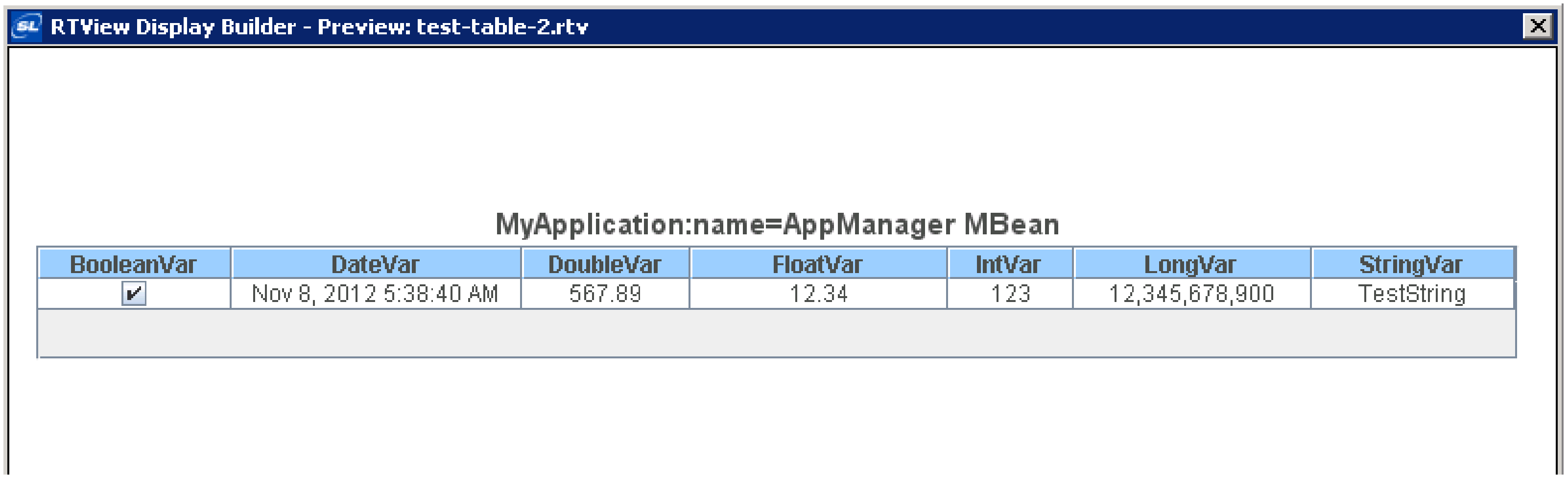
MBeanAttributeInfo

Name	Value
Attribute:	
Name	IntVar
Description	Attribute exposed for management
Readable	true
Writable	true
Is	false
Type	int

2) Simple Data Model

Viewed in RTView Builder …

One tabular row per MBean, permitting aggregation across multiple instances



RTView Display Builder - Preview: test-table-2.rtv

MyApplication:name=AppManager MBean

BooleanVar	DateVar	DoubleVar	FloatVar	IntVar	LongVar	StringVar
✓	Nov 8, 2012 5:38:40 AM	567.89	12.34	123	12,345,678,900	TestString

2) Simple Data Model

Other Useful Data Types:

Should be used with care (supported by RTView, but not all JMX tools)

- Array
- CompositeData
- TabularData

2) Simple Data Model Array

RTView and jconsole can view all array elements at once …

In MyApplication:

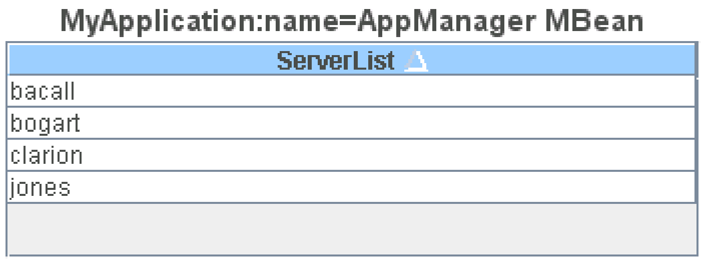
```
String[] serverList = { "bogart", "bacall", "jones", "clarion" };
```

In AppManager:

```
public String[] getServerList () { return myApp.serverList; }
```

In AppManagerMBean:

```
public String[] getServerList () ;
```



MyApplication:name=AppManager MBean

ServerList △
bacall
bogart
clarion
jones

2) Simple Data Model CompositeData

Single row data structure, consisting of multiple typed fields (items)

```
CompositeType ctype = new CompositeType(typeName, indexNames, itemNames, itemNames, itemTypes);
```

Not recommended for general use …

Many developers use Composite, but difficult to use by clients

RTView can see them easily, but jconsole can only view one element at a time
(as well as most other JMX tools)

2) Simple Data Model TabularData

Tabular data structure, consisting of multiple Composite rows

```
CompositeType ctype = new CompositeType("typeName", "description", itemNames, itemDescriptions, itemTypes);
TabularType ttype = new TabularType("typeName", "description", ctype, indexNames)
```

Recommended for use when performance is an issue …

More work to use by clients, but is most efficient for large tables

Alternative to multiple instances of single MBean

RTView can see them easily, but jconsole can only view one element at a time (as well as most other JMX tools)

2) Simple Data Model TabularData

Sample use case:

Oracle Coherence = distributed cache system

e.g. 100 Nodes x 50 caches distributed = 5000 MBeans

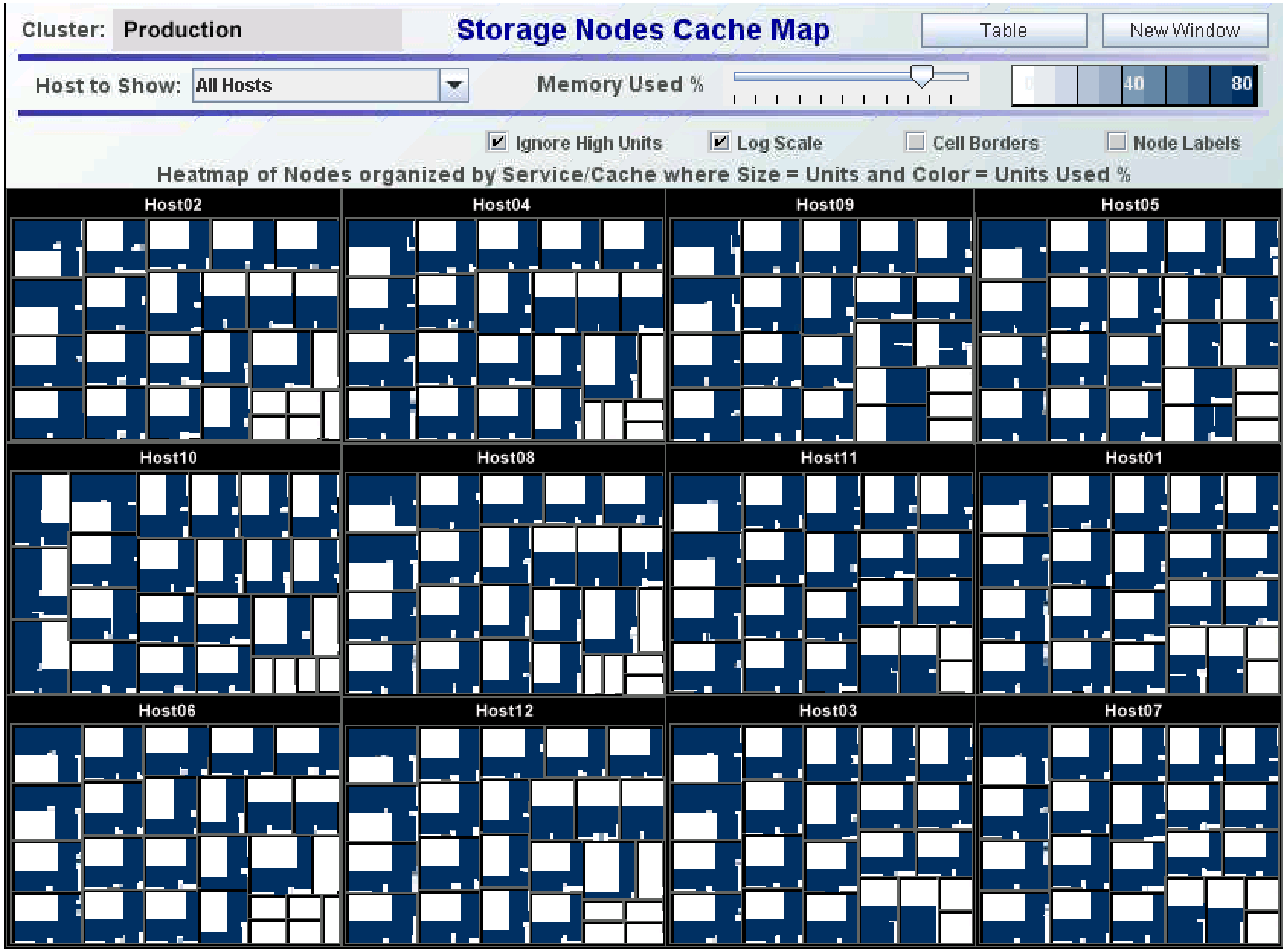
SL provided optimized TabularData version of same data:

> 100 TabularData MBeans with 50 rows each

Used 3 X less network bandwidth to transfer and 10 X speed improvement

2) Simple Data Model TabularData

RTView display showing data from thousands of Mbeans in heatmap



3) Pluggable Transport Mechanism

Separate monitoring data structures from transport code

In-memory monitoring data stored in uniform fashion = input to transport mechanism

3) Pluggable Transport Mechanism

The JMX Data Model, e.g. SimpleType and TabularType = important

Transport can be anything

3) Pluggable Transport Mechanism

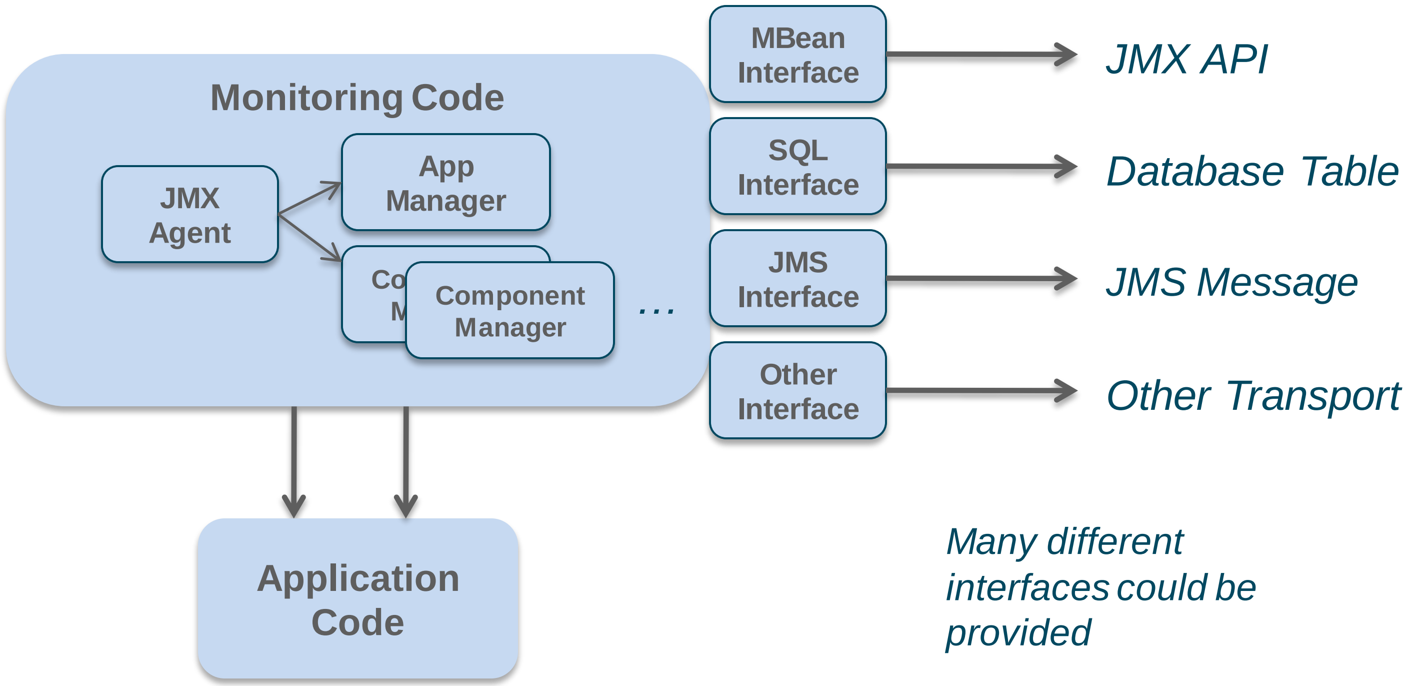
The JMX Data Model, e.g. SimpleType and
  TabularType = important

Transport can be anything:

  Log File
  Database Table
  JMS Message
  …

3) Pluggable Transport Mechanism Architecture View

Provide Transport Plugins / Interfaces



Many different interfaces could be provided

4) Some things NOT to do …

Forget to include index columns in your data

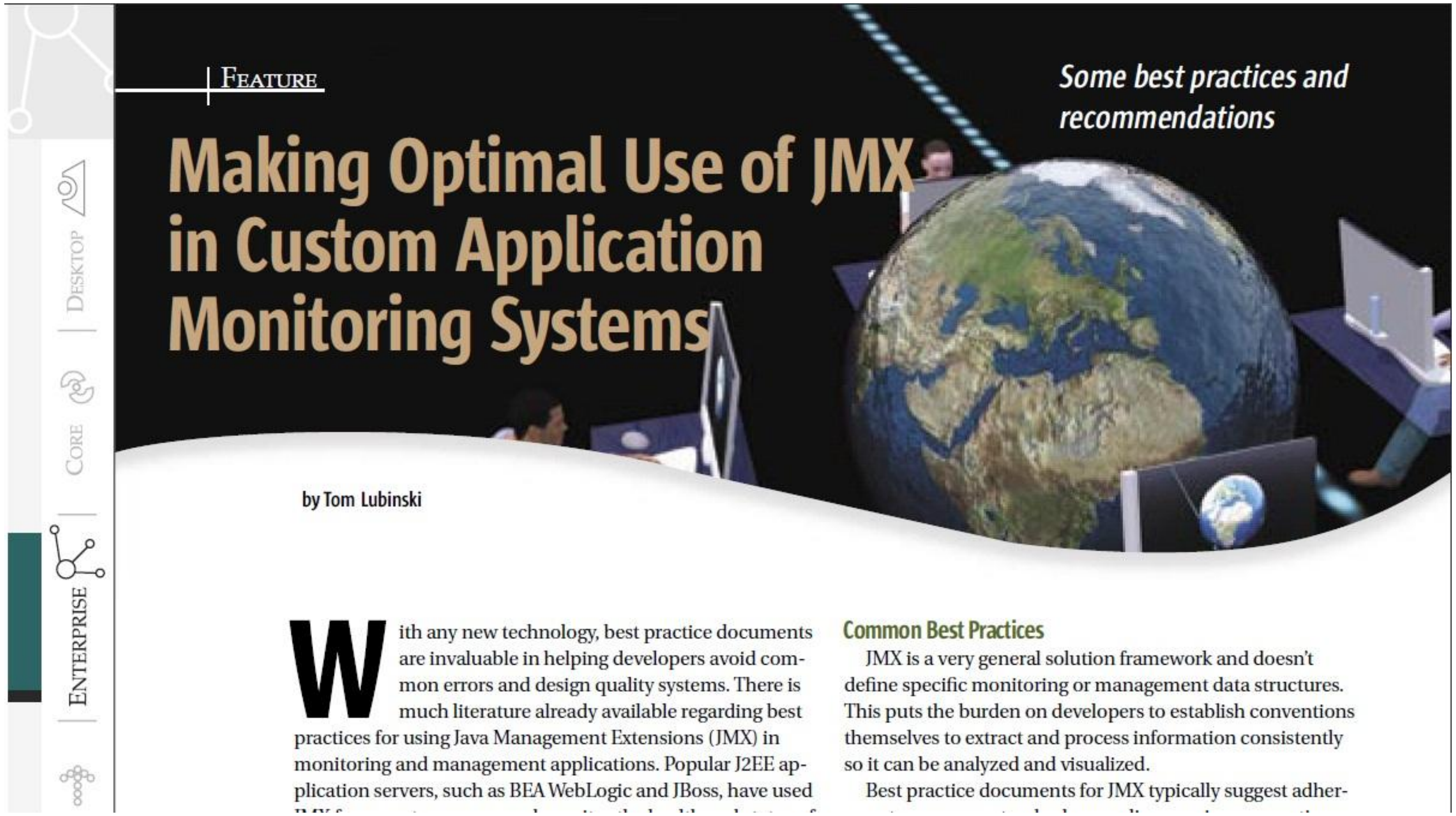
Encode monitoring data in XML string fields

Use overly complex keys

Inconsistent key, index column mappings

4) Some things NOT to do …

Additional Info in JDJ Technical Paper:


DESKTOP
CORE
ENTERPRISE

FEATURE

Some best practices and recommendations

Making Optimal Use of JMX in Custom Application Monitoring Systems

by Tom Lubinski

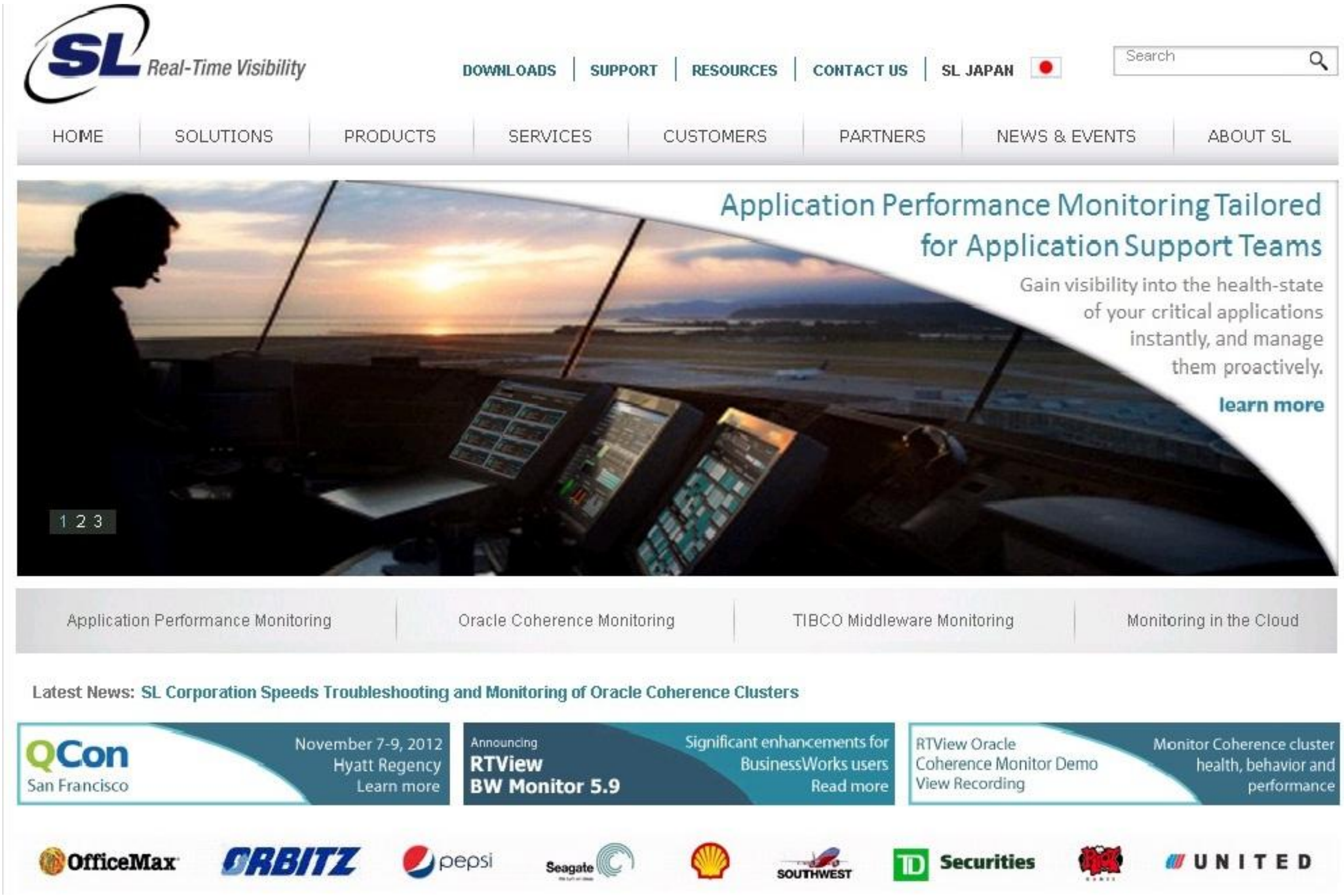
With any new technology, best practice documents are invaluable in helping developers avoid common errors and design quality systems. There is much literature already available regarding best practices for using Java Management Extensions (JMX) in monitoring and management applications. Popular J2EE application servers, such as BEA WebLogic and JBoss, have used

Common Best Practices

JMX is a very general solution framework and doesn't define specific monitoring or management data structures. This puts the burden on developers to establish conventions themselves to extract and process information consistently so it can be analyzed and visualized.

Best practice documents for JMX typically suggest adher-

Resources: www.sl.com



Real-Time Visibility

DOWNLOADS | SUPPORT | RESOURCES | CONTACT US | SL JAPAN

Search

HOME SOLUTIONS PRODUCTS SERVICES CUSTOMERS PARTNERS NEWS & EVENTS ABOUT SL

Application Performance Monitoring Tailored for Application Support Teams

Gain visibility into the health-state of your critical applications instantly, and manage them proactively.

learn more

1 2 3

Application Performance Monitoring | Oracle Coherence Monitoring | TIBCO Middleware Monitoring | Monitoring in the Cloud

Latest News: SL Corporation Speeds Troubleshooting and Monitoring of Oracle Coherence Clusters

QCon San Francisco — November 7-9, 2012 Hyatt Regency Learn more

Announcing RTView BW Monitor 5.9 — Significant enhancements for BusinessWorks users Read more

RTView Oracle Coherence Monitor Demo View Recording — Monitor Coherence cluster health, behavior and performance

OfficeMax ORBITZ pepsi Seagate SOUTHWEST TD Securities UNITED

Q & A