

# TWaver Java 开发培训

v 3.0

Serva Software LLC

# TWaver Java基础介绍

TWaver™

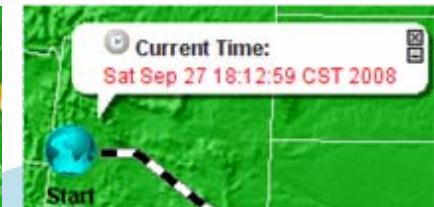
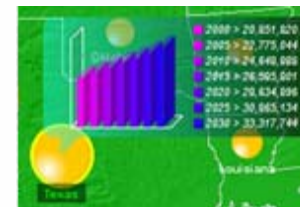
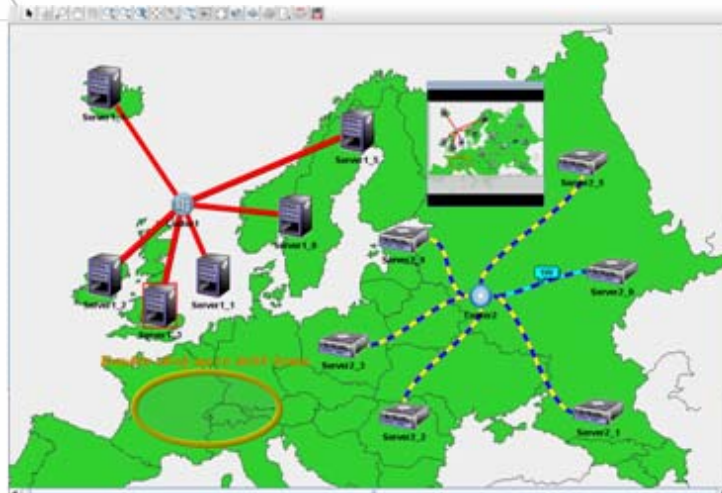
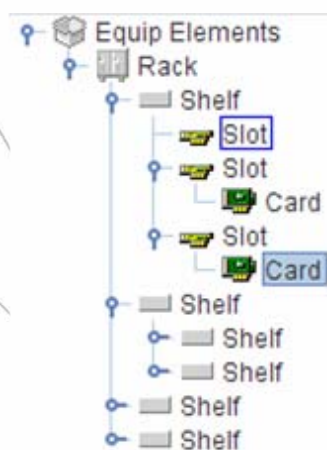
- TWaver是什么？
- Java Swing基础
- TWaver基础
  - Hello TWaver !
  - MVC设计模式(数据驱动事件机制组件联动)
  - 数据元素 / 数据容器 / 组件介绍

# TWaver 是什么？

- 高效轻量的图形组件库
- 提供多种平台解决方案
- TWaver产品包内容
- TWaver授权许可

# TWaver是图形组件库

- TWaver关注数据的图形化展示
- TWaver面向开发人员，需要二次开发
- TWaver提供文档，许可，培训和技术支持

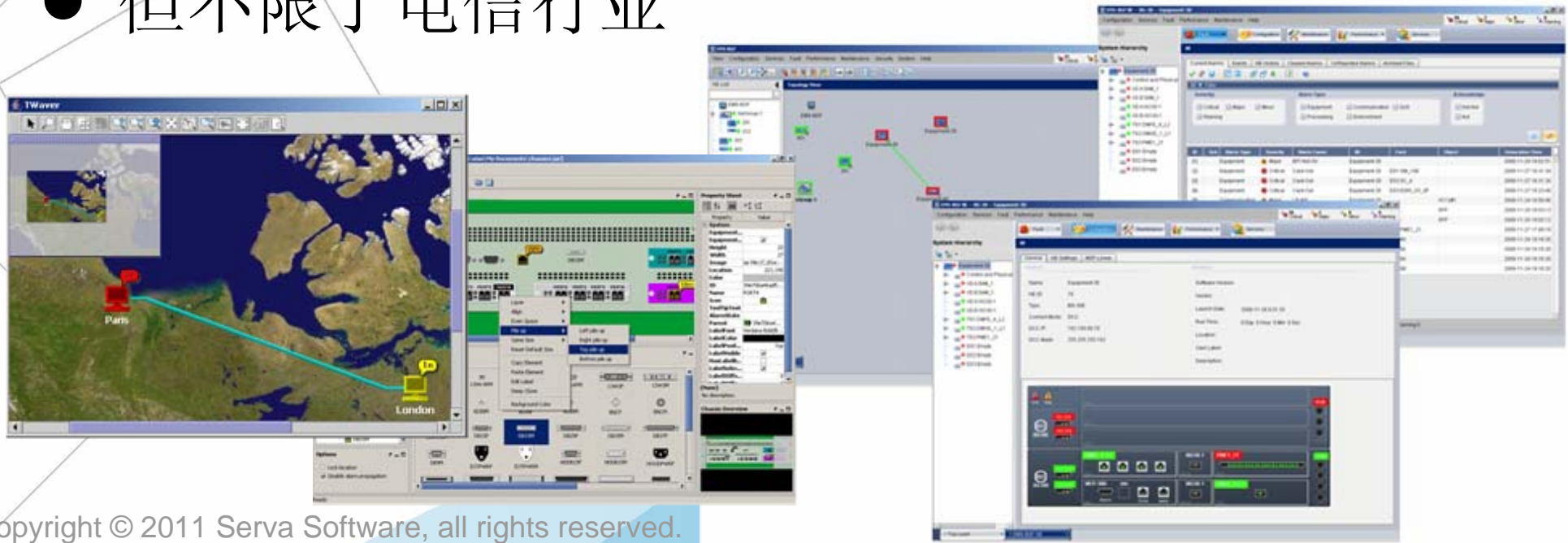


1/3Page 1-20/58Row

AlarmID	AlarmSeverity	Acked	ElementID	P
57	Major		Computer9	Loss of multi frame
56	Minor		Computer2	Invalid MSU receive
55	Critical		Computer5	Link failure
54	Warning			High wind
53	Indeterminate			Data set or modem
52	Critical			Battery breakdown

# TWaver关注于电信网管

- 电信相关的业务模型（设备面板，告警传递.....）
- 积累了大量电信行业的应用案例
- 但不限于电信行业

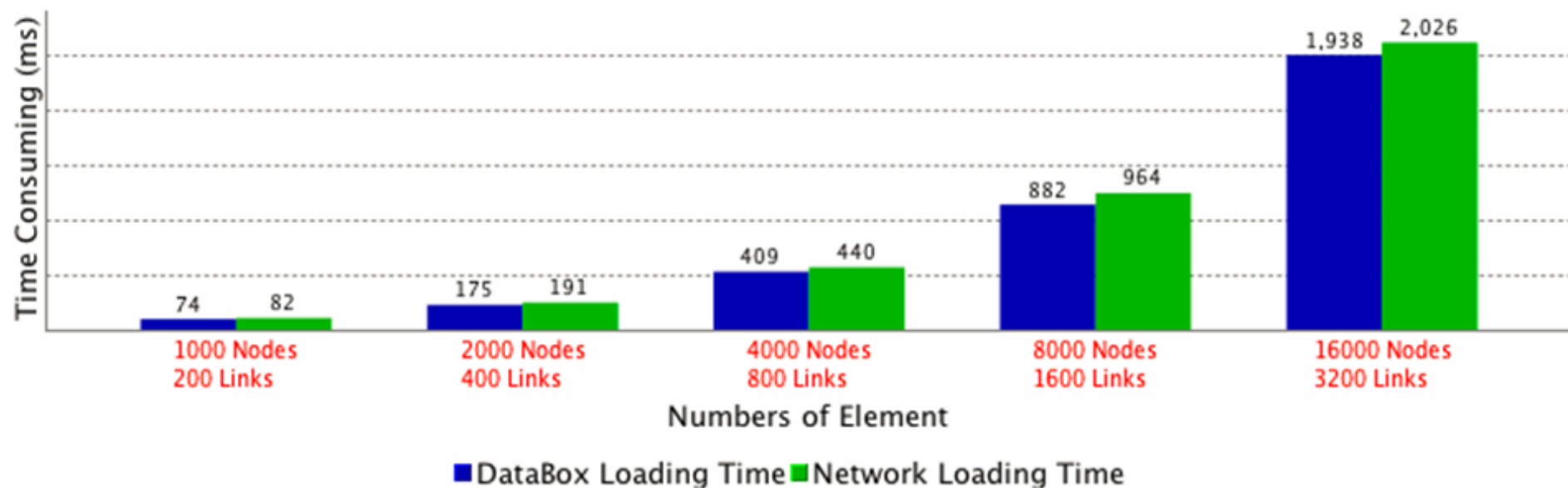




# 高效轻量的图形组件库

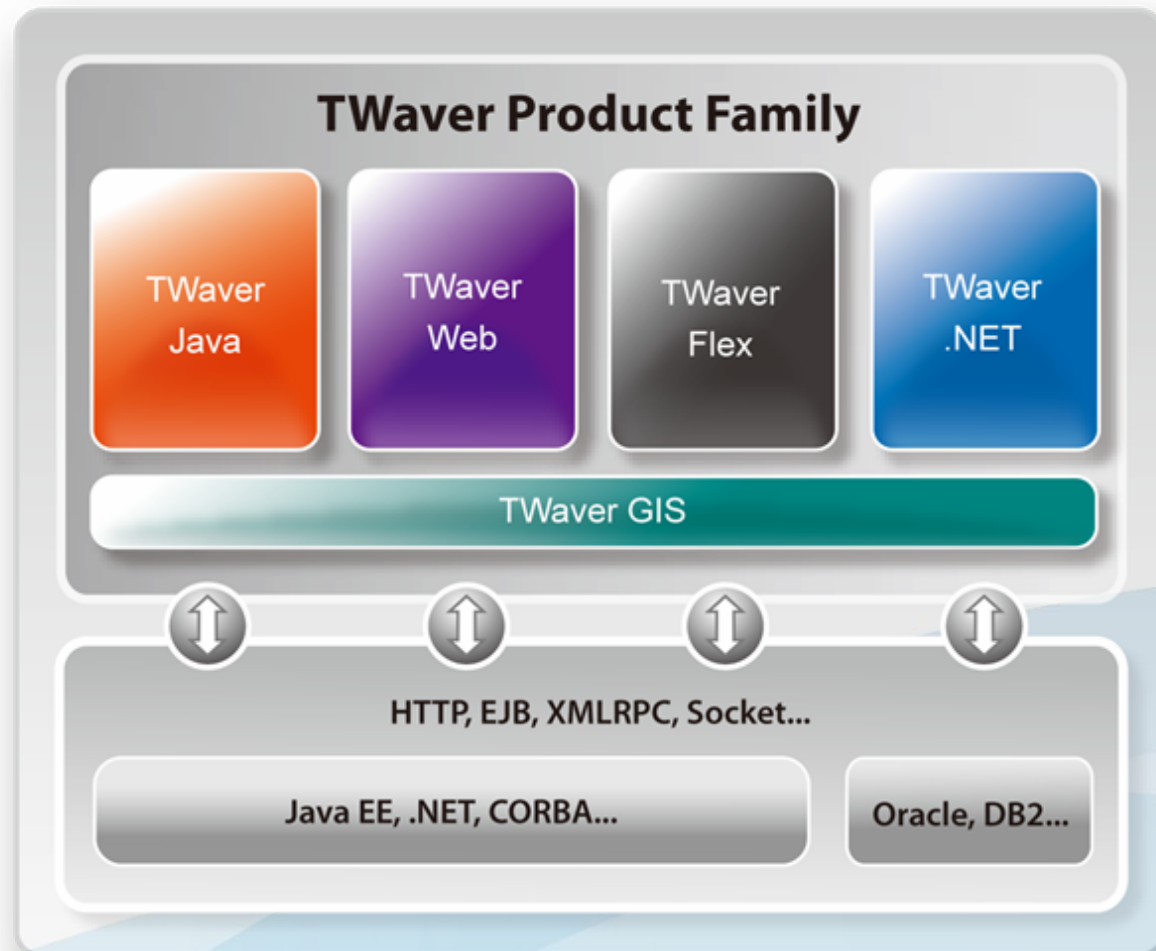
- twaver.jar 1.6MB
- 拓扑图支持上万网元
- 表格支持十万网元

TWaver Performance Report



# 提供多种平台解决方案

- Java
- Web
- Flex
- .NET
- HTML5...



TWaver™

# 丰富的客户应用案例





# TWaver Java产品包结构

TWaver™

twaver-java-3.7	--
demo.bat	4 KB
demo.jar	1.8 MB
demo.sh	4 KB
documents	--
TWaver Java 3.7 Developer Guide.pdf	5.0 MB
twaver-3.7.dtd	4 KB
twaver-beaninfo-3.7.dtd	4 KB
TWaver.template.xml	57 KB
javadoc	--
lib	--
twaver.jar	1.8 MB
License.txt	16 KB
README.txt	4 KB
src	--

demo运行文件

开发手册

API 文档

twaver.jar

demo源代码

# TWaver授权许可

- TWaver许可分三种，试用版，开发版，运行版。查看许可信息可在Swing界面使用 **Ctrl+Shift+T** 快捷键
- **试用版：** 可免费申请，用于前期预言或技术选型阶段，组件界面带有"TWaver Evaluation Version"水印
- **开发版：** 用于项目实际开发阶段，无水印，每隔两个小时会自动显示许可信息对话框
- **运行版：** 用于项目实际运行阶段，无水印，也不会每隔两小时自动显示许可信息对话框

# 许可文件的使用

- **license.dat**是一个文本文件，包含许可相关信息
- 三种使用方式：
  - 直接替换twaver.jar中的/resource/license.dat
  - 放置在项目java源代码目录下/resource/license.dat
  - 使用API调用：  
`TWaverUtil.validateLicense("/resource/license.dat");`

# Java Swing基础

- TWaver与Swing
- 事件派发线程
- LookAndFeel
- Java2D基础



# TWaver与Swing

TNetwork TTree TElementTable ...



TWaver

JTree JPanel JTable ...



JComponent

JFrame

Swing



Container → Window → Frame

AWT

↑  
Component

# TWaver与Swing

TWaver™

TWaver组件	继承于Swing组件
TNetwork	JComponent
TElementTable	JTable
TPropertySheet	JTable
TTree	JTree
***Chart	JPanel
TList	JList
...	...

# 事件派发线程

- 事件派发线程(Event Dispatch Thread), 简称EDT
- EDT是个队列, 上个事件处理完, 才能开始下个事件
- 任何导致界面变化的代码都要在EDT中调用
- 不要在EDT中执行非常耗时的代码
- 如何使用: `SwingUtilities.invokeLater/invokeAndWait`, 排队和插队

```
SwingUtilities.invokeLater(new Runnable() {  
    public void run() {  
        sheet.updateTViewUI();  
    }  
});
```

# EDT与TWaver

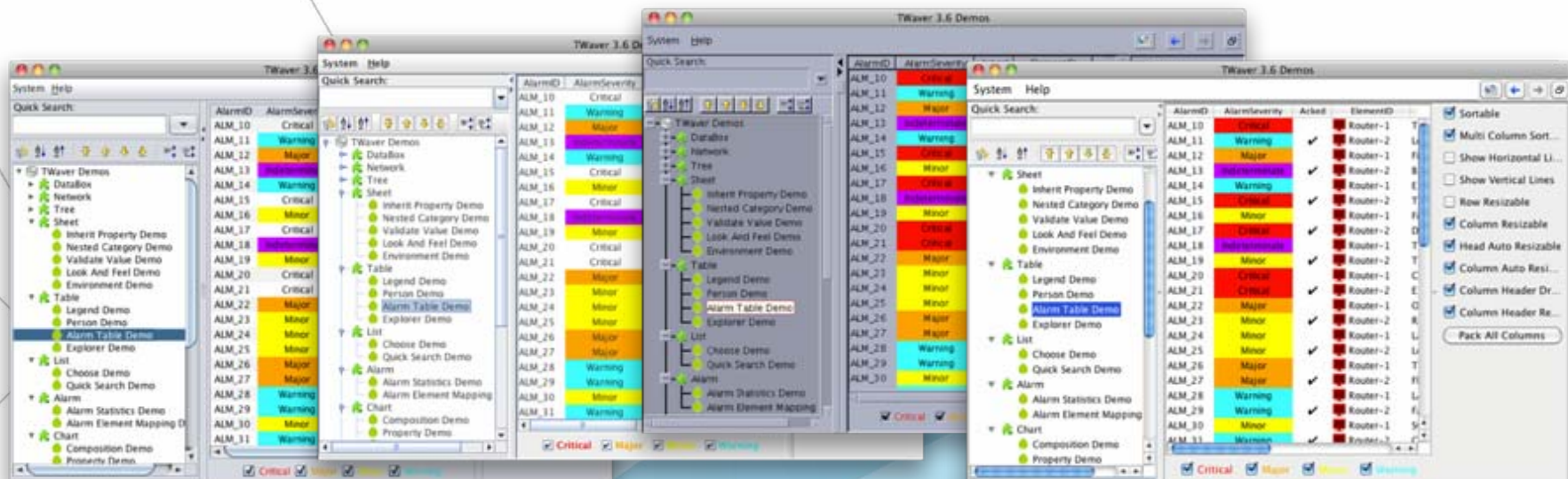
- 对TDataBox, Element的操作都不是线程安全的
- 如果dataBox与组件关联, 则任何对dataBox的操作必须在EDT中执行
- 如果element放在dataBox, 且与组件关联, 则对element设置属性需要在EDT中执行
- 一个element只能放在一个dataBox中



# LookAndFeel

- UIManager中包含LookAndFeel的属性信息
- 使用get\*\*(key)和put(key, value)来设置默认属性
- 使用 setLookAndFeel (...)设置look and feel

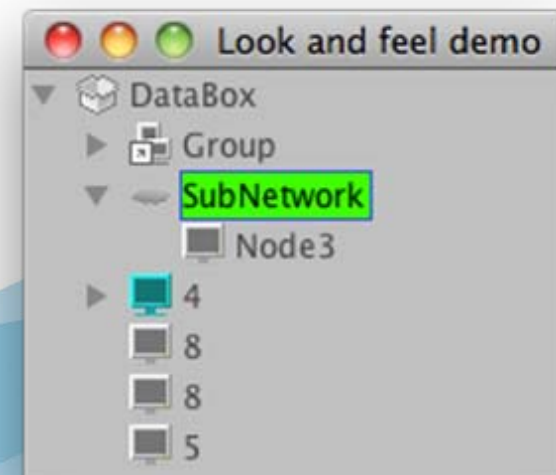
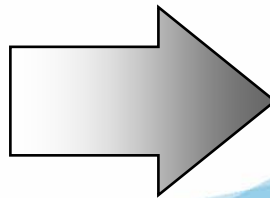
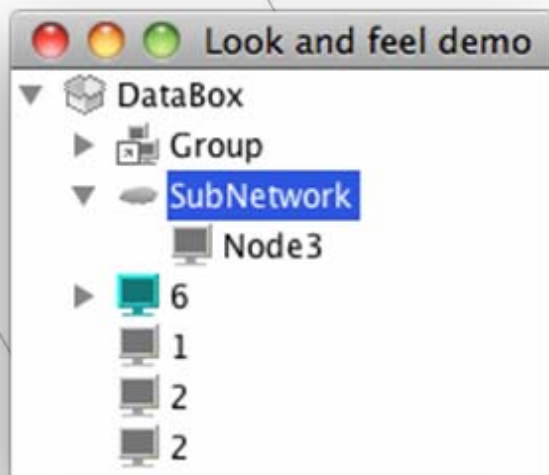
UIManager.setLookAndFeel("com.sun.java.swing.plaf.nimbus.NimbusLookAndFeel");



# LookAndFeel

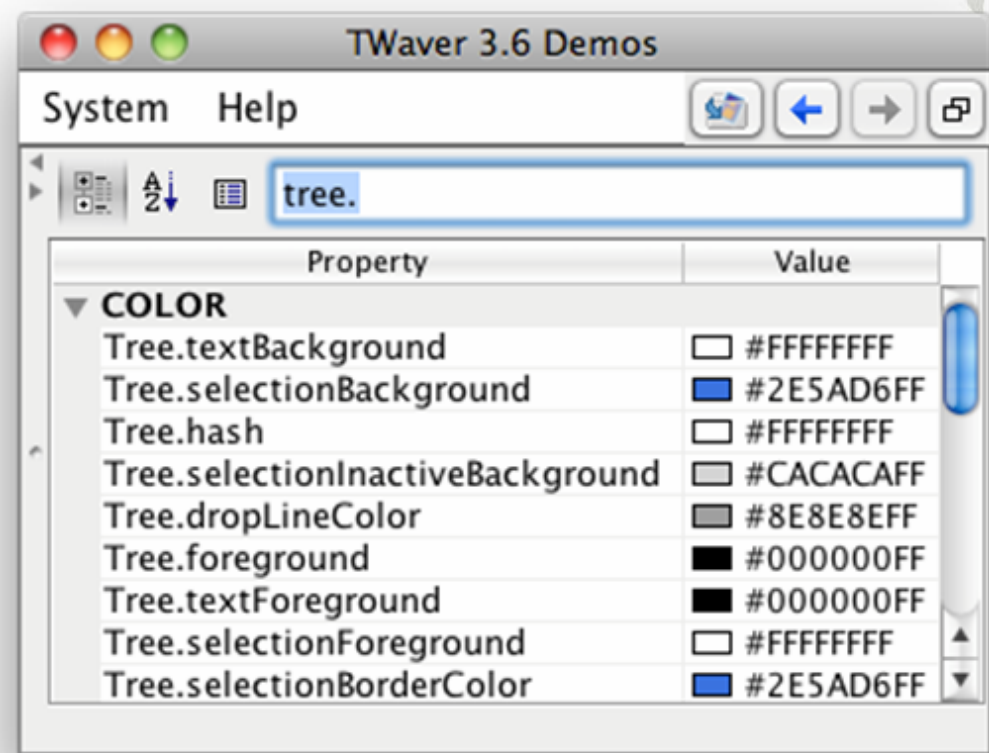
```
UIManager.put("Tree.textForeground", Color.DARK_GRAY);  
UIManager.put("Tree.textBackground", Color.LIGHT_GRAY);  
UIManager.put("Tree.background", Color.LIGHT_GRAY);  
UIManager.put("Tree.selectionForeground", Color.BLACK);  
UIManager.put("Tree.selectionBackground", Color.GREEN);
```

```
TestUtil.showFrame("Look and feel demo", TestUtil.getTree());
```



# TWaver的LookAndFeel查询工具

- LookAndFeelDemo.java
- 使用属性表组件列举了Swing中所有的lookAndFeel默认属性，分组显示，可以方便查询
- 见TWaver Java Demo



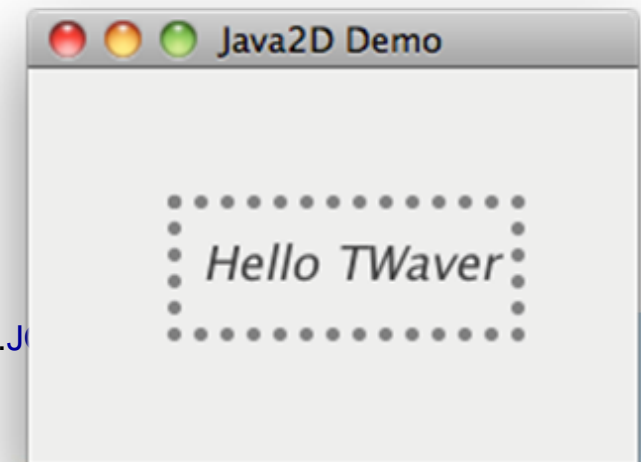
# Java2D

- 扩展UI，你需要了解Java2D相关知识
- Graphics2D
- Shape, Area, Paint, Composite
- Stroke, AffineTransform
- ...



# Java2D示例

```
public class Java2DDemo extends JComponent {  
    protected void paintComponent(Graphics g) {  
        Graphics2D g2d = (Graphics2D) g;  
        g2d.setRenderingHint(RenderingHints.KEY_ANTIALIASING, RenderingHints.VALUE_ANTIALIAS_ON);  
        Rectangle bounds = this.getBounds();  
        g2d.translate(bounds.getCenterX(), bounds.getCenterY());  
        g2d.setColor(Color.DARK_GRAY);  
        g2d.setFont(new Font(Font.SANS_SERIF, Font.ITALIC, 18));  
        g2d.drawString("Hello TWaver", -50, 5);  
        g2d.setColor(Color.GRAY);  
        g2d.setStroke(new BasicStroke(5, BasicStroke.CAP_ROUND, BasicStroke.JOIN_ROUND, 10, 0));  
        g2d.drawRect(-60, -25, 130, 50);  
    }  
    public static void main(String[] args) {  
        TestUtil.showFrame("Java2D Demo", new Java2DDemo());  
    }  
}
```



# TWaver基础

- Hello TWaver
- MVC设计模式(数据驱动事件机制组件联动)
- 数据元素 / 数据容器介绍
- 组件类型介绍

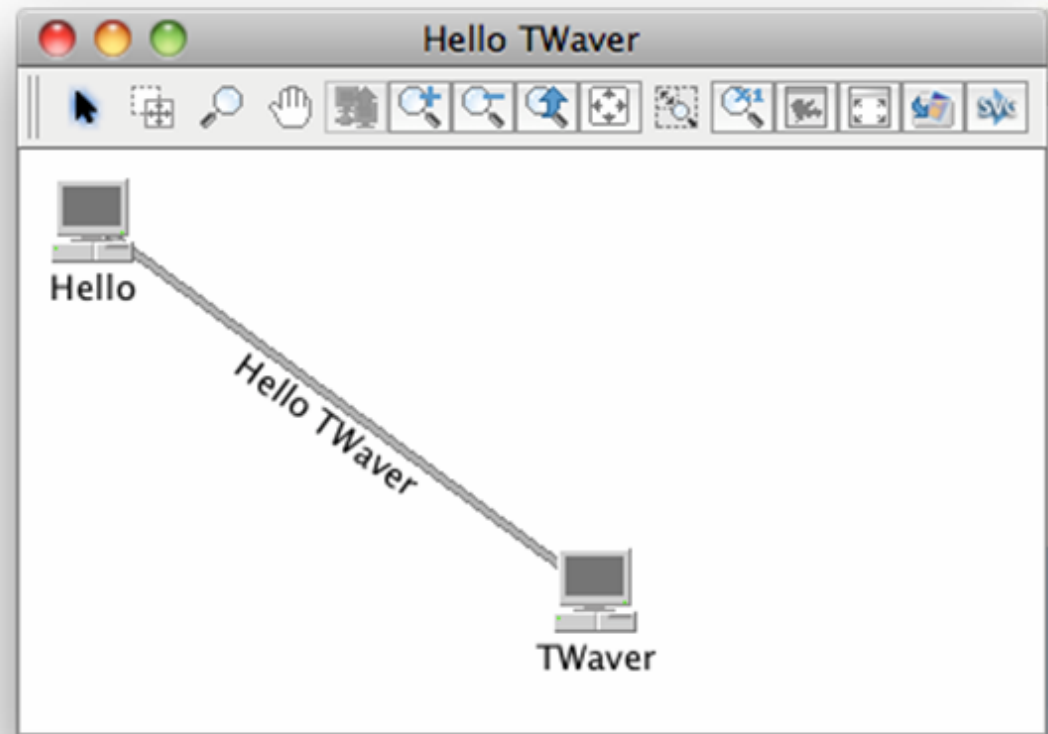
# Hello Twaver

## Twaver Java 开发环境

- twaver.jar
- JDK 1.4+
- Eclipse, NetBeans, JBuilder ...

# Hello TWaver

```
TDataBox box = new TDataBox();  
Node node = new Node();  
node.setName("Hello");  
node.setLocation(10, 10);  
box.addElement(node);  
  
Node node2 = new Node();  
node2.setName("TWaver");  
node2.setLocation(200, 150);  
box.addElement(node2);  
  
Link link = new Link(node, node2);  
link.setName("Hello TWaver");  
link.putLinkLabelRotatable(true);  
box.addElement(link);  
  
TNetwork network = new TNetwork(box);  
  
showFrame("Hello TWaver", network);
```



TWaver™



# 增加Tree, Sheet, Table

```
TNetwork network = new TNetwork(box);
```

```
TTree tree = new TTree(box);
```

```
TPropertySheet sheet = new TPropertySheet(box);
```

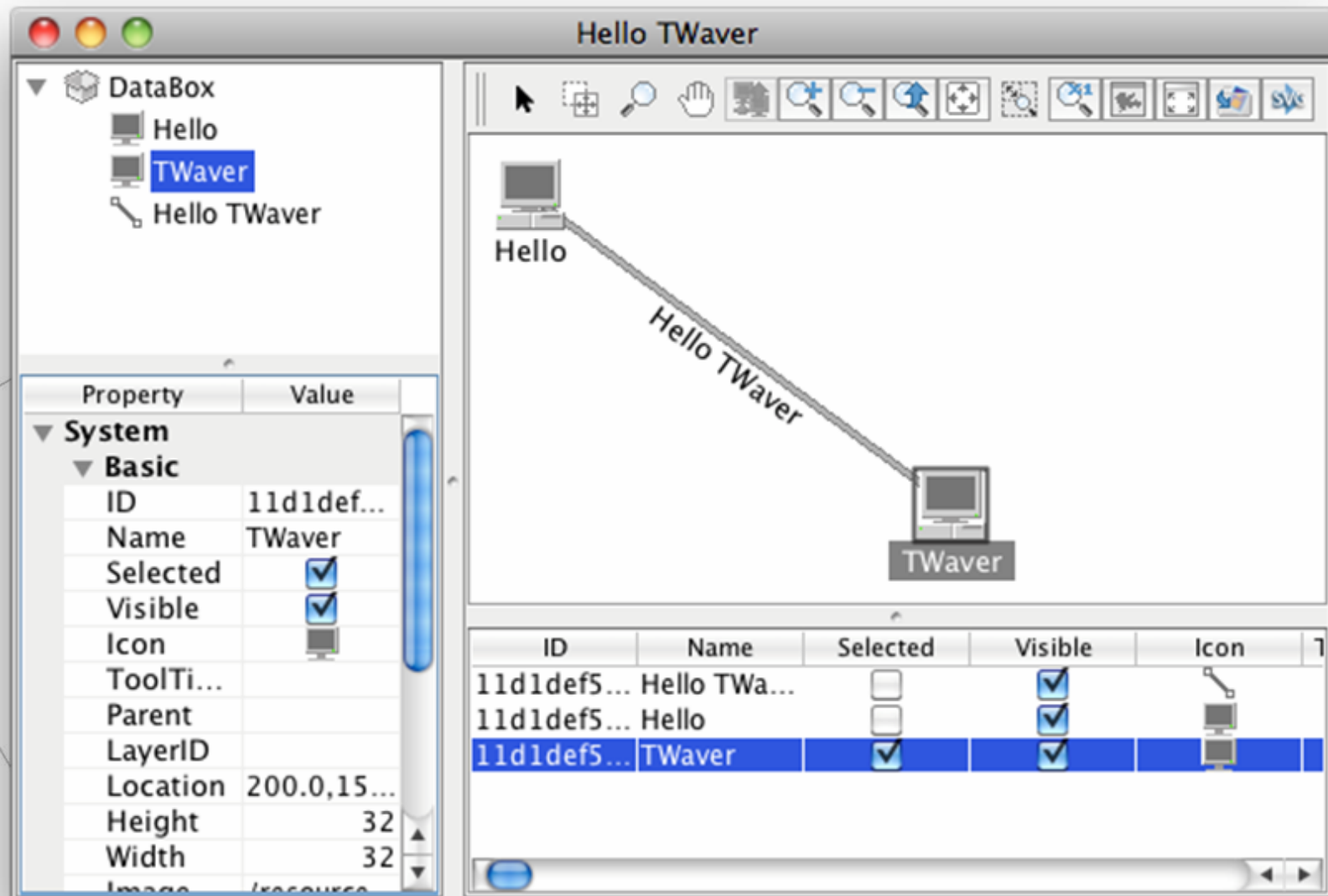
```
TElementTable table = new TElementTable(box);  
table.setElementClass(Element.class);
```

```
JScrollPane tablePanel = new JScrollPane(table);
```

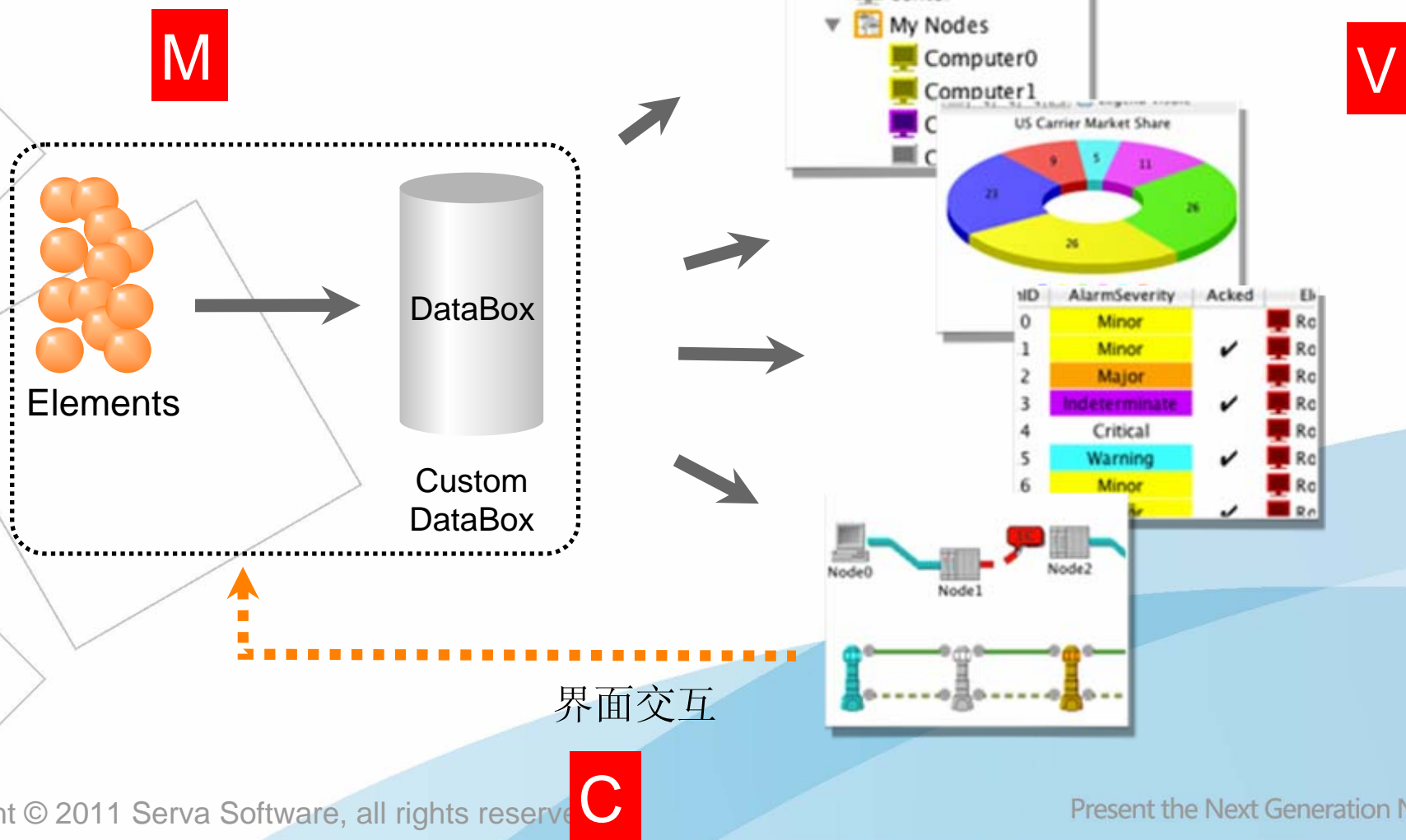
```
JScrollPane sheetPanel = new JScrollPane(sheet);
```

# Hello TWaver

TWaver™

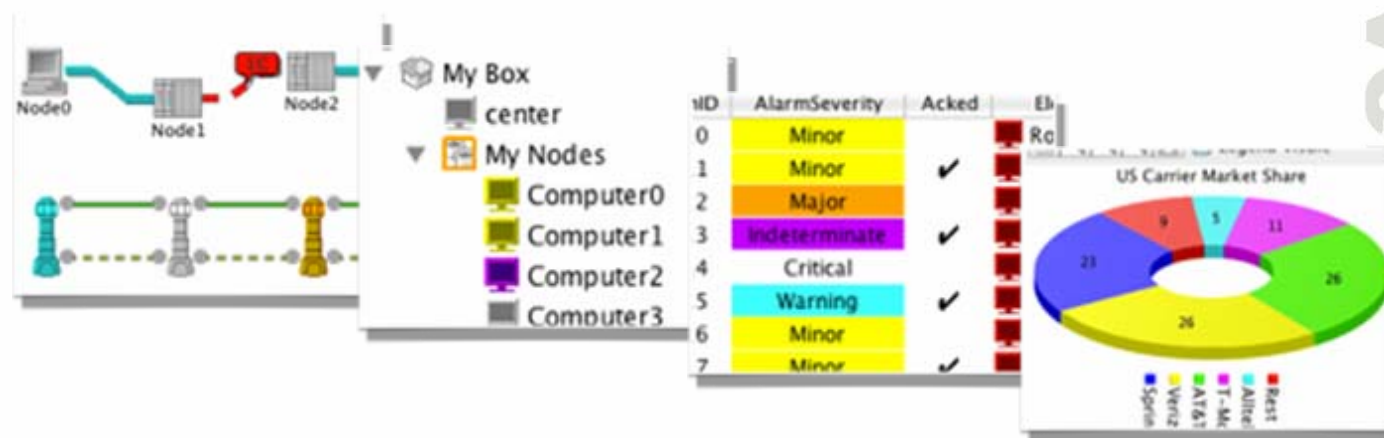


# MVC设计模式



# 数据驱动

组件(V)



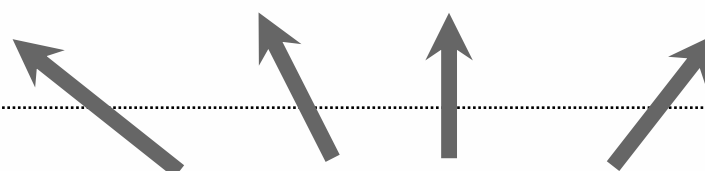
模型(M)

Element

Node  
Link  
Group  
...

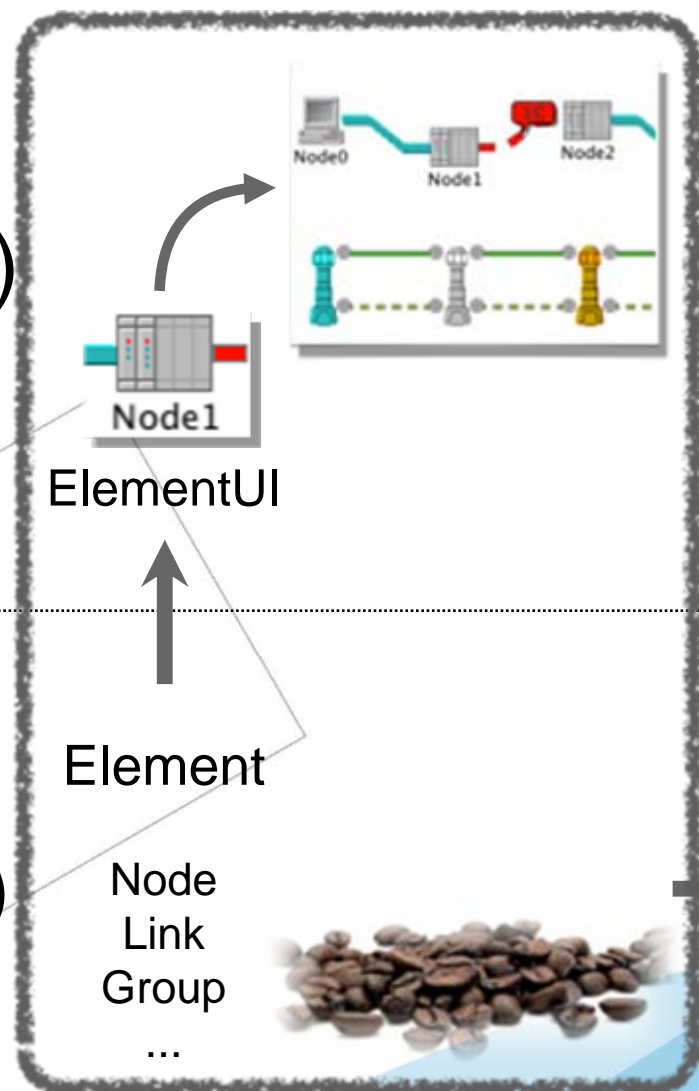


TDataBox



# 数据驱动

组件(V)

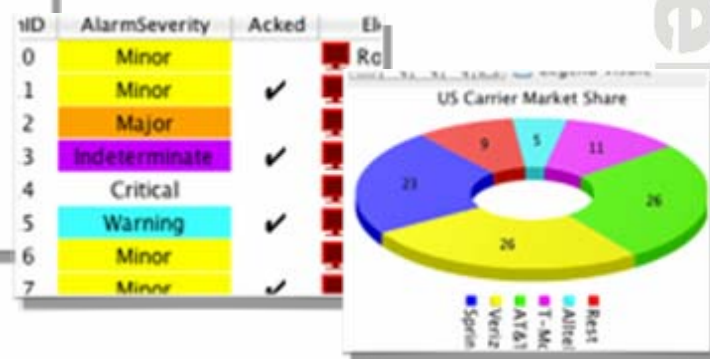


ElementUI

Element

Node  
Link  
Group  
...

TDataBox

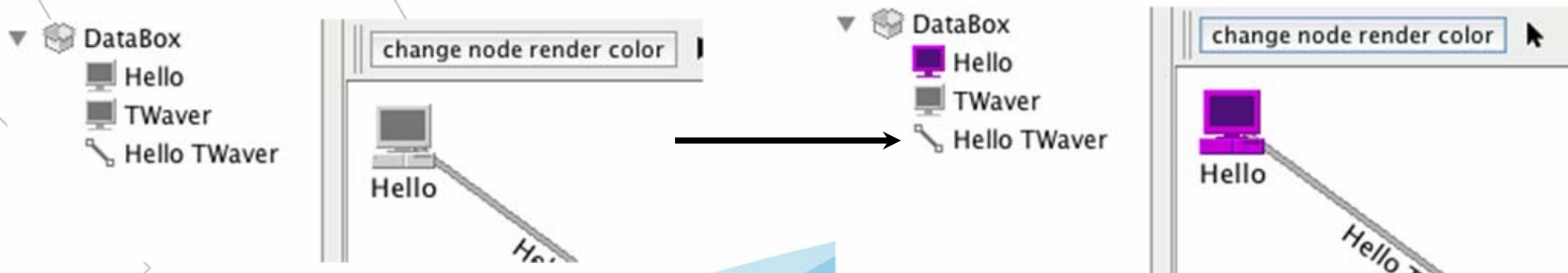




# 数据驱动

```
JButton changeRenderColorButton = new JButton("change node render color");
changeRenderColorButton.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        node.putRenderColor(TWaverUtil.getRandomColor());
    }
});
network.getToolBar().add(changeRenderColorButton, 0);
```

修改数据，视图自动刷新



# 事件机制

**Element**

node.putRenderColor(Color.RED)

firePropertyChange

**DataBox**

elementPropertyChangeListeners

fireElementPropertyChange

**View**

network, tree, chart ...

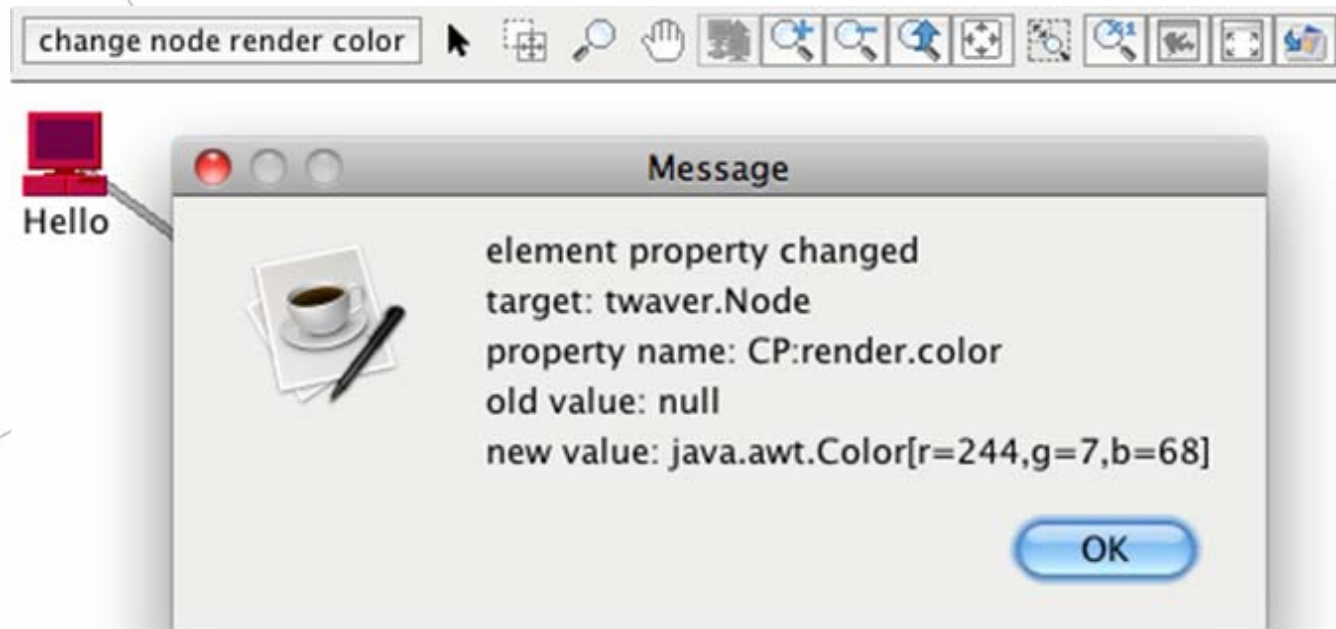
dataBoxPropertyChangeListener

repaint(element)



# 事件监听

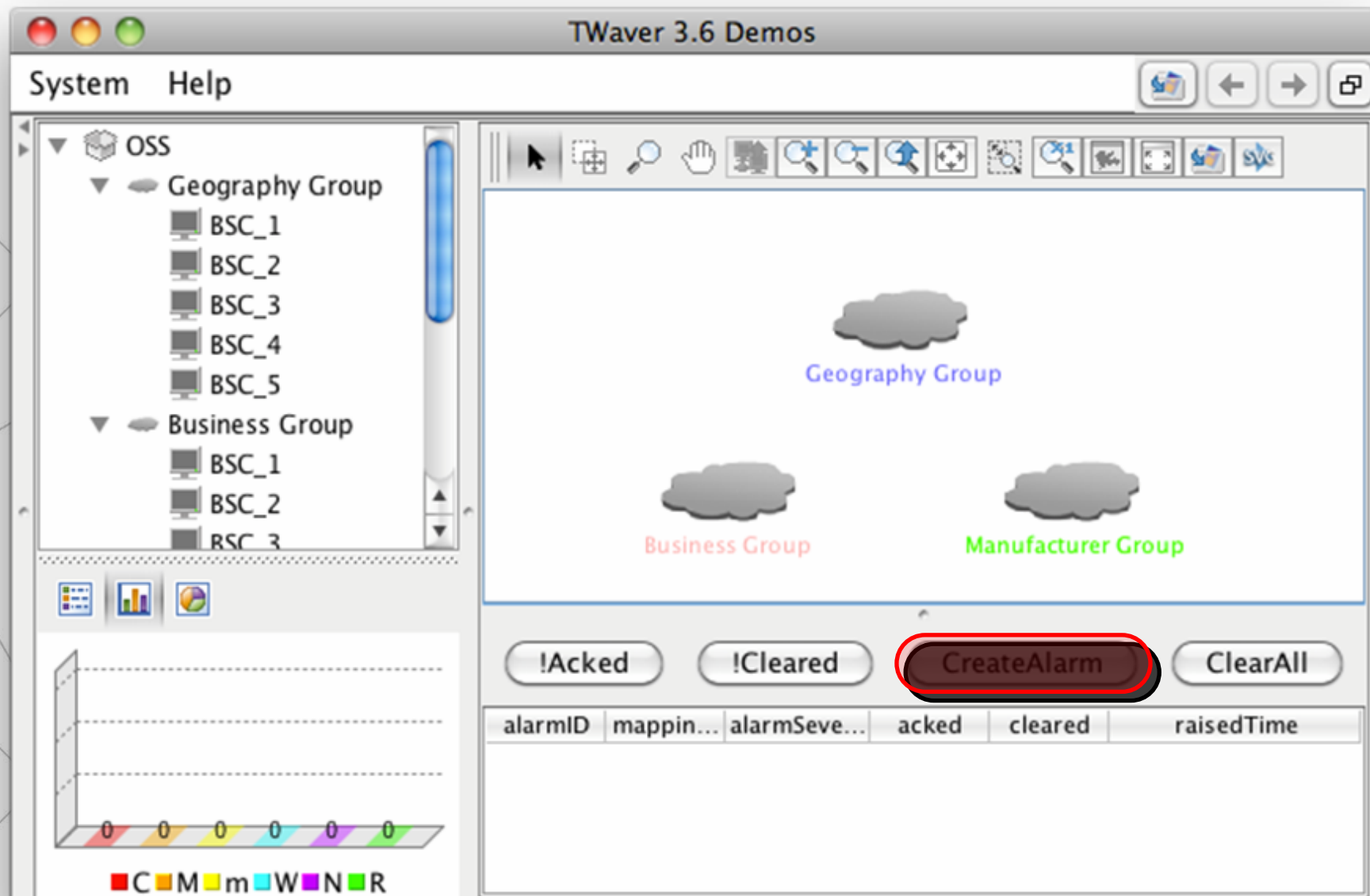
```
box.addElementChangeListener(new PropertyChangeListener() {  
    public void propertyChange(PropertyChangeEvent evt) {  
        JOptionPane.showMessageDialog(network,  
            "element property changed\ntarget: " + evt.getSource() +  
            "\nproperty name: " + evt.getPropertyName() +  
            "\nold value: " + evt.getOldValue() +  
            "\nnew value: " + evt.getNewValue());  
    }  
});
```



# 事件监听

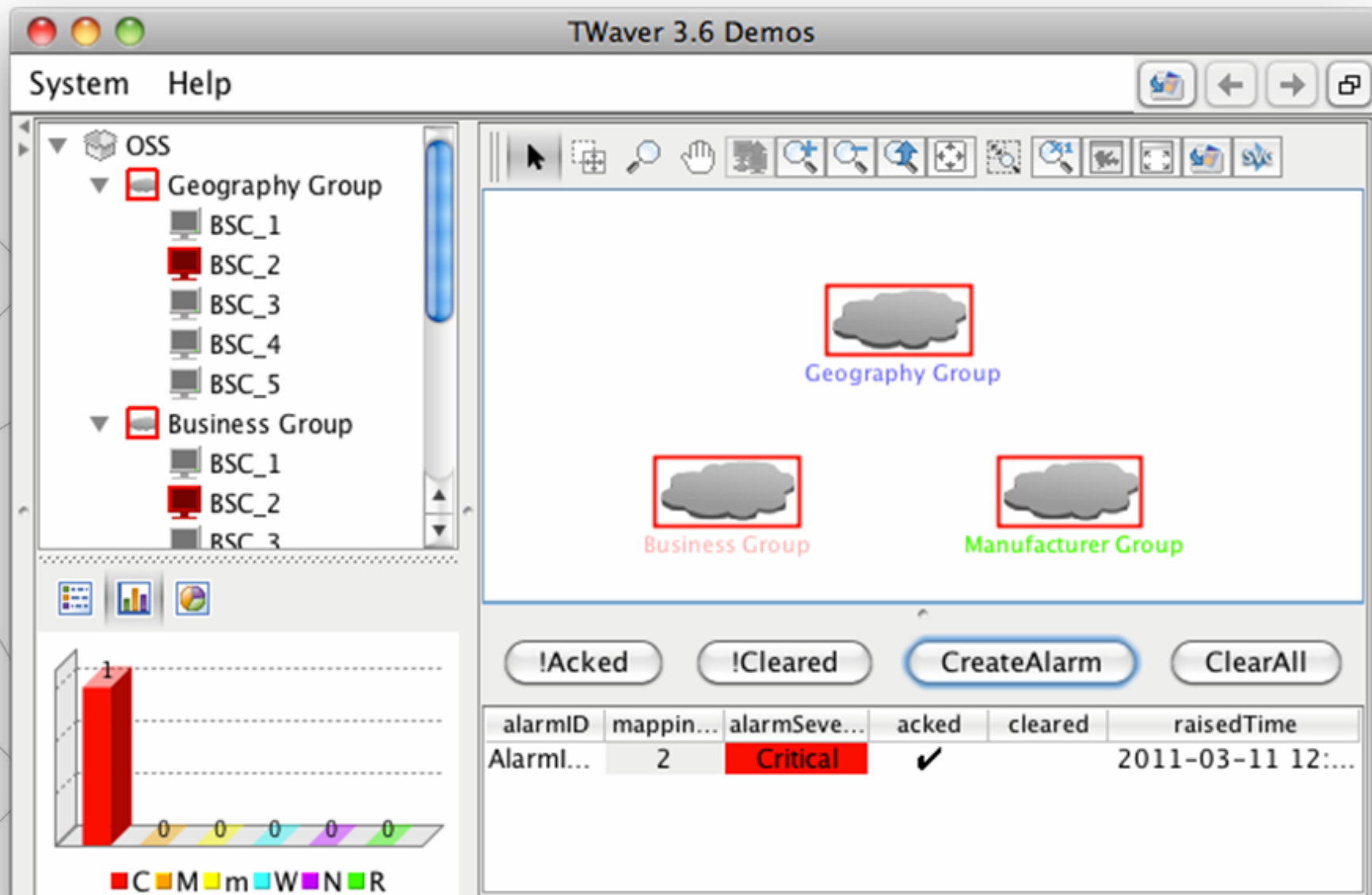
Model	Listener
TDataBox	DataBoxListener DataBoxSequenceListener BatchListener
DataBoxSelectionModel	DataBoxSelectionListener
UndoRedoManager	UndoRedoListener
AlarmModel	AlarmModelListener
AlarmSeverity	AlarmSeverityChangeListener
Network	ZoomListener InteractionListener CanvasPaintListener
LayerManager	LayerModelListener
TTable	PageListener

# 组件联动





# 组件联动



# 数据元素 / 数据容器

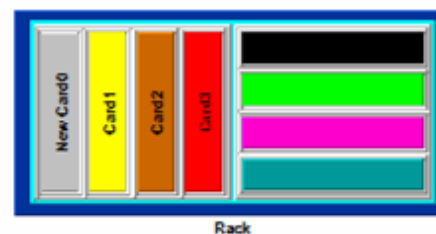
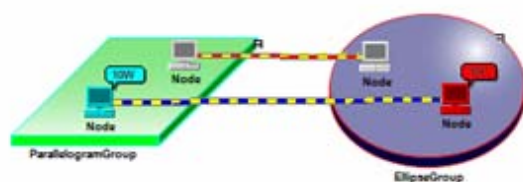
**数据元素：**数据的基本单位

**数据容器：**管理数据元素集合的容器，提供对数据元素的增减操作，监听数据元素的属性变化

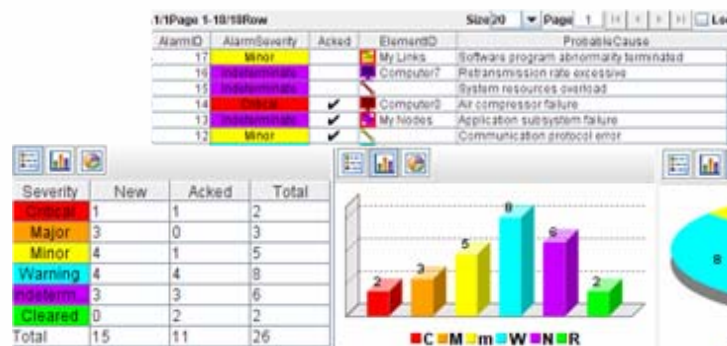
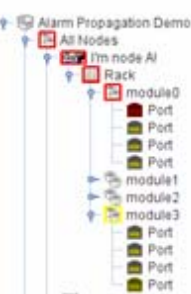
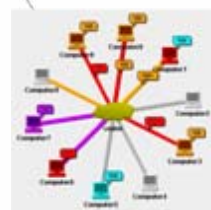
数据元素	数据容器
Element	TDataBox
Alarm	AlarmModel
Layer	LayerModel

# 数据元素类型介绍

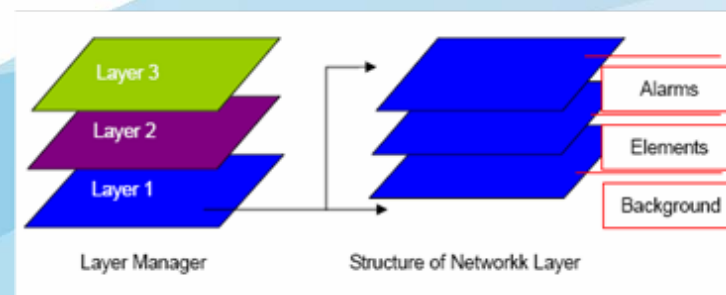
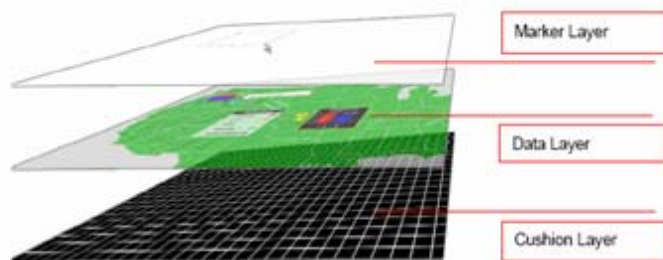
Element (Node / Link / Group / SubNetwork / ...)



Alarm

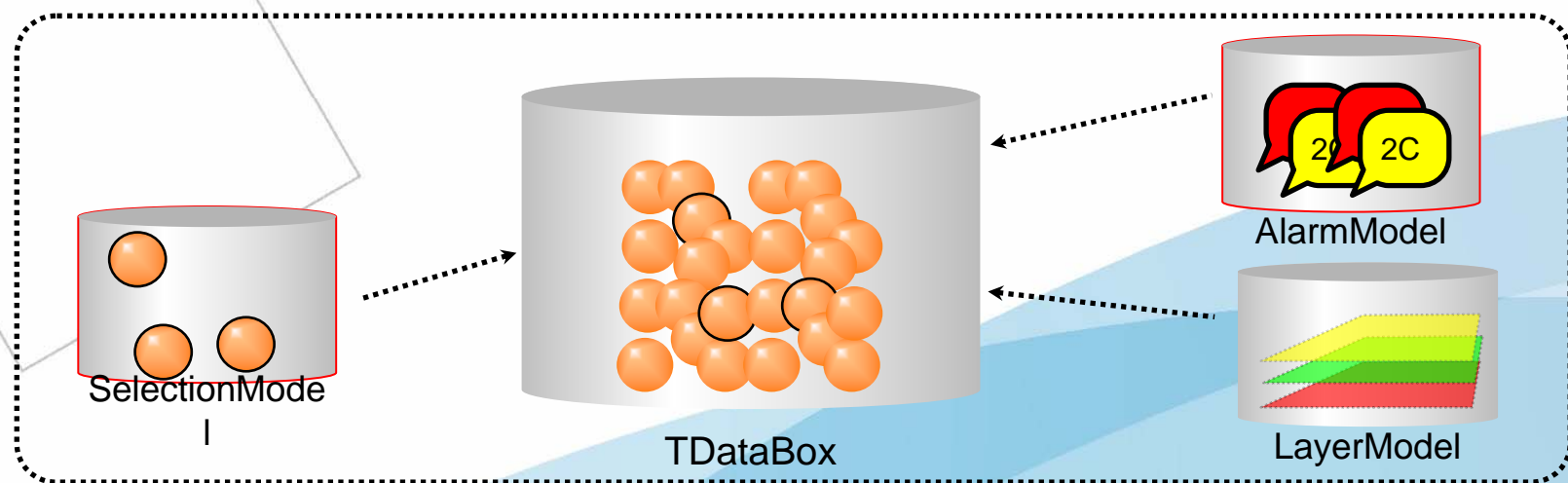


Layer



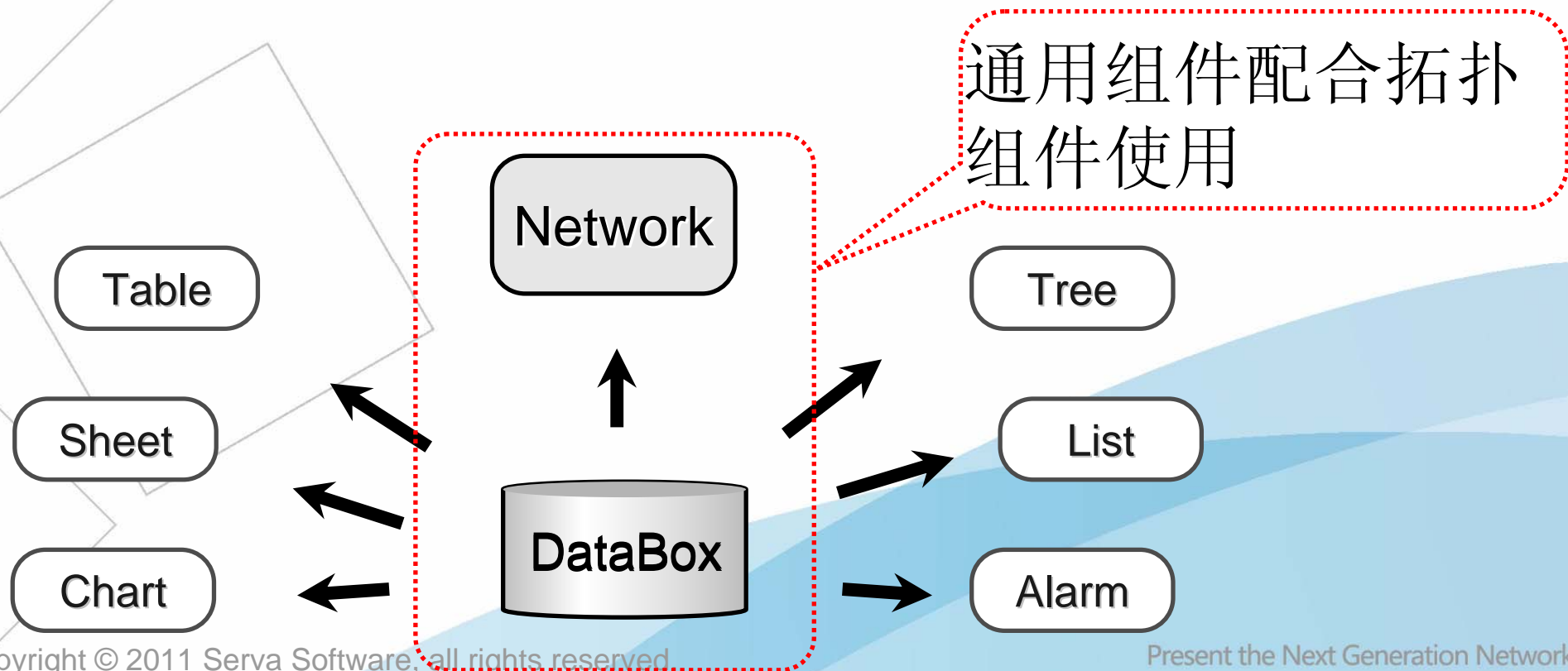
# 数据容器之间关系

- **TDataBox**是管理网元的数据容器，同时他也包含了告警容器，图层容器，选中模型等。



# 组件类型

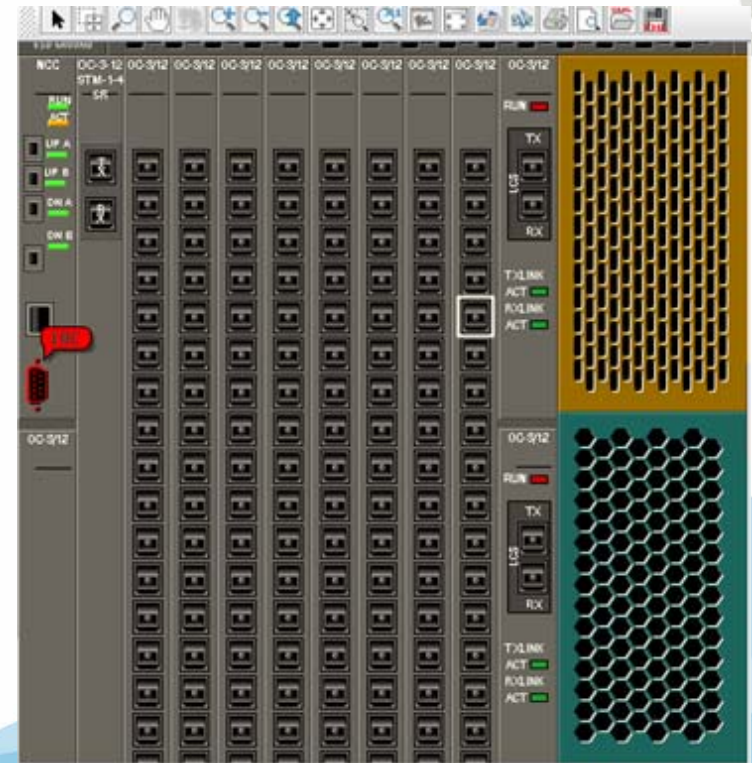
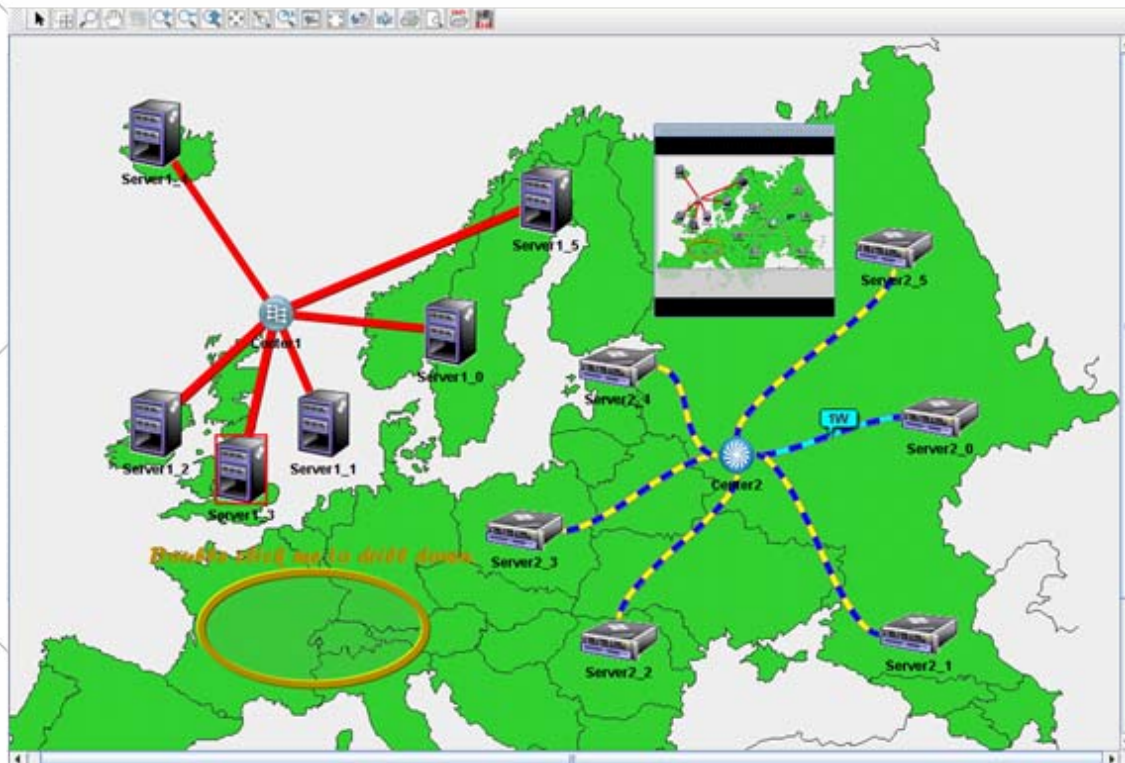
- 拓扑组件: Network
- 通用组件: Tree, Table, Sheet, TreeTable, Chart





# Network

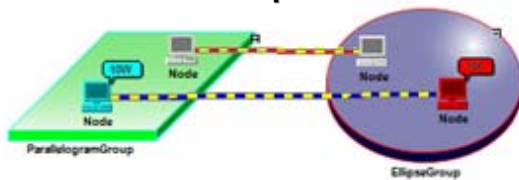
呈现拓扑图，设备面板...



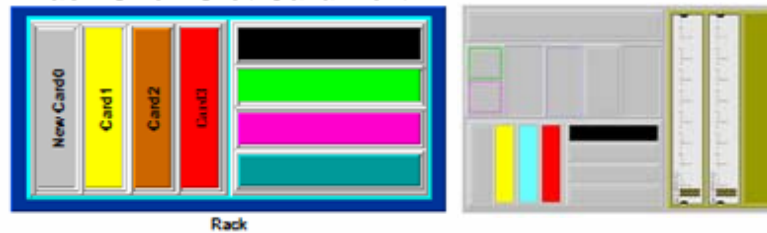
TWaver™

# 拓扑网元

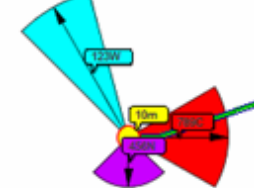
Node Link Group



Rack Shelf Slot Card Port



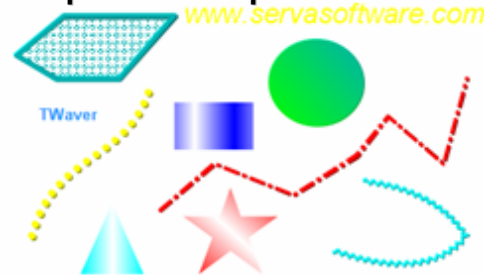
BTS Antenna



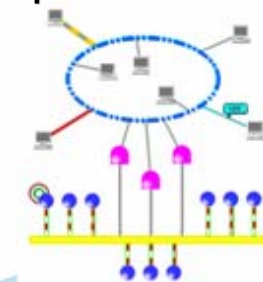
SubNetwork



ShapeNode ShapeLink CustomVector Text



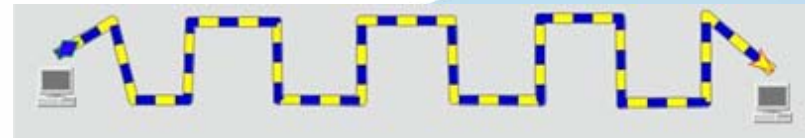
EllipseBus LineBus



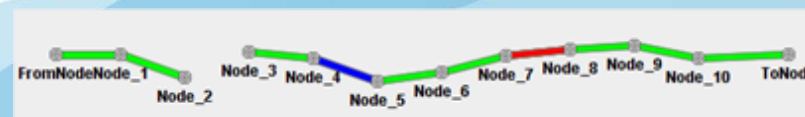
ResizableNode Follower



ShapeLink



Polyline

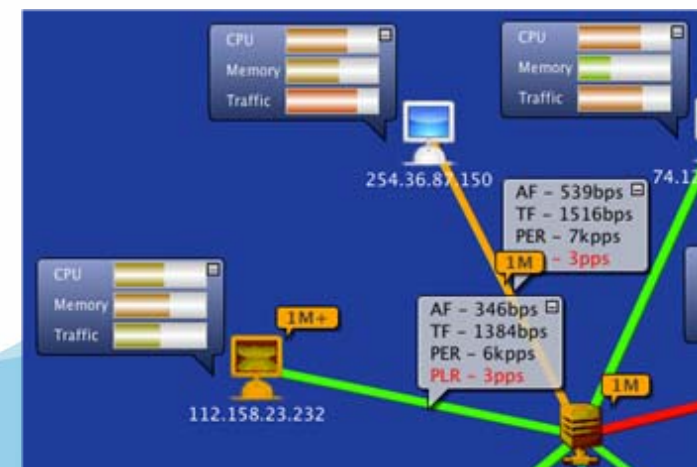
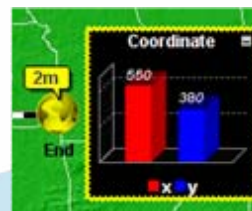
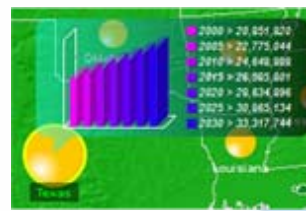
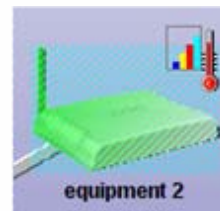
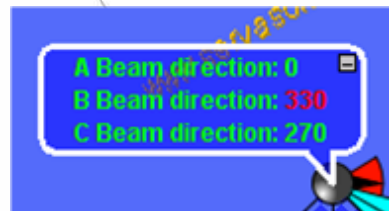
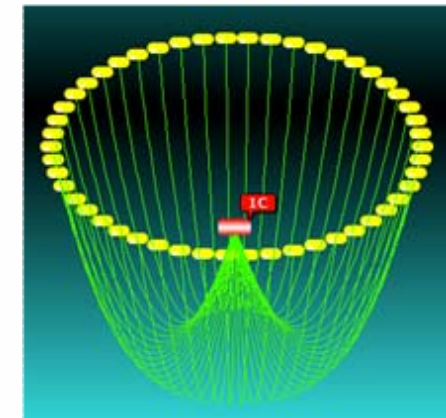




# UI定制扩展

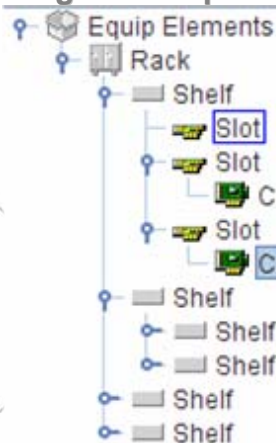


	0	1	2	3	4	5	6	7	8
0	0-0	0-1	0-2	0-3	0-4	0-5	0-6	0-7	0-8
1	1-0	1-1	1-2	1-3	1-4	1-5	1-6	1-7	1-8
2	2-0	2-1	2-2	2-3	2-4	2-5	2-6	2-7	2-8
3	3-0	3-1	3-2	3-3	3-4	3-5	3-6	3-7	3-8

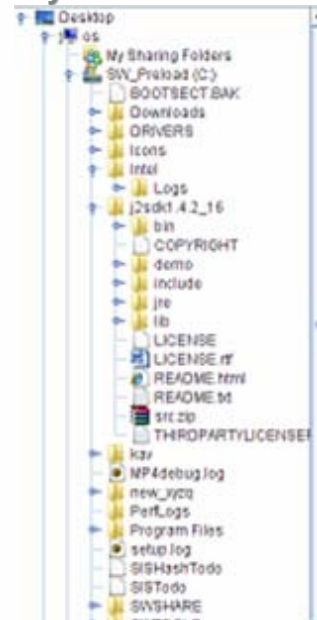


# Tree

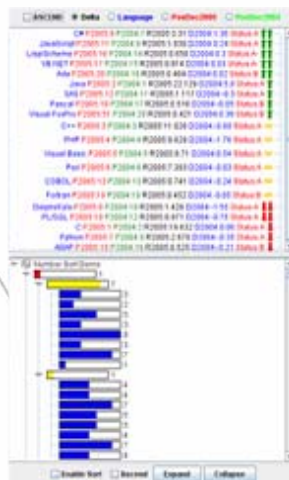
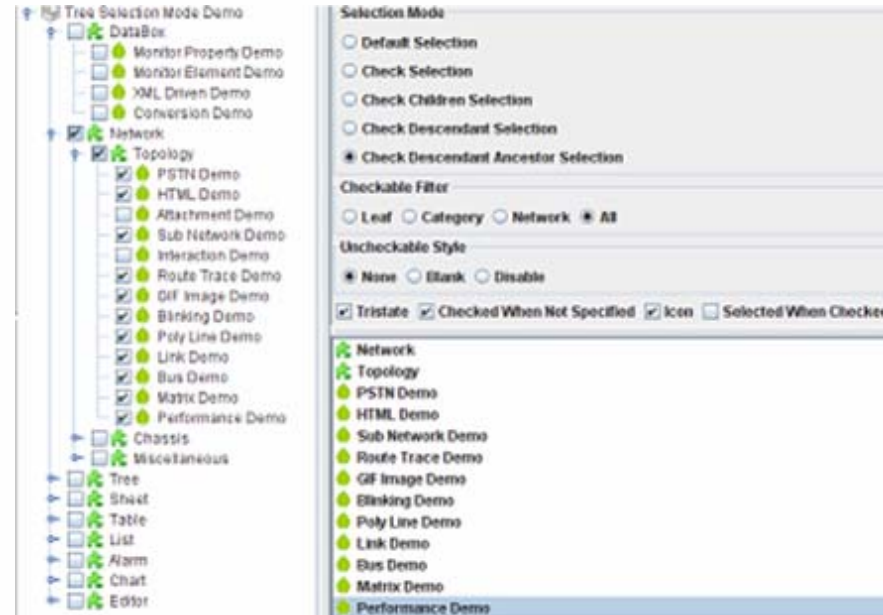
## Drag and Drop



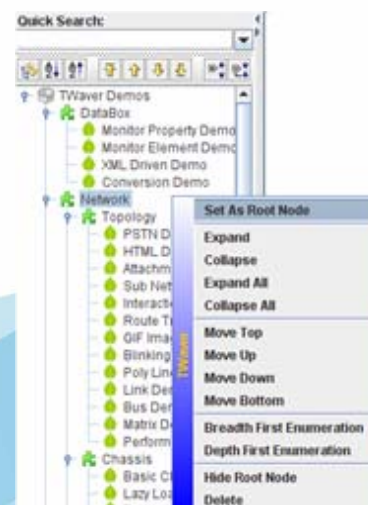
## Lazy Load



## Check Selection Mode



## Compare Sort Custom Icon



## Switch Root Node Move node up/down/top/bottom Breadth/Depth First Enumeration Visible Filter



# Table

VisibleFilter Multi-Column Sort

First Name	Last Name	Full Name	Age	Birth	First Name	Last Name	Full Name	Salary	Married	Sex	Severity	Description
John	Bravo	John Bravo	31	1974	John	Bravo	John Bravo	\$1000.0	<input type="checkbox"/>	M	Warning	John Bravo with monthly salary of \$1000.0
Frank	Barker	Frank Barker	40	1977	Frank	Barker	Frank Barker	\$2000.0	<input type="checkbox"/>	M	Critical	Frank Barker with monthly salary of \$2000.0
Celso	Aguiar	Celso Aguiar	34	1923	Celso	Aguiar	Celso Aguiar	\$1000.0	<input type="checkbox"/>	M	Major	Celso Aguiar with monthly salary of \$1000.0
Clyde	Ashe	Clyde Ashe	31	1924	Clyde	Ashe	Clyde Ashe	\$1000.0	<input type="checkbox"/>	M	Critical	Clyde Ashe with monthly salary of \$1000.0
Arnd	Batra	Arnd Batra	34	1943	Arnd	Batra	Arnd Batra	\$11000.0	<input type="checkbox"/>	M	Minor	Arnd Batra with monthly salary of \$11000.0
Jason	Boyer	Jason Boyer	42	1947	Jason	Boyer	Jason Boyer	\$1000.0	<input type="checkbox"/>	M	Warning	Jason Boyer with monthly salary of \$1000.0
Christopher	Brickford	Christopher Brickford	38	1943	Christopher	Brickford	Christopher Brickford	\$1000.0	<input type="checkbox"/>	M	Critical	Christopher Brickford with monthly salary of \$1000.0
Purnell	Akops	Purnell Akops	48	1959	Purnell	Akops	Purnell Akops	\$1000.0	<input type="checkbox"/>	M	Critical	Purnell Akops with monthly salary of \$1000.0
Gangadhar	Bathula	Gangadhar Bathula	43	1964	Gangadhar	Bathula	Gangadhar Bathula	\$1000.0	<input type="checkbox"/>	M	Critical	Gangadhar Bathula with monthly salary of \$1000.0
Art	Americus	Art Americus	42	1951	Art	Americus	Art Americus	\$1000.0	<input type="checkbox"/>	M	Minor	Art Americus with monthly salary of \$1000.0

TreeTable PackColumn

File	File Size	File Type	Last Modified
Desktop			2008-09-27 18:58:58
My Sharing Folders		System Folder	1970-01-01 08:00:00
SW_Preload (C:)		Local Disk	2008-09-27 21:26:19
BOOTSECT.BAK	8 KB	BAK File	2006-11-10 09:23:37
Downloads		File Folder	2008-09-27 20:56:37
DRIVERS		File Folder	2007-11-27 17:13:35
Icons		File Folder	2007-11-27 16:46:09
Intel		File Folder	2007-11-27 17:29:37
J2sdk1.4.2_16		File Folder	2008-01-28 21:44:28
bin		File Folder	2008-01-28 21:44:12
COPYRIGHT	4 KB	File	2007-09-17 02:21:50
demo		File Folder	2008-01-28 21:43:31
include		File Folder	2008-01-28 21:43:17
jre		File Folder	2008-01-28 21:44:21
lib		File Folder	2008-01-28 21:44:12
LICENSE	19 KB	File	2007-09-17 01:33:46
LICENSE.txt	20 KB	RTF 格式	2007-09-17 01:33:46
README.html	18 KB	HTML Document	2007-09-17 01:33:46
README.txt	11 KB	Text Document	2007-09-17 01:33:46
src.zip	11,550 KB	WinRAR ZIP archive	2007-09-17 01:33:19
THIRDPARTYLICENSEREADME.txt	10 KB	Text Document	2008-01-28 21:43:16
jav		File Folder	2008-01-29 15:02:07
MP4dubup.log	1 KB	Text Document	2008-02-18 11:44:59
Realtek		File Folder	2008-01-28 21:43:31

Binding AlarmModel Paging Lock

1/3Page 1-20/58Row			Size 20 Page 1			Lock	
AlarmID	AlarmSeverity	Acked	ElementID	ProbableCause	key2		
57	Major		Computer9	Loss of multi frame	2008-09-27 22:16:28	Th	
56	Minor		Computer2	Invalid MSU received	2008-09-27 22:16:27	Th	
55	Critical		Computer5	Link failure	2008-09-27 22:16:26	Th	
54	Warning			High wind	2008-09-27 22:16:25	Th	
53	Indeterminate			Data set or modem error	2008-09-27 21:13:37	Th	
52	Critical			Battery breakdown	2008-09-27 21:13:36	Th	





# Sheet

Filter &amp; Sort Properties

**Property Sheet**

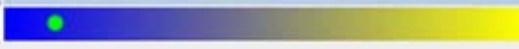



Filter & Sort Properties

Property	Value
<b>System</b>	
Equipment...	
Equipment...	<input checked="" type="checkbox"/>
Height	22
Width	27
Image	jar:file:/C:/Doc...
Location	221,190
Color	
ID	39e706a46ad5...
Name	PORT4
Icon	
ToolTipText	
AlarmState	
Parent	 39e706a4...
LabelFont	Verdana Bold/B...
LabelColor	
LabelPosit...	Top
LabelVisible	<input checked="" type="checkbox"/>
HasLabelB...	<input type="checkbox"/>
LabelSelec...	<input checked="" type="checkbox"/>
LabelXOffs...	0
LabelYOffs...	0

(None)  
No description.

Nested Category &amp; Property Description

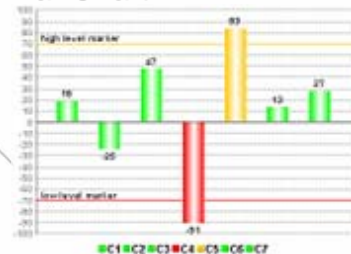
Nested Category & Property Description

Property	Value
<b>System</b>	
GROUP1	
PropertyA	ABC
PropertyB	123456789@#\$%^&
1980-03-06~2080-04-23	
Scale	
GROUP2	
Married	<input checked="" type="checkbox"/>
Color	 #0A5A33FF
State	 STOPPED
City	New York
Language	53
Character	<T>
Proportion	 0.8%

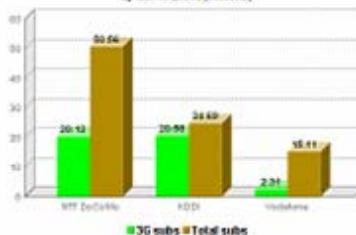
1980-03-06~2080-04-23:

# Chart

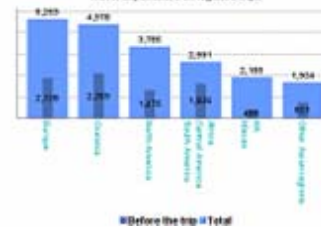
## Bar Chart



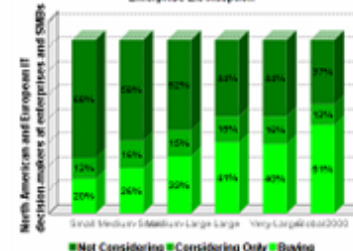
## Japan's cellular subs & 3G penetration by carrier, 2005 (millions)



## Travel expenditure on leisure trips



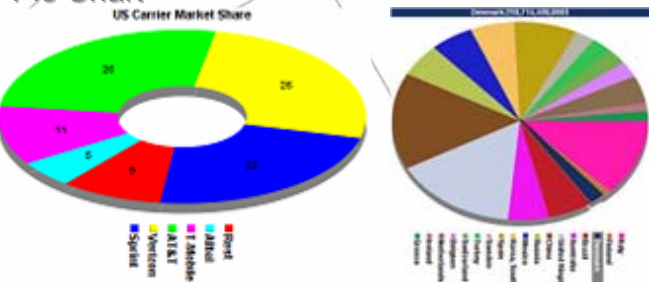
## Enterprise 2.0 Adoption



## Dial Chart



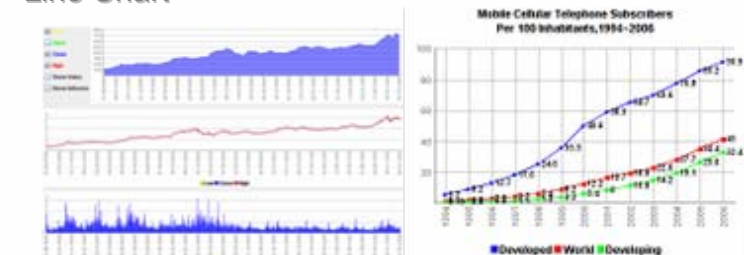
## Pie Chart



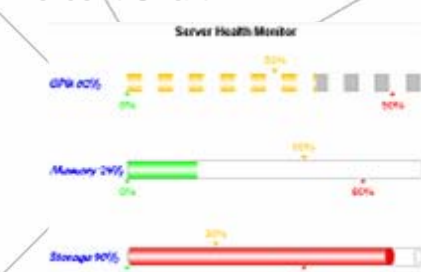
## Radar Chart



## Line Chart



## Percent Chart



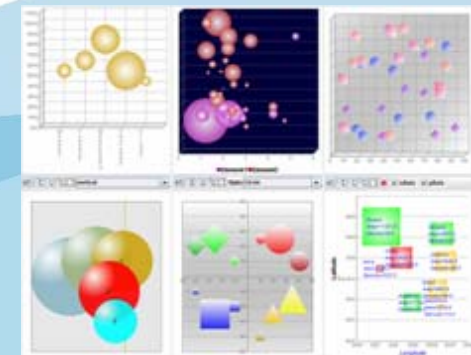
## Using Ajax frameworks, toolkits, or libraries:



## Using Ajax in conjunction with:



## Bubble Chart





- 论坛: [twaver.servasoft.com/forum](http://twaver.servasoft.com/forum)
- MSN: [twavercn@hotmail.com](mailto:twavercn@hotmail.com)

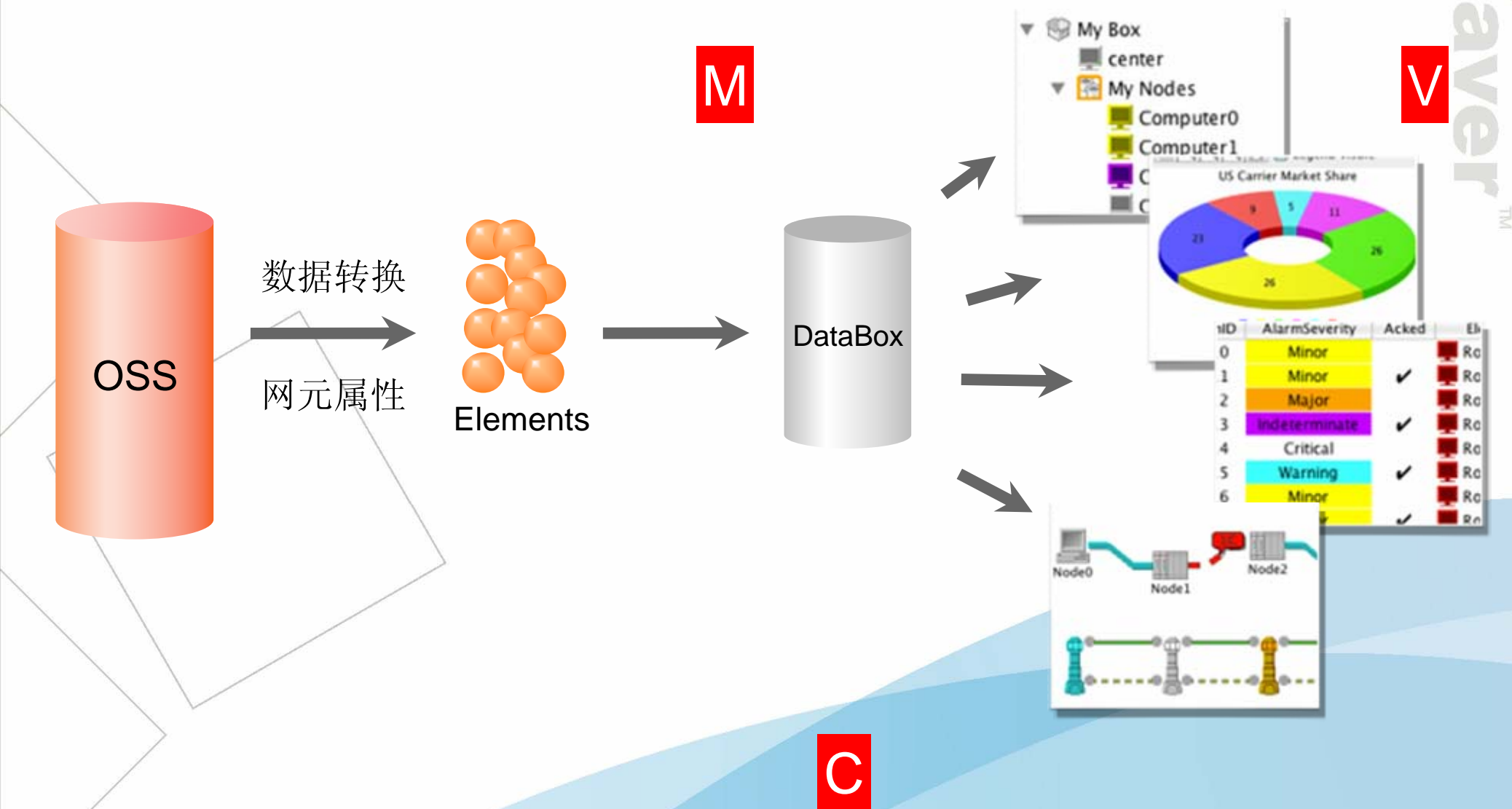
# TWaver Java核心组件使用

TWaver™

- 开发流程
- TDataBox的使用
- 预定义网元
- TNetwork的使用

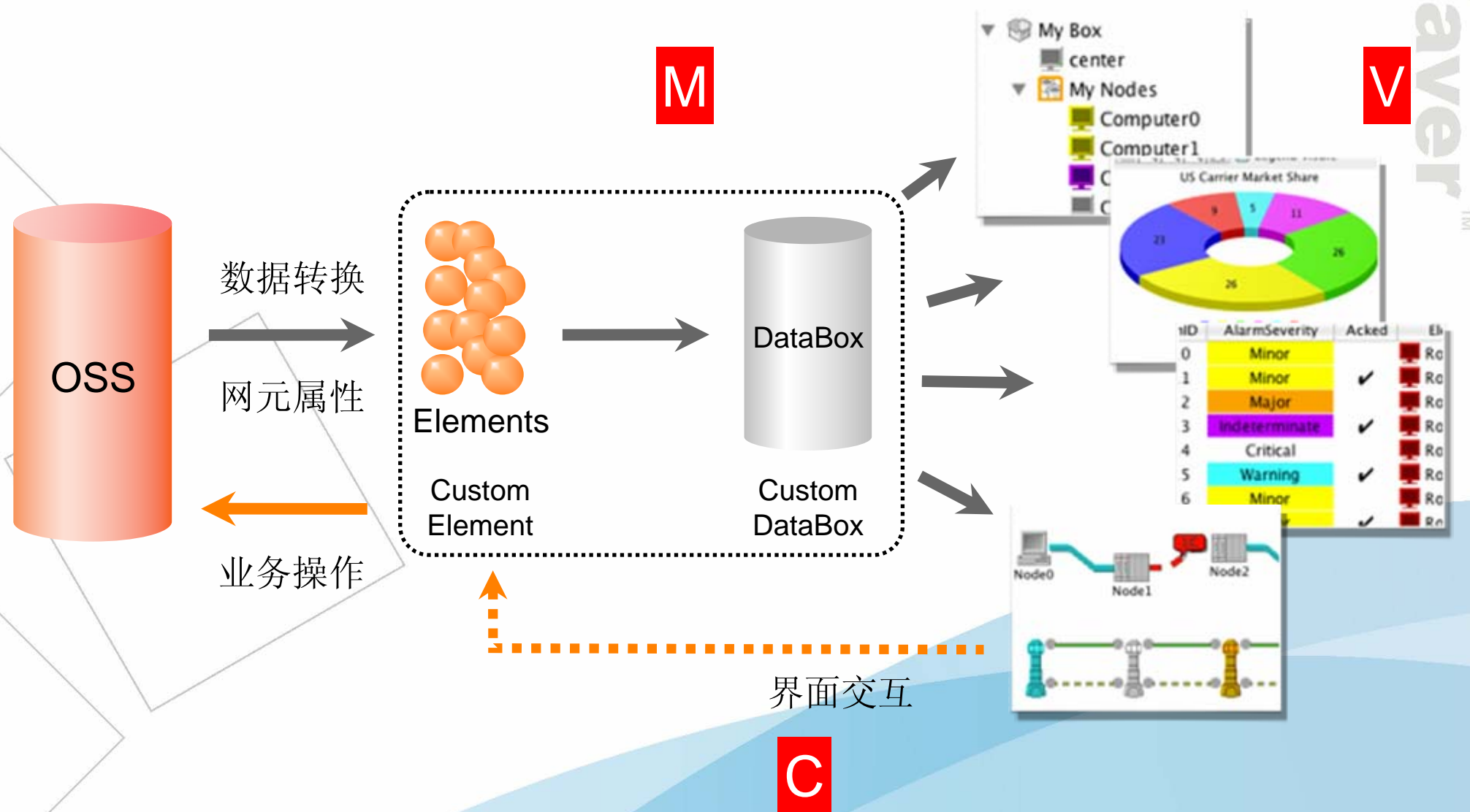


# TWaver开发流程





# TWaver开发流程



# TWaver开发流程示例

//数据采集

```
List<Device> devices = getDevicesFromOSS();  
List<Relationship> relationships = getDevicesRelationshipFromOSS();
```

//数据转换

```
TDataBox box = new TDataBox();  
translateToTWaverNode(box, devices, relationships);
```

//界面呈现

```
final TNetwork network = new TNetwork(box);  
network.doLayout(TWaverConst.LAYOUT_SYMMETRIC);  
showFrame("Develop Flow", network);
```

//添加交互

```
network.addElementDoubleClickedActionListener(new ActionListener() {  
    public void actionPerformed(ActionEvent e) {  
        Element element = (Element)e.getSource();  
        String newName = JOptionPane.showInputDialog(network, "change device name from : " + element.getName() + " to:");  
        if(newName != null){  
            element.setName(newName);  
            //入库业务操作  
        }  
    }  
});
```

```
private static List<Device> getDevicesFromOSS() {
    List<Device> devices = new ArrayList<Device>();
    devices.add(new Device("R1", "router"));
    devices.add(new Device("R2", "router"));
    devices.add(new Device("S1", "device"));
    devices.add(new Device("S2", "device"));
    return devices ;
}

private static List<Relationship> getDevicesRelationshipFromOSS(){
    List<Relationship> relationships = new ArrayList<Relationship>();
    relationships.add(new Relationship("2I", "R1", "R2"));
    relationships.add(new Relationship("3I", "R1", "S2"));
    return relationships;
}

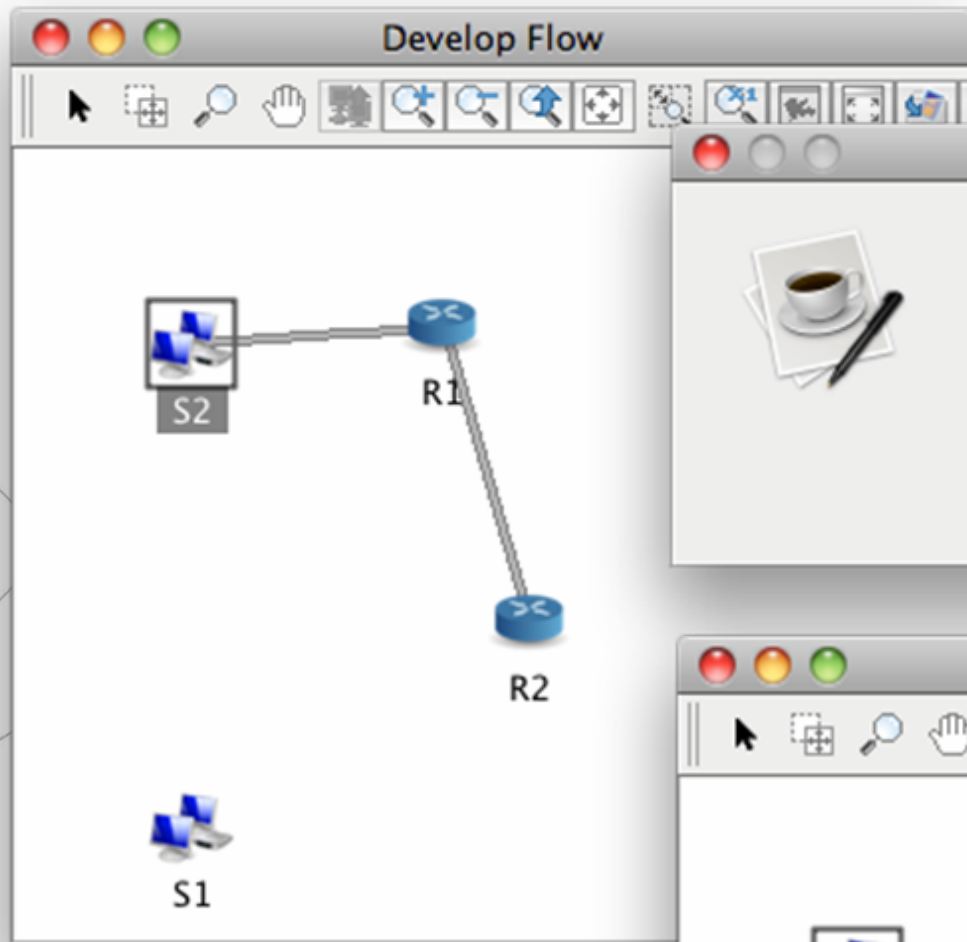
private static void translateToTWaverNode(TDataBox box, List<Device> devices, List<Relationship> relationships) {
    for(Device device : devices){
        Node node = new Node();
        node.setName(device.name);
        node.setImage("/ppt/images/" + device.type + ".png");
        box.addElement(node);
    }
    for(Relationship relationship : relationships){
        Link link = new Link((Node)box.getElementByName(relationship.fromName),
        (Node)box.getElementByName(relationship.toName));
        box.addElement(link);
    }
}
```

```
static class Device{
    String name;
    String type;

    Device(String name, String type){
        this.name = name;
        this.type = type;
    }
}

static class Relationship{
    String name;
    String fromName;
    String toName;

    Relationship(String name, String from, String to){
        this.name = name;
        this.fromName = from;
        this.toName = to;
    }
}
```

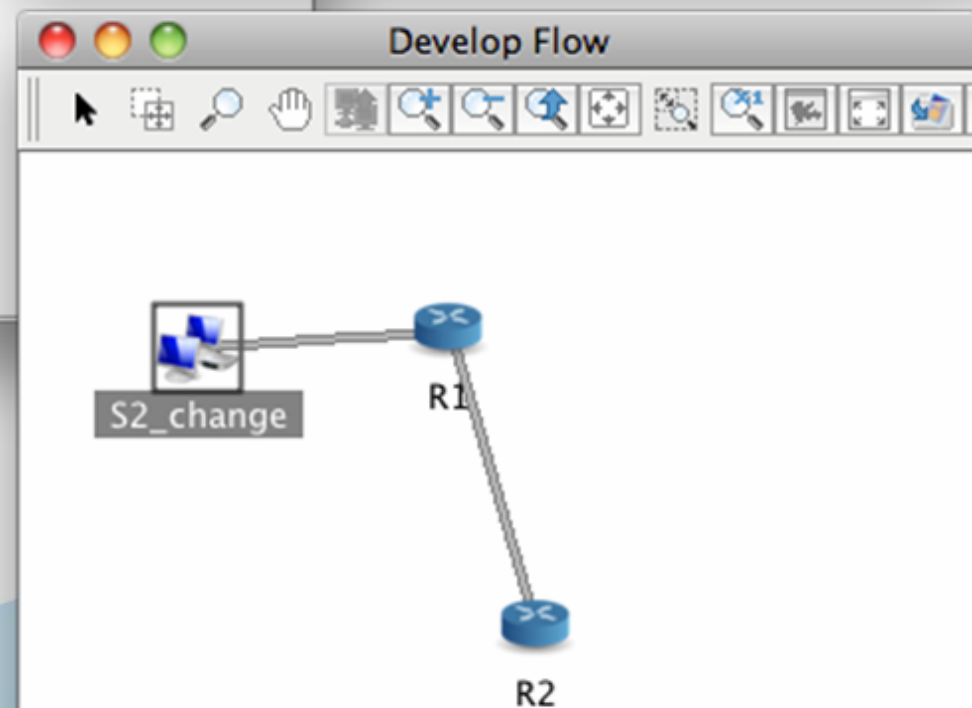


Input



change device name from : 'S2' to:

Cancel OK

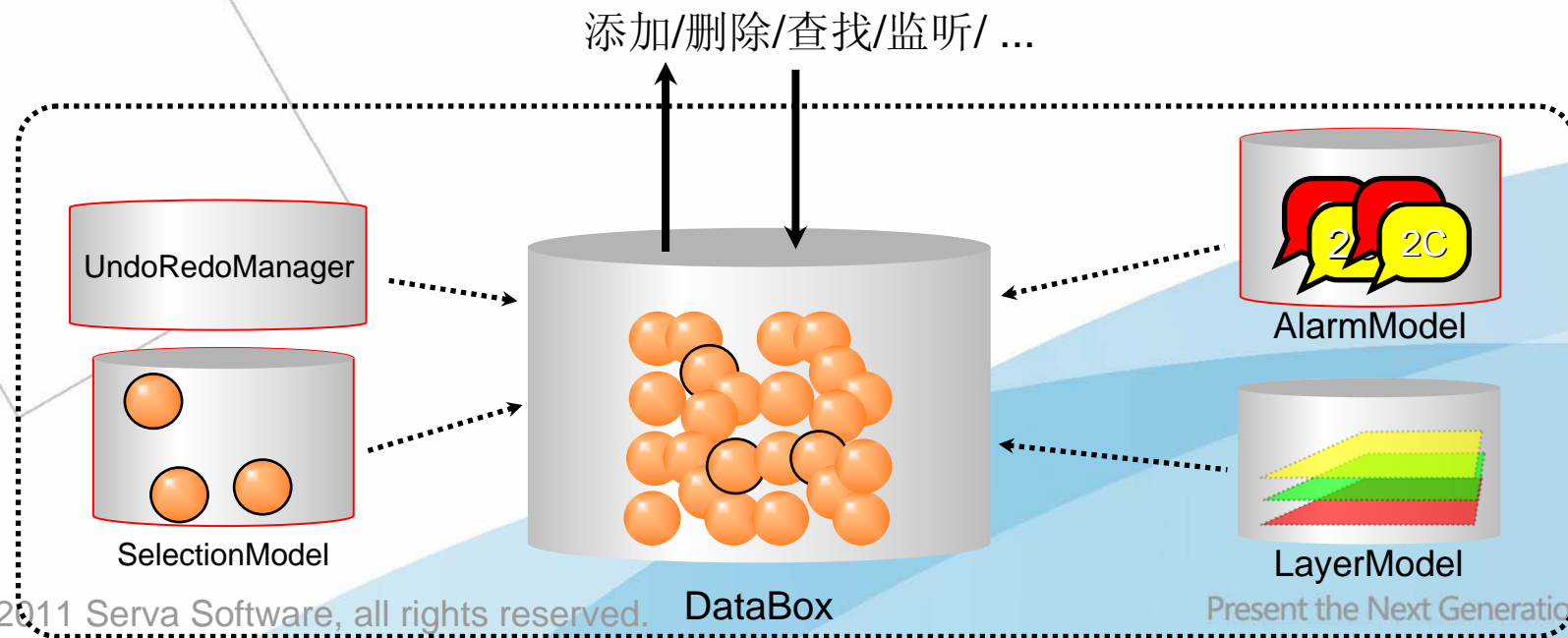




# TDataBox的使用

TDataBox: 用于管理网元的数据容器。

提供对网元添加, 删除, 更改层次, 遍历等操作;  
可以监听网元属性变化, 容器变化, 层次变化等事件;  
包含告警模型, 图层模型, 选中模型等数据容器



# 网元的增减清除

## ● 增加网元

`addElement(final Element element) / addElement(int index, final Element element) /`  
`addElement(Element element, Element parentOfRootElement)`

`addElementWithDescendant(Element element) /`  
`addElementWithDescendant(Element element, VisibleFilter filter)`

`addElements(Collection collection) / addElements(Collection collection, Element`  
`parentOfRootElement)`

## ● 删除网元

`removeElement / removeDescendant / removeElementByID /`  
`removeSelectedElements`

## ● 清除网元

`clear`

# 网元的访问

- 获取网元 `get***Element***`

`getAllElements()` / `getAllElementsReverse()` / `getRootElements()` /

`getRootElementsReverse()``getElementByID(Object id)`

`getElementByName(String name)`

- 遍历网元 `iterator***`

`iterator()` / `iterator(Class clazz)` / `iteratorReverse()` /

`iteratorSelection(ElementCallbackHandler handler)` /

`iteratorReverse(ElementCallbackHandler handler)`

`iteratorReverseByLayer(ElementCallbackHandler handler)` /

`iteratorByLayer(ElementCallbackHandler handler)`

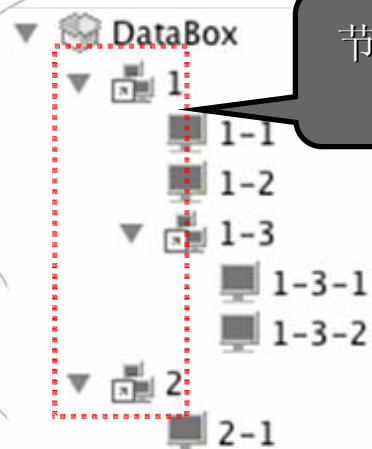
- 深度遍历，广度遍历

`depthFirstEnumeration()` / `depthFirstEnumeration(Element root)` /

`breadthFirstEnumeration()` / `breadthFirstEnumeration(Element root)`

# 网元的层次关系

- TDataBox中的网元通过父子关系形成层次结构
- move\*\*\*方法可以调整网元在当层的前后顺序



节点1, 2位于最根层  
(rootElements)

```
TDataBox
  ● moveTo(int, Element) : void
  ● moveToUp(Element) : void
  ● moveToDown(Element) : void
  ● moveToTop(Element) : void
  ● moveToBottom(Element) : void
  ● moveSelectionToUp() : void
  ● moveSelectionToDown() : void
  ● moveSelectionToTop() : void
  ● moveSelectionToBottom() : void
  ● moveElements(Iterator, MovableFilter, double, double)
```

# 网元的快速查找

快速查找器(DataBoxQuickFinder):

用于查找具有某个属性的网元，如找出所有名称为“PC”的网元: `List result =`

`box.createJavaBeanFinder("name").find("PC");`

创建快速查找器:

`TDataBox#create***Finder(String name)`

- `createJavaBeanFinder` — 通过javaBean属性查找
- `createClientPropertyFinder` — 通过ClientProperty查找
- `createUserPropertyFinder` — 通过userProperty查找



# 网元的快速查找示例

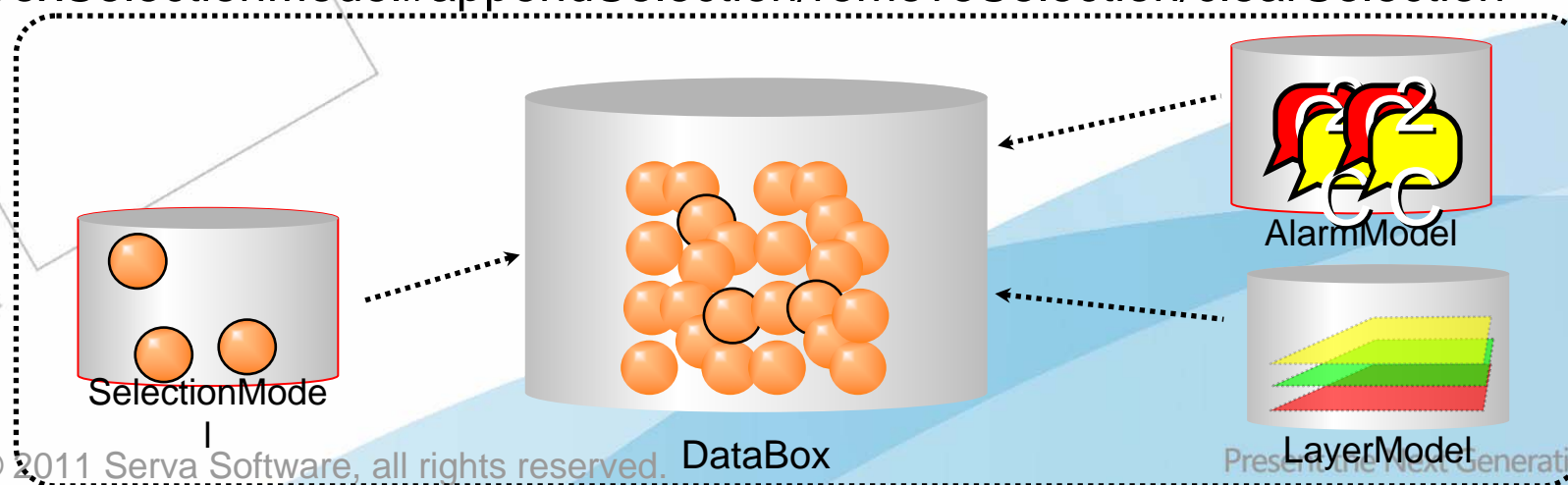
```
TDataBox box = new TDataBox();
for (int i = 0; i < 30; i++) {
    Node node = new Node();
    node.setName("node_" + i);
    node.putClientProperty("age", new Integer(i%10));
    box.addElement(node);
}
DataBoxQuickFinder finder = box.createClientPropertyFinder("age");
List elements = finder.find(new Integer(8));
for(Object node : elements){
    System.out.println(((Element)node).getName());
}
```

输出结果:  
node\_28  
node\_18  
node\_8

# TDataBox中的数据容器

TWaver™

- TDataBox中包含图层管理，告警管理，选中模型
- `dataBox.getLayerModel()`  
`LayerModel#addLayer/removeLayer/moveTo`
- `dataBox.getAlarmModel()`  
`AlarmModel#addAlarm/removeAlarm/clear`
- `dataBox.getSelectionModel()`  
`DataBoxSelectionModel#appendSelection/removeSelection/clearSelection`



# TDataBox中的监听器

监听器	作用
dataBoxListener	监听网元的添加删除，清除
propertyChangeListener	监听dataBox属性变化
elementPropertyChangeListener	监听网元属性变化
dataBoxSequenceListener	监听网元层次变化
batchListener	批处理监听

添加监听器：TDataBox#add\*\*\*Listener

删除监听：TDataBox#remove\*\*\*Listener

```
box.addElementPropertyChangeListener(new PropertyChangeListener() {  
    public void propertyChange(PropertyChangeEvent evt) {  
        System.out.println(((Element)evt.getSource()).getName() + "'s property changed");  
    }  
});
```

# 监听器使用示例

```
TDataBox box = new TDataBox();  
//添加元素属性变化监听  
box.addElementPropertyChangeListener(new PropertyChangeListener() {  
    public void propertyChange(PropertyChangeEvent evt) {  
        System.out.println(((Element)evt.getSource()).getName() + "'s property changed");  
    }  
});  
//添加数据容器变化监听  
box.addDataBoxListener(new DataBoxAdapter() {  
    public void elementRemoved(DataBoxEvent e) {  
        System.out.println("element removed: " + e.getElement().getName());  
    }  
    public void elementAdded(DataBoxEvent e) {  
        System.out.println("element added: " + e.getElement().getName());  
    }  
});  
Node node = new Node();  
node.setName("001");  
//添加元素到容器  
box.addElement(node);  
//修改元素属性  
node.setDisplayName("server-001");  
//从容器中删除元素  
box.removeElement(node);
```

输出结果：  
element added: 001  
001's property changed  
element removed: 001

# TDataBox的导入导出

- TDataBox中元素数据可以导出导入xml
- 导出的xml包含所有网元属性和box自身属性
- 网元图片也可以通过base64编码保存到xml中



# TDataBox的导入导出

## ●导出

**dataBox.output\*\*\*(...) / dataBox.toXML(...)**

```
public void output(String fileName,  
    boolean withElementId,  
    boolean withAlarmState,  
    ElementPersistentFilter elementFilter,  
    ClientPropertyPersistentFilter clientPropertyFilter)
```

## ●导入

**dataBox.parse\*\*\*(...)**

```
public void parse(String url, Element parentOfRootElement)
```

# TDataBox导入导出示例

```
TDataBox box = new TDataBox();
box.putClientProperty("databoxPropertyA", "a box
property");
```

```
Node node = new Node();
node.setName("a node");
node.setLocation(100, 100);
box.addElement(node);
```

//output xml

```
String xml=box.toXML();
System.out.println(xml);
```

//parse xml

```
TDataBox box2 = new TDataBox();
box2.parseXML(xml);
System.out.println(box2.getClientProperty("databoxPrope
rtyA"));
System.out.println(box2.getElementByName("a
node").getLocation());
```

```
<?xml version="1.0" encoding="UTF-8"?>
<java version="1.6.0_24" class="java.beans.XMLDecoder">
  <string>databoxPropertyA</string>
  <string>a box property</string>
  <object class="twaver.Node">
    <void property="location">
      <object class="java.awt.Point">
        <int>100</int>
        <int>100</int>
      </object>
    </void>
    <void property="name">
      <string>a node</string>
    </void>
  </object>
</java>
```

a box property

java.awt.Point[x=100,y=100]

Twaver™

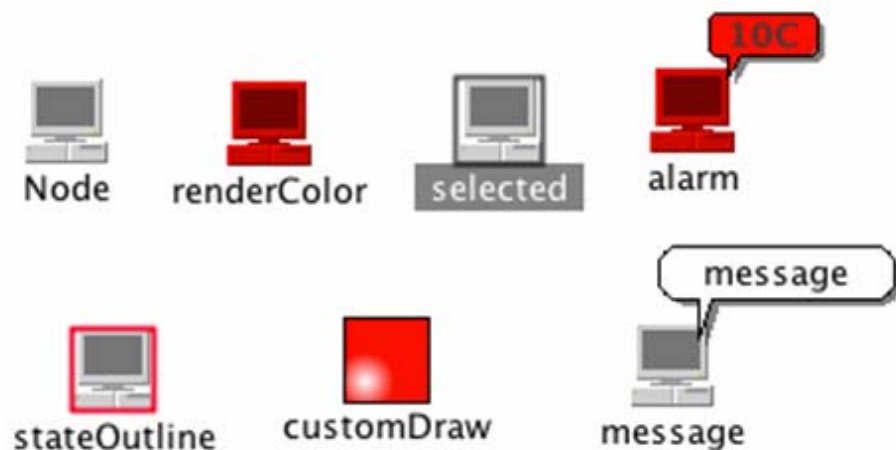
# 拓扑网元

## Element 拓扑元素

- |-- **Node** — 节点
  - |-- **Dummy** — 隐蔽节点
  - |-- **Group** — 分组
  - |-- **Text** — 文字
  - |-- **ShapeNode** — 多边形
    - |-- **ShapeImage/ShapeSubNetwork**
  - |-- **ResizableNode** — 可调整大小节点
  - |-- **Follower** — 跟随者
    - |-- **BaseEquipment** — 设备面板
      - |-- **BTS/BTSAntenna/Card/Port/Rack/Shelf/Slot/Grid**
    - |-- **SubNetwork** — 子网
- |-- **Link** — 连线
  - |-- **LinkSubNetwork** — 连线子网

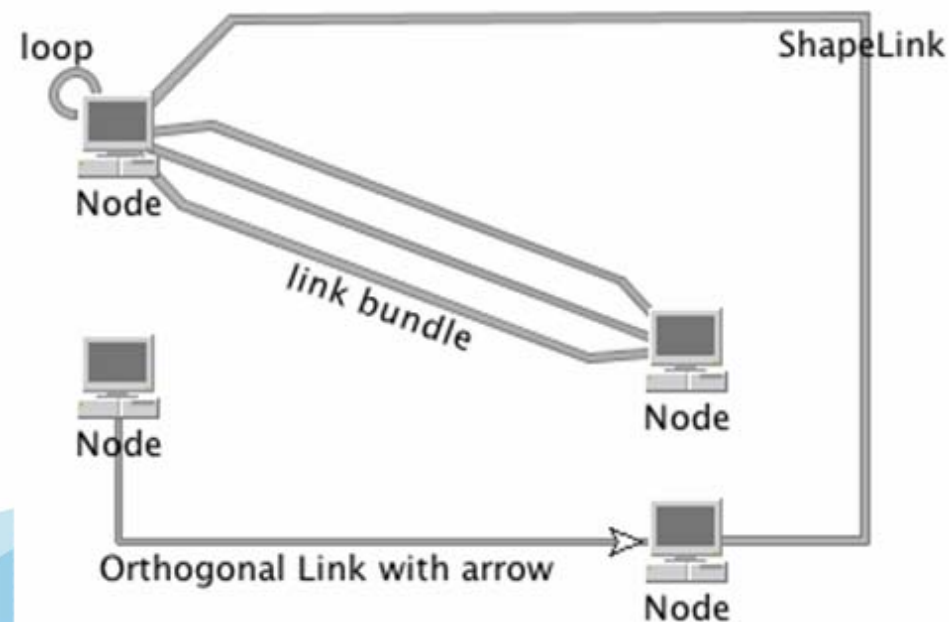
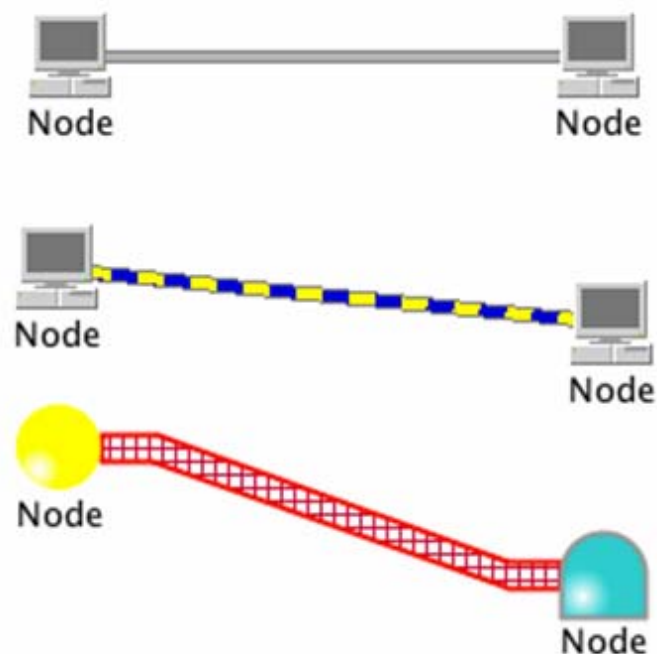
# Node

- 普通网元，可以设置图片，边框，颜色渲染...
- **twaver.Node**是其他主要网元类型的基类



# Link

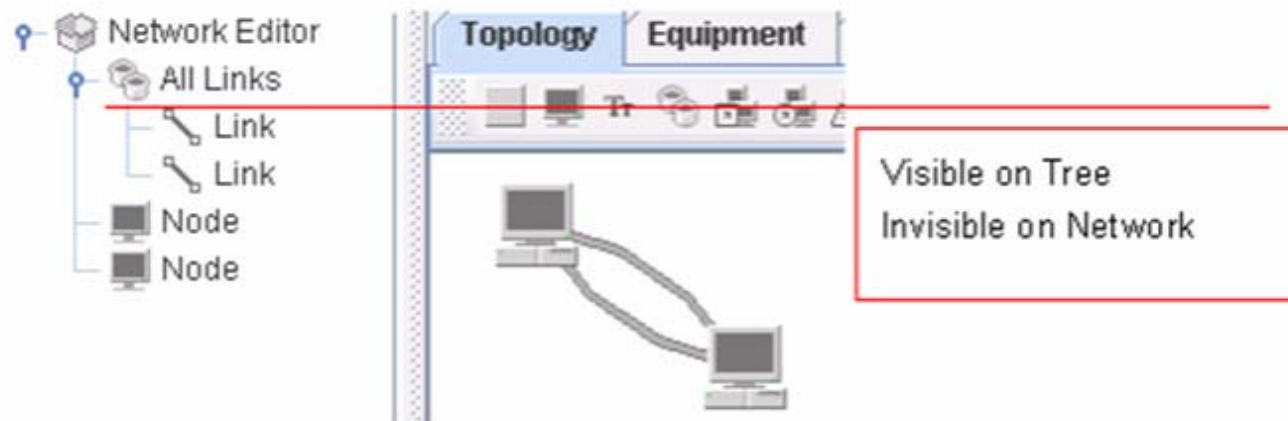
- 连线，连接节点的网元类型，可以设置多种样式，支持自环，绑定，箭头，流动效果等
- twaver.Link是其他连线类型的基类





# Dummy

- Dummy在Network中不可见，在树、表格以及属性表中都可见
- Dummy对象用来组织树结构，且不影响Network

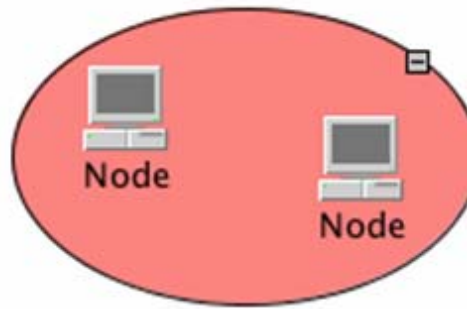


# Group

- 分组，分椭圆，矩形，平行四边形，八边形，3D等类型



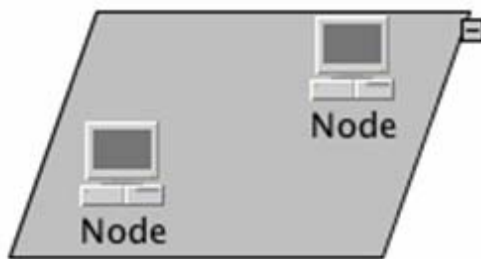
Group



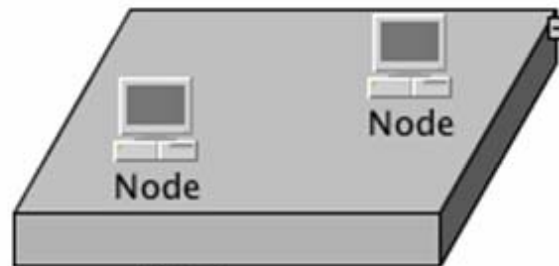
Ellipse Group



Octagon Group



Parallelogram Group



3D Group



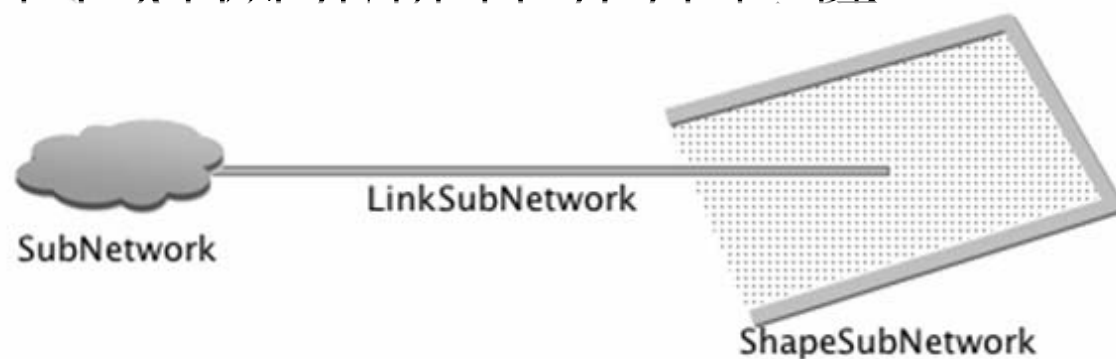
Round Rectangle Group

# SubNetwork

- 子网，通过双击操作，可以进入或退出子网

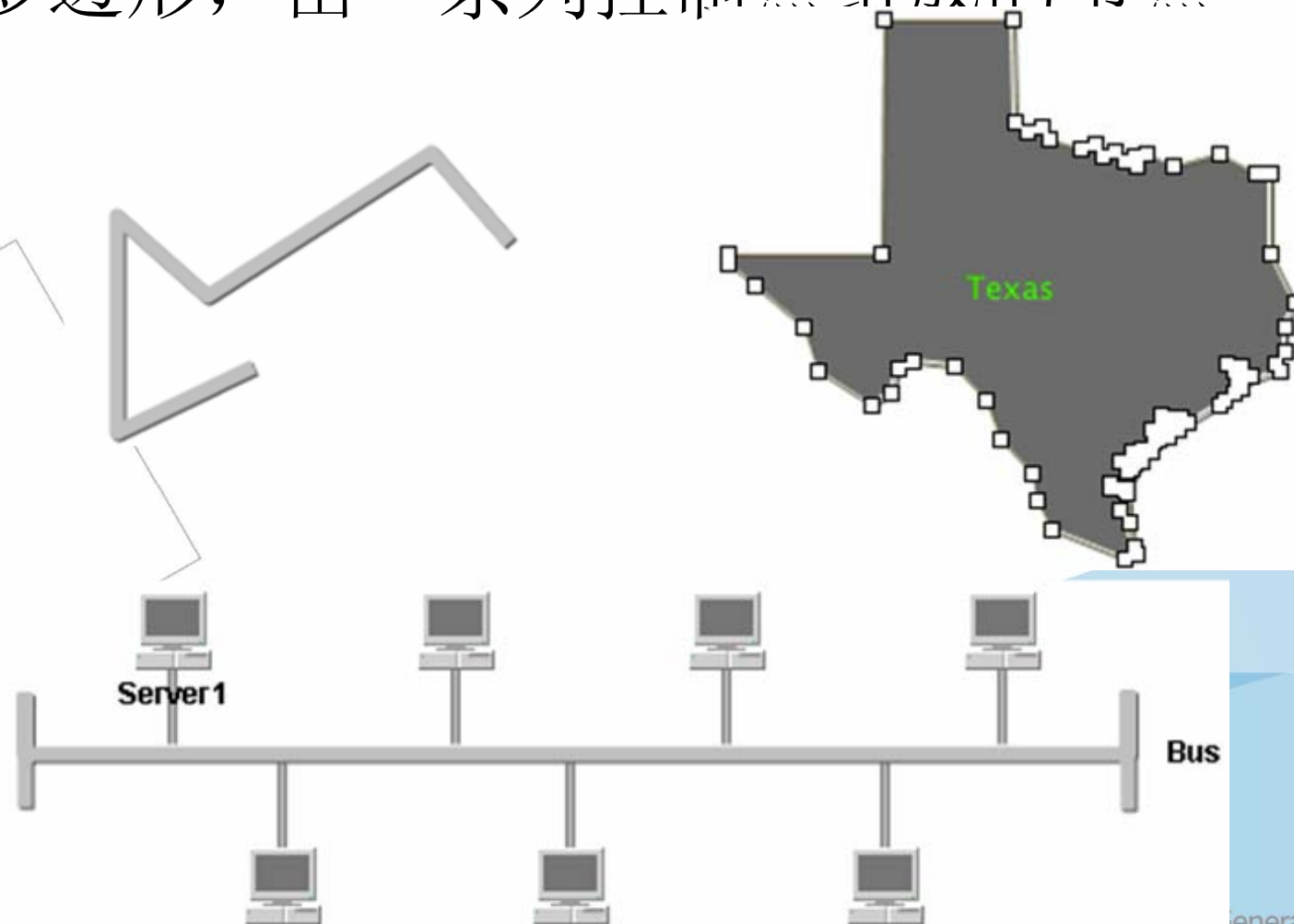
进入子网，则显示该子网中的网元，退出子网，则进入上一级子网，当前子网为null时，则表示为顶层

- 子网下可以设置自己的背景
- 可用于对网元进行分片管理



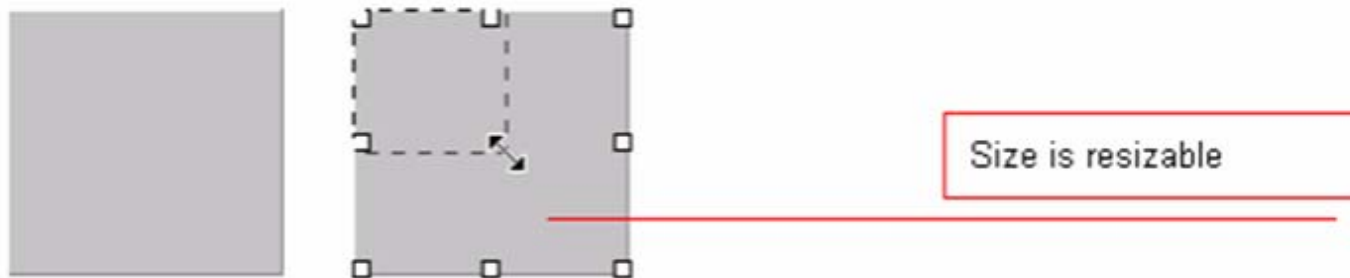
# ShapeNode

- 多边形，由一系列控制点组成的节点



# ResizableNode

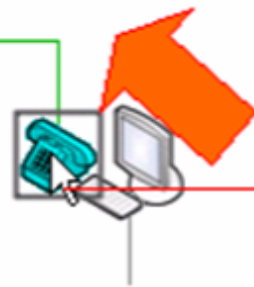
- 可调整大小节点





# Follower

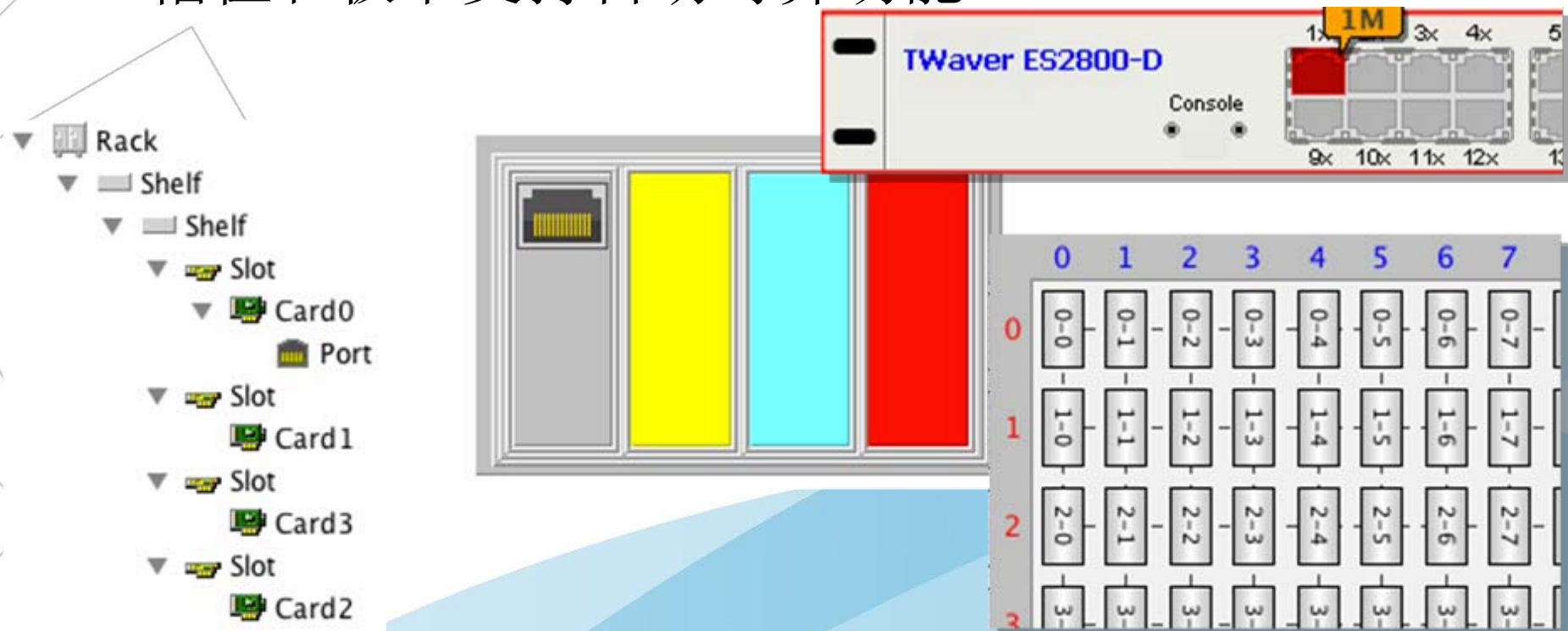
- 跟随者，可设置host节点，host移动，follower网元跟随移动



I move, and you move

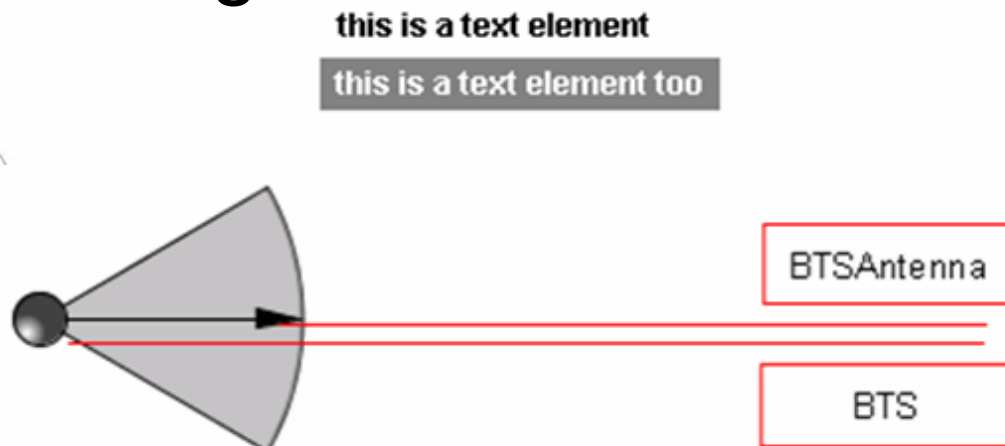
# Equipment

- 设备面板，继承于**Follower**类型，定义了机柜、机框、槽位、板卡、端口等网元类型
- 槽位和板卡支持自动对齐功能



# 其他类型

- 文字, twaver.Text
- BTS, BTSAntenna
- Shapelmage
- ...



# 网元属性

Element

JavaBeans - name, id ...

Client Properties - 图形相关属性: color, border

User Properties - 用户业务属性

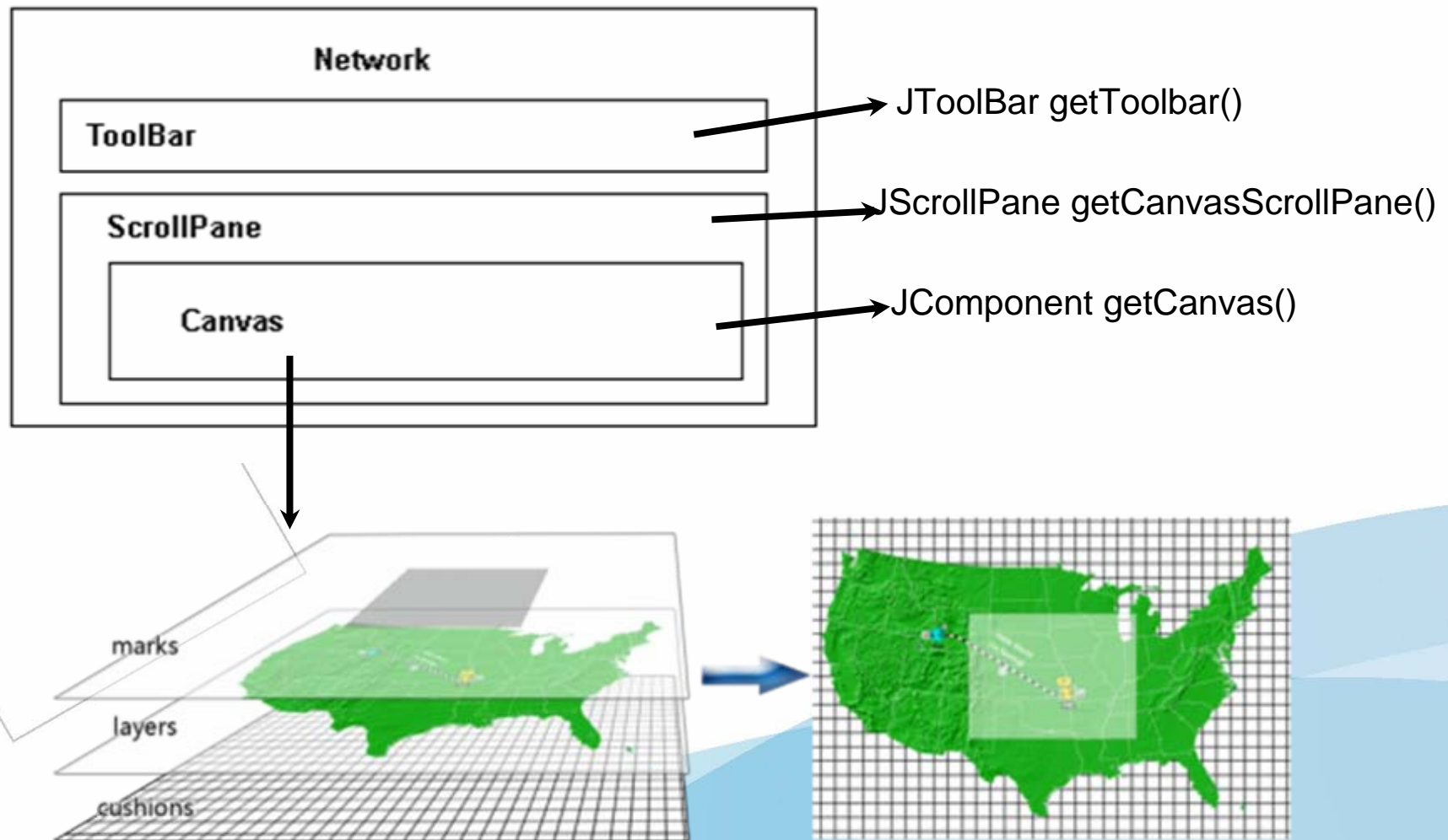
Business Object - 附属于网元的Java对象

# TNetwork的使用

- TNetwork的层次结构
- TNetwork的交互处理
- TNetwork的过滤器
- TNetwork的生成器
- ElementUI的定制



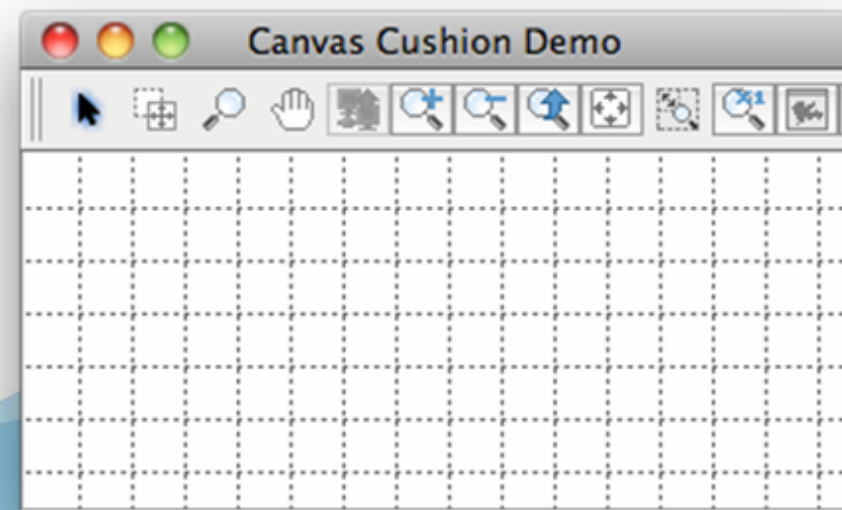
# TNetwork的层次结构



# Marker & Cushion

add/removeCanvasMarker(CanvasMarker  
m)add/removeCanvasCushion(CanvasCushion canvasCushion)

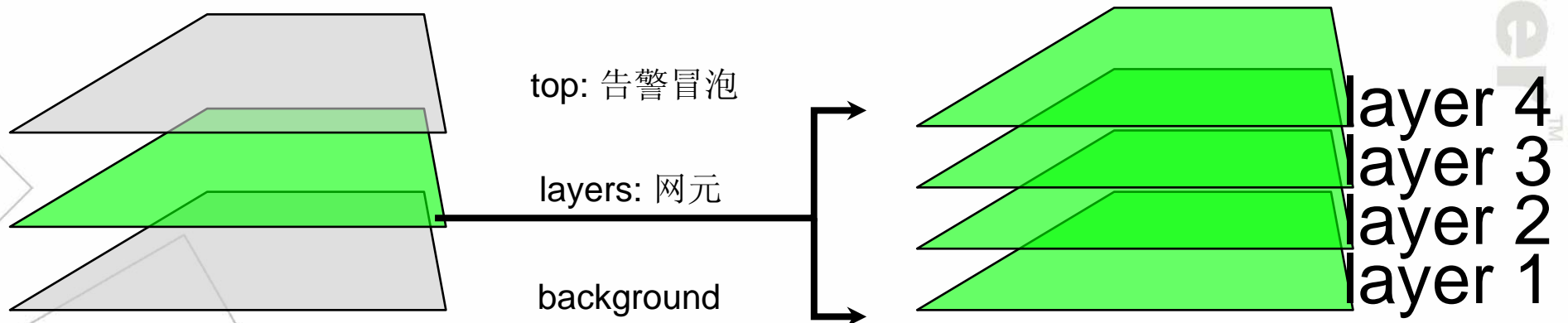
```
final TNetwork network = new TNetwork();
CanvasCushion canvasCushion = new CanvasCushion() {
    Color color = new Color(100, 100, 100);
    Stroke stroke = new BasicStroke(0, BasicStroke.CAP_BUTT, BasicStroke.JOIN_ROUND, 10.0f, new float[]{2}, 0.0f);
    int gridSize = 20;
    public void paint(Graphics2D g2d) {
        g2d.setColor(color);
        g2d.setStroke(stroke);
        int width = network.getCanvas().getWidth();
        int height = network.getCanvas().getHeight();
        for (int x = gridSize; x < width; x += gridSize) {
            g2d.drawLine(x, 1, x, height);
        }
        for (int y = gridSize; y < height; y += gridSize) {
            g2d.drawLine(0, y, width, y);
        }
    }
};
network.addCanvasCushion(canvasCushion);
```



使用Cushion绘制背景网格

# 图层管理

TWave™



```
LayerModel layerModel = box.getLayerModel();  
Layer bottomLayer = new Layer("bottom");  
layerModel.addLayer(0, bottomLayer);
```

```
ResizableNode bottomNode = new ResizableNode();  
bottomNode.setName("node at bottom");  
bottomNode.setLayerID(bottomLayer.getID());  
box.addElement(bottomNode);
```

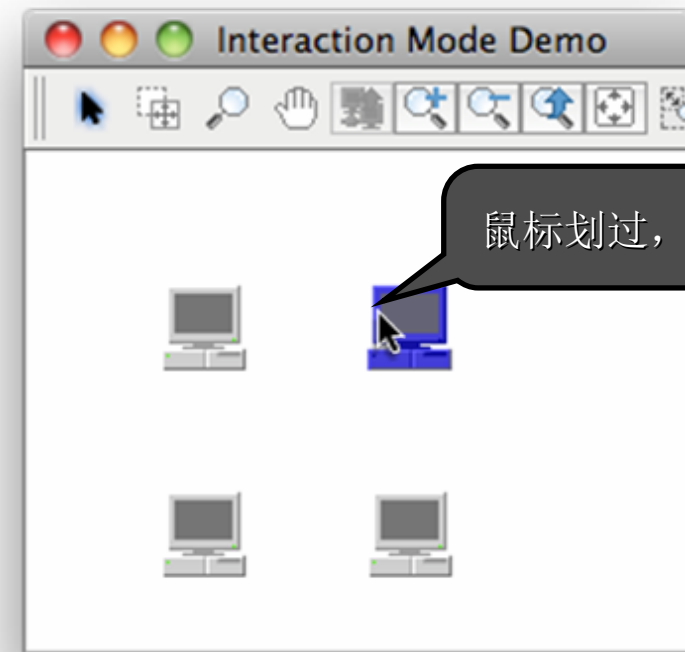
# TNetwork的交互

- **InputHandler** - 监听器可接收所有的鼠标键盘事件
- **InteractionMode** - 一组监听器组成一种交互模式
- **Network**预定义了多种交互模式，如默认模式，创建连线模式，编辑模式...

```
//添加
InputHandler[] listeners = new InputHandler[] {
    //添加选择处理器，让network可选
    new SelectionInputHandler(network),
    //在network上添加移动选择处理器.
    new EagerMoveInputHandler (network),
    //添加交互输入事件，用于激活network
    new InteractionInputHandler(network),
    //添加右击菜单的处理器，当用户右击network时可弹出右击菜单
    new PopupMenuInputHandler(network),
};
InteractionMode defaultMode = new InteractionMode(listeners);
network.setInteractionMode(defaultMode);
```

# 定制交互示例

```
public static void main(String[] args) {  
    TNetwork network=new TNetwork();  
    network.getDataBox().addElement(new Node());  
    network.getDataBox().addElement(new Node());  
    network.getDataBox().addElement(new Node());  
    network.getDataBox().addElement(new Node());  
    network.doLayout(TWaverConst.LAYOUT_SYMMETRIC);  
  
    InputHandler[] handlers = new InputHandler[] {  
        new DefaultInputHandler(network),  
        new ResizeInputHandler(network),  
        new SelectionInputHandler(network),  
        new EagerMoveInputHandler(network),  
        new InteractionInputHandler(network),  
        new PopUpMenuInputHandler(network),  
        new HighlightInputHandler(network)  
    };  
    network.setInteractionMode(new InteractionMode(handlers));  
    showFrame("Interaction Mode Demo", network);  
}
```





```
public static class HighlightInputHandler extends InputAdapter {
    private TNetwork network;
    private Element highlightElement;
    private Color highLightColor = new Color(100, 100, 255);
    private Color oldColor;

    public HighlightInputHandler(TNetwork network) {
        this.network = network;
    }

    public void mouseMoved(MouseEvent e) {
        Element element = network.getElementLogicalAt(e.getPoint());
        if(element != null){
            highlight(element);
        }else{
            reset();
        }
    }

    private void highlight(Element element) {
        if(element == null || element.equals(highlightElement)){
            return;
        }
        reset();
        highlightElement = element;
        oldColor = element.getRenderColor();
        element.putRenderColor(highLightColor);
    }

    private void reset() {
        if (highlightElement != null) {
            highlightElement.putRenderColor(oldColor);
            highlightElement = null;
        }
    }
}
```

# TNetwork过滤器

过滤器用于全局控制网元状态，例如控制网元的显示或隐藏，可移动还是不可移动

## add\*\*\*Filter/remove\*\*\*Filter

```
network.addVisibleFilter(new VisibleFilter() {  
    public boolean isVisible(Element element) {  
        return !(element instanceof Link);  
    }  
});
```

## set\*\*\*Filter

```
network.setMovableFilter(new MovableFilter(){  
    public boolean isMovable(Element element) {  
        if(editMode.isSelected()){  
            return true;  
        }  
        return !(element instanceof Equipment);  
    }  
});
```

# TNetwork过滤器种类

生成器类型	作用
visibleFilter	可见过滤，控制网元隐藏显示
movableFilter	移动过滤器，控制网元能否移动
selectableFilter	控制网元能否选择
resizableFilter	能否调整大小
sendToTopFilter	网元选中是否置顶显示
paintSelectionStateFilter	打印网元选中状态
elementPropertyChangeRepaintFilter	网元属性变化是否重绘
elementBoundsInvalidatableFilter	网元属性变化是否无效网元范围
doubleClickableFilter	能否双击
elementLabelEditableFilter	网元标签能否编辑

add  
remove

set  
get

# TNetwork生成器

## set\*\*\*Generator

生成器用于全局设置拓扑图中网元显示属性，而不用修改每个网元的属性

```
network.setElementLabelGenerator(new Generator() {  
    public Object generate(Object object) {  
        Element element = (Element) object;  
        return element.getClientProperty("STATE_NAME");  
    }  
});
```



# TNetwork生成器种类

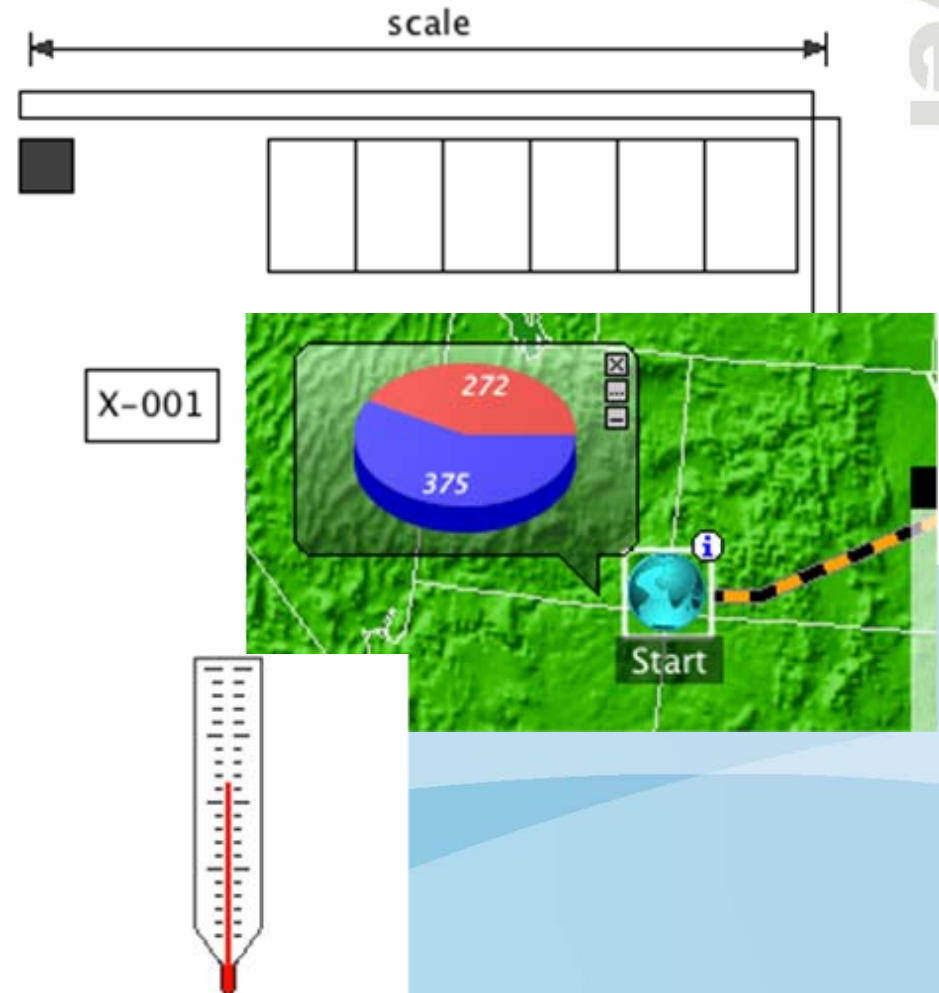
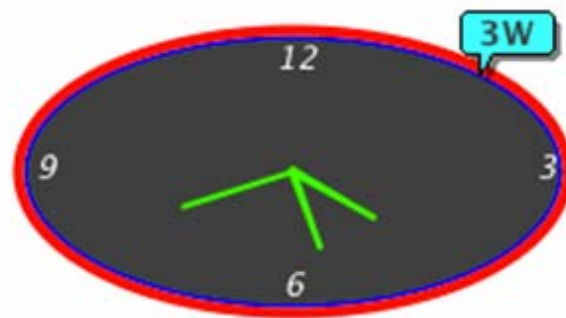
生成器类型	作用
ElementLabelGenerator	网元文本，默认显示displayName或name
AlarmLabelGenerator	告警冒泡标签，默认显示新发告警数量和最高级别
ElementToolTipTextGenerator	网元提示文本，默认是tooltipText或name
MessageContentGenerator	网元消息文本，默认是messageContent
ElementSelectColorGenerator	网元选中边框颜色，默认是borderColor
PopupMenuGenerator	右键菜单
AlarmColorGenerator	告警冒泡渲染色，默认是最高级别告警颜色
ElementOutlineGenerator	网元边框颜色，默认是outlineColor
ElementBodyColorGenerator	网元渲染色，默认是自身最高级别告警颜色或者renderColor



# ElementUI定制

定制UI有两种方式:

- 重绘ElementUI
- 挂载Attachment



# 重绘ElementUI

- 定义MyElementUI类，定制网元绘制

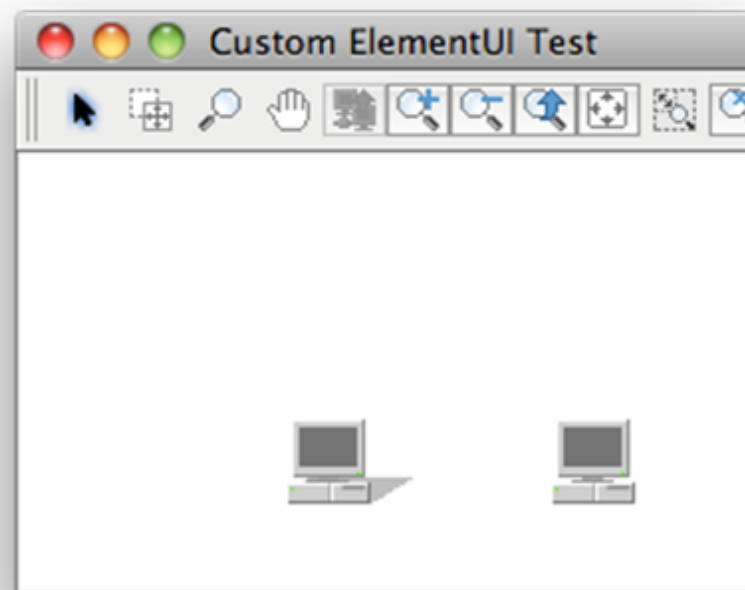
```
public class MyElementUI extends NodeUI {  
    ...  
    public void paintBody(Graphics2D g2d) {  
        //custom draw  
        super.paintBody(g2d);  
    }  
}
```

- 网元关联MyElementUI类，通过重写方法Element#public String getUIClassID()

```
public class MyElement extends Node {  
    ...  
    public String getUIClassID() {  
        return MyElementUI.class.getName();  
    }  
}
```

# 重绘ElementUI示例

- 下面的示例通过自定义网元的绘制方法，实现阴影的效果，并通过网元的“showShadow”属性来控制阴影是否显示



```
public static class MyElement extends Node {  
    //增加“showShadow”属性  
    public final static String PROPERTYNAME_SHOW_SHADOW = "propertyname.show.shadow";  
    public void putShowShadow(boolean show){  
        this.putClientProperty(PROPERTYNAME_SHOW_SHADOW, show);  
    }  
    public boolean isShowShadow(){  
        return Boolean.TRUE.equals(this.getClientProperty(PROPERTYNAME_SHOW_SHADOW));  
    }  
    //构造函数  
    public MyElement() {  
        super();  
    }  
    public MyElement(Object id){  
        super(id);  
    }  
    //关联MyElementUI  
    public String getUIClassID() {  
        return MyElementUI.class.getName();  
    }  
}
```

```
public static class MyElementUI extends NodeUI {  
    public MyElementUI(TNetwork network, Node element) {  
        super(network, element);  
        resetShadowShape();  
    }  
  
    public void paintBody(Graphics2D g2d) {  
        if (shadow != null) {  
            Color shadowColor = new Color(128, 128, 128, 128);  
            g2d.setColor(shadowColor);  
            g2d.fill(shadow);  
        }  
        super.paintBody(g2d);  
    }  
    private Polygon shadow = null;  
    private Polygon resetShadowShape() {  
        //...  
    }  
  
    public void elementPropertyChange(PropertyChangeEvent evt) {  
        super.elementPropertyChange(evt);  
        resetShadowShape();  
    }  
  
    public Rectangle getUIBounds() {  
        Rectangle result = super.getUIBounds();  
        if (shadow != null) {  
            result.add(shadow.getBounds());  
        }  
        return result;  
    }  
}
```



```

public static void main(String[] args) {
    TDataBox box = new TDataBox();
    MyElement node = new MyElement();
    node.putShowShadow(true);
    node.setLocation(100, 100);
    box.addElement(node);

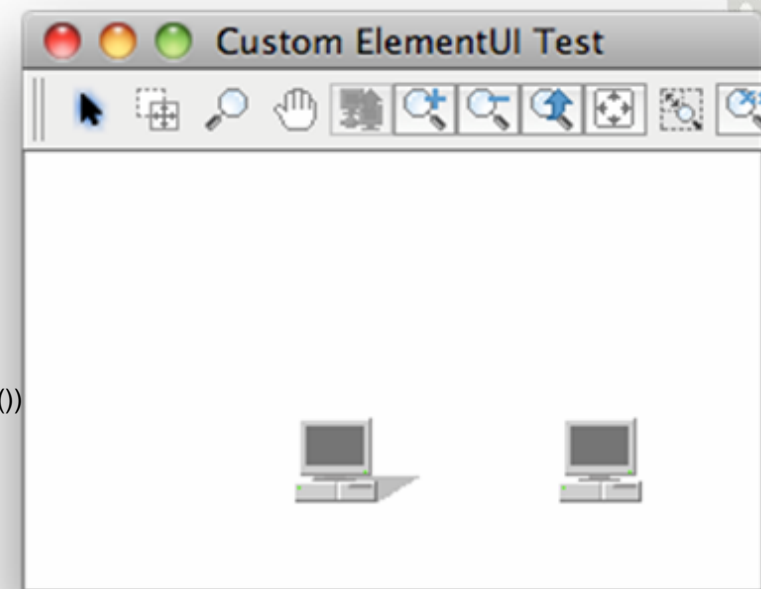
    node = new MyElement();
    node.setLocation(200, 100);
    box.addElement(node);

    PopupMenuGenerator popupMenuGenerator = new PopupMenuGenerator(){
        public JPopupMenu generate(TView tview, MouseEvent mouseEvent) {
            TNetwork network=(TNetwork)tview;

            final Element element=network.getElementLogicalAt(mouseEvent.getPoint())
            if(!(element instanceof MyElement)){
                return null;
            }

            final boolean showShadow = ((MyElement)element).isShowShadow();
            JPopupMenu menu=new JPopupMenu();
            JMenuItem menuItem=new JMenuItem(showShadow ? "Hide Shadow" : "Show Shadow");
            menuItem.addActionListener(new ActionListener() {
                public void actionPerformed(ActionEvent e) {
                    ((MyElement)element).putShowShadow(!showShadow);
                }
            });
            menu.add(menuItem);
            return menu;
        }
    };
    TNetwork network = new TNetwork(box);
    network.setPopupMenuGenerator(popupMenuGenerator);
    TestUtil.showFrame("Custom ElementUI Test", network);
}

```



# 定制Attachment

- twaver.network.ui.Attachment接口

实现类:

- LabelAttachment/AlarmAttachment/MessageAttachment/IconAttachment...
- ComponentAttachment



# 定制Attachment

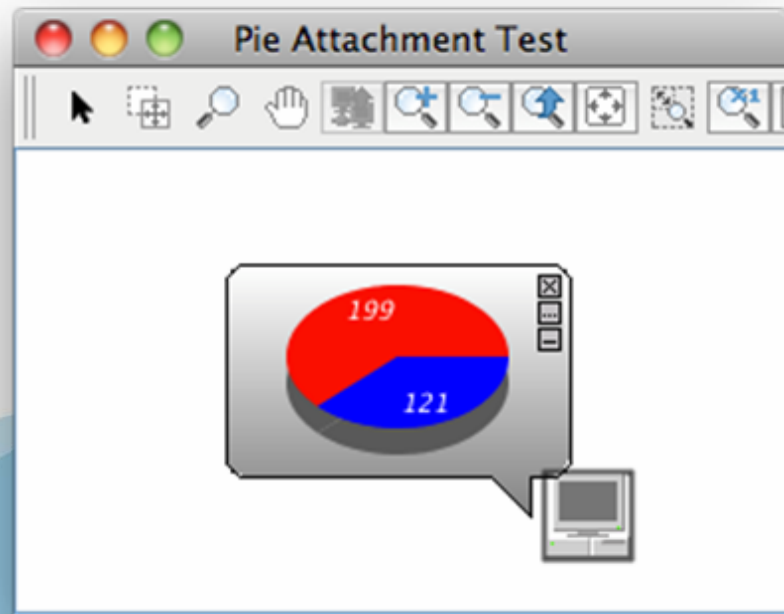
- 定义MyAttachment:  
public class MyAttachment extends  
ComponentAttachment
- 注册MyAttachment:  
TUIManager.registerAttachment("myAttachment",  
MyAttachment.class)
- 添加给网元:  
node.addAttachment("myAttachment")

# 使用Attachment

```
static {  
    TUIManager.registerAttachment("PieAttachment", PieAttachment.class);  
}  
  
public static void main(String[] args) {  
    TDataBox box = new TDataBox();  
    Node node = new Node();  
    node.setLocation(100, 100);  
    node.addAttachment("PieAttachment");  
    box.addElement(node);  
  
    TestUtil.showFrame("Pie Attachment Test", new TNetwork(box));  
}
```

注册附件

挂载附件



# 定义Attachment

定义附件类

```
public class PieAttachment extends ComponentAttachment {
```

```
private PieChart pie;
private Item xItem;
private Item yItem;
```

```
public PieAttachment(String name, ElementUI ui) {
```

```
super(name, ui);
```

```
xItem = new Item("x", element.getX(), Color.RED);
yItem = new Item("y", element.getY(), Color.BLUE);
Vector items = new Vector();
items.addElement(xItem);
items.addElement(yItem);
```

```
pie = new PieChart(items, null, Color.WHITE);
```

```
pie.setValueTextColor(Color.WHITE);
pie.setValueTextFont(TWaverUtil.getFont(Font.ITALIC, 10));
pie.getLegendPanel().setVisible(false);
pie.setBackgroundColor(Color.GRAY.darker());
pie.setOpaque(false);

this.setClosable(true);
this.setMinimizable(true);
this.setDraggable(true);
this.setStyle(TWaverConst.ATTACHMENT_STYLE_BUBBLE);
this.setPosition(TWaverConst.POSITION_LEFT);
this.setDirection(TWaverConst.ATTACHMENT_DIRECTION_TOP_LEFT);
this.setBorderVisible(true);
this.setBorderColor(Color.BLACK);
this.setBodyVisible(true);
this.setBodyGradient(true);
this.setBackgroundColor(Color(0, 0, 125));
this.setBodyGradientColor(new Color(255, 255, 125));
this.setWidth(100);
this.setHeight(80);
```

```
this.setComponent(pie);
```

```
public void elementPropertyChange(PropertyChangeEvent evt) {
super.elementPropertyChange(evt);
if (evt.getPropertyName().equals(TWaverConst.PROPERTYNAME_LOCATION)) {
```

```
xItem.setValue(element.getX());
yItem.setValue(element.getY());
```

```
}}
}
```

定义构造函数

创建一个Chart组件

设置这个Chart组件作为挂载主体

监听网元属性变化，实时更新Chart界面





- 论坛: [twaver.servasoft.com/forum](http://twaver.servasoft.com/forum)
- MSN: [twavercn@hotmail.com](mailto:twavercn@hotmail.com)

# 通用组件和告警的使用

## ●通用组件

- TTree的使用
- TElementTable的使用
- TTreeTable的使用
- TPropertySheet的使用
- Chart的使用

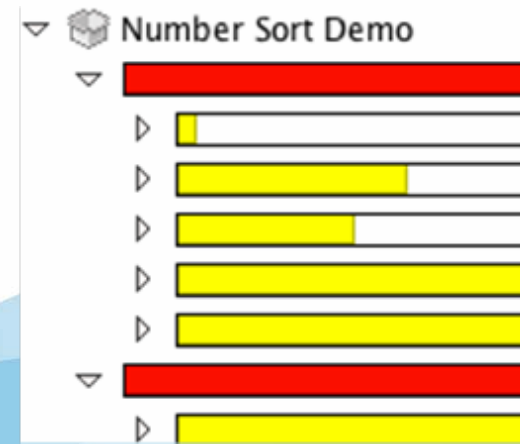
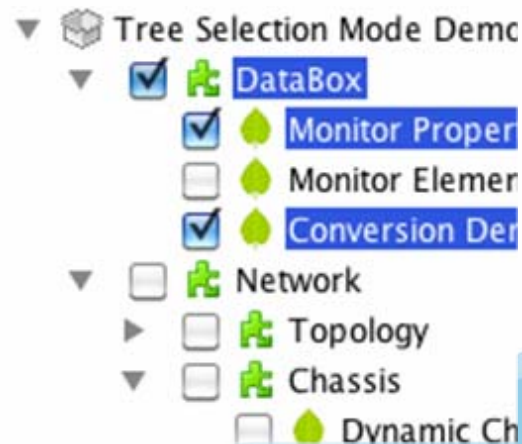
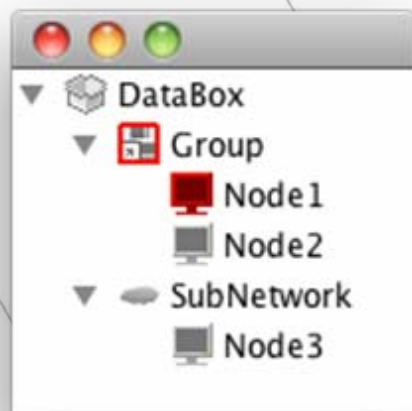
## ●告警的使用

- 告警对象 / 告警级别 / 告警容器 / 告警状态 / 告警统计 / 告警呈现

# TTree的使用

twaver.tree.TTree 用于展示TDataBox中元素父子层次关系的组件

```
TTree tree = new TTree(box);
```

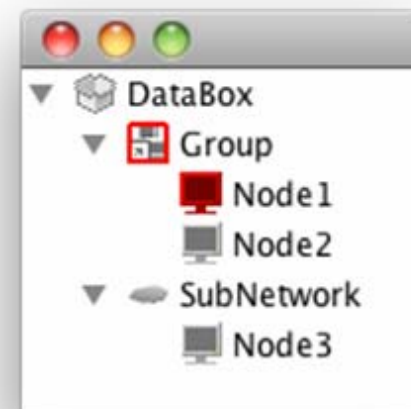


# TTree的创建

```
TDataBox box = new TDataBox();
TTree tree = new TTree(box);
Group group = new Group();
box.addElement(group);
SubNetwork subnetwork = new SubNetwork();
box.addElement(subnetwork);

Node node1 = new Node();
node1.setName("Node1");
node1.setParent(group);
node1.getAlarmState().addNewAlarm(AlarmSeverity.CRITICAL);
box.addElement(node1);
Node node2 = new Node();
node2.setName("Node2");
node2.setParent(group);
box.addElement(node2);
Node node3 = new Node();
node3.setName("Node3");
node3.setParent(subnetwork);
box.addElement(node3);

showFrame("Tree Demo", tree);
```



网元的父子关系决定树的层次关系

# 定制树节点

- 设置图标

`element.putElementTreeIcon(icon)`

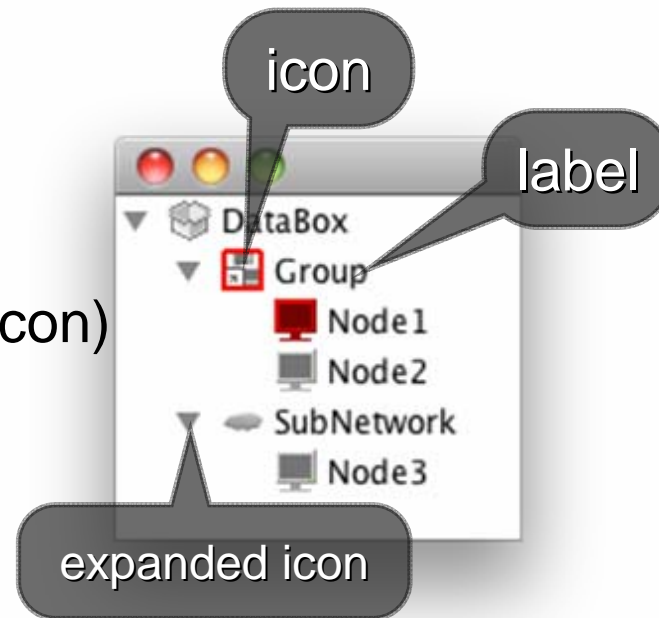
- 设置节点文本

`node.setName(name);`

- 设置展开合并图标

`tree.setCollapsedIcon(icon)`

`tree.setExpandedIcon(icon)`



```
tree.setCollapsedIcon(TWaverUtil.getIcon("/demo/sheet/nested/collapse.png")
);tree.setExpandedIcon(TWaverUtil.getIcon("/demo/sheet/nested/expand.png")
);
```



# 树节点层次与顺序

- 层次按网元父子关系  
`node.setParent(parent);`  
`node.addChild(child);`
- 同层节点顺序按加入到**TDataBox**的先后顺序  
可以通过调整网元在**dataBox**中的顺序来更改树节点的次序  
`box.moveTo***(node);`
- 可以设置比较器进行排序  
`tree.setSortComparator(comparator);`

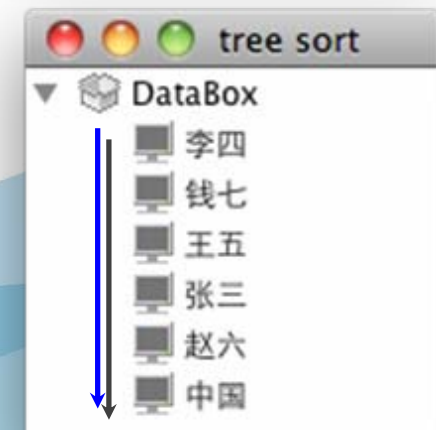
# 树节点中文排序示例

```
TDataBox box = new TDataBox();
createNode(box, "中国");
createNode(box, "张三");
createNode(box, "李四");
createNode(box, "王五");
createNode(box, "赵六");
createNode(box, "钱七");

final Comparator comparator = new Comparator() {
    Comparator cmp = Collator.getInstance(java.util.Locale.CHINA);
    public int compare(Object obj1, Object obj2) {
        Element node1 = (Element) obj1;
        Element node2 = (Element) obj2;
        return cmp.compare(node1.getName(), node2.getName());
    }
};

TTree tree = new TTree(box);
tree.setSortComparator(comparator);

showFrame("tree sort", tree);
showFrame("tree no sort", new TTree(box));
```

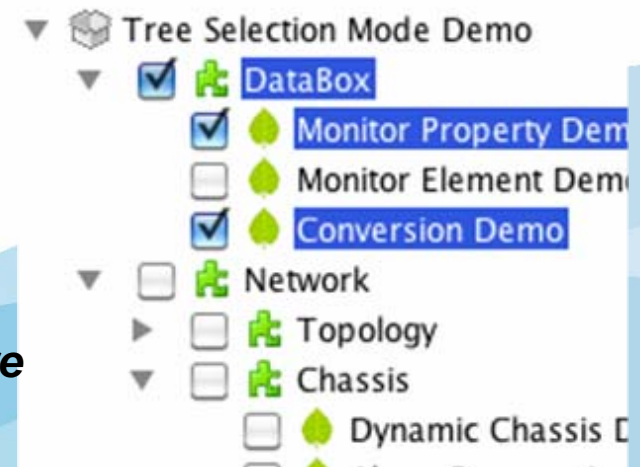


# TTree的框选模式

- TTree提供框选模式，可通过下面的方法切换  
*TTree.setTTreeSelectionMode(int mode)*

TTree有五种选择模式，详见Demo:  
*TreeSelectionModeDemo.java*

```
tree.setTTreeSelectionMode(TTree.CHECK_SELECTION);tree.  
e.setPaintSelectedStateWhenChecked(true);
```



# TElementTable的使用

- 与TDataBox绑定，显示网元信息的表格，每行对应一个网元
- 网元属性变化，同步更新，可编辑
- 可以通过API或XML配置表格列
- 可以对表格进行排序，支持多列排序
- TElementTable继承与JTable，支持JTable的renderer和editor设置

# TElementTable的配置

- TElementTable配置包括指定显示网元类型，设置网元显示哪些属性：

- 设置网元类型：

```
public void setElementClass(Class elementClass)
```

- 注册网元显示列：

```
public void registerElementClassXML(Class elementClass, String url)
```

```
public void registerElementClassXML(Class elementClass, InputStream inputStream)
```

```
public void registerElementClassAttributes(Class elementClass, List attributes)
```

```
table.registerElementClassXML(Person.class, "/resource/bean/table.xml");
```

```
table.setElementClass(Person.class);
```



# TElementTable的配置示例

TWaver™

```
TDataBox box = new TDataBox();
TElementTable table = new TElementTable(box);
table.setAutoResizeMode(JTable.AUTO_RESIZE_ALL_COLUMNS);
```

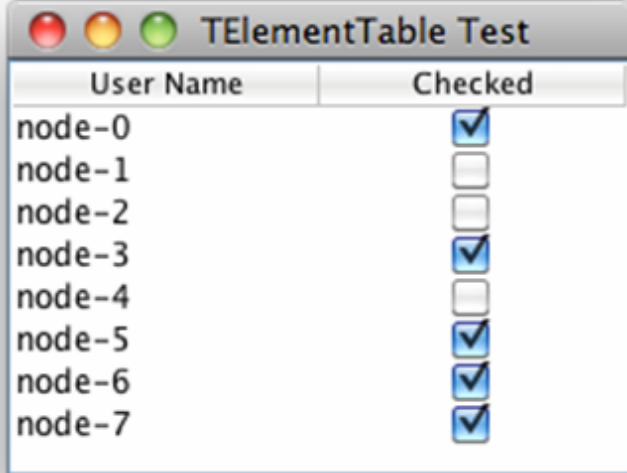
```
List attributes = new ArrayList();
ElementAttribute attribute = new ElementAttribute();
attribute.setUserPropertyKey("userName");
attribute.setDisplayName("User Name");
attributes.add(attribute);
```

```
attribute = new ElementAttribute();
attribute.setUserPropertyKey("checked");
attribute.setDisplayName("Checked");
attributes.add(attribute);
```

```
table.registerElementClassAttributes(Node.class, attributes);
table.setElementClass(Node.class);
```

```
for (int i = 0; i < 8; i++) {
    Node element = new Node();
    element.putUserProperty("userName", "node-" + i);
    element.putUserProperty("checked", TWaverUtil.getRandomBool());
    box.addElement(element);
}
```

```
TestUtil.showFrame("TElementTable Test", new JScrollPane(table));
```



User Name	Checked
node-0	<input checked="" type="checkbox"/>
node-1	<input type="checkbox"/>
node-2	<input type="checkbox"/>
node-3	<input checked="" type="checkbox"/>
node-4	<input type="checkbox"/>
node-5	<input checked="" type="checkbox"/>
node-6	<input checked="" type="checkbox"/>
node-7	<input checked="" type="checkbox"/>

# 使用XML配置表格

使用XML配置文件可以实现相同的功能

```
<beaninfo>  
  <attribute  
    userPropertyKey="userName"  
    displayName="User Name"/>  
  <attribute  
    userPropertyKey="checked"  
    displayName="Checked"/>  
</beaninfo>
```

```
table.registerElementClassXML(Node.class, "/ppt/table/table.xml");  
table.setElementClass(Node.class);
```

# TElementTable数据顺序

TWaver™

- TElementTable中数据的默认顺序

- 默认初始顺序与网元加入到dataBox中的先后顺序一致，先加入的在上，后加入的在下
- 可以设置反序排列，调用：  
*table.setConverseIncreaseOrder(true)*

- 表格的另一种顺序，按层次顺序：

*table.setIteratorByHierarchy(true)*

这种顺序下可以通过dataBox.move\*\*(element)函数调整行序

# 表格排序

通过TTableModel获取列信息:

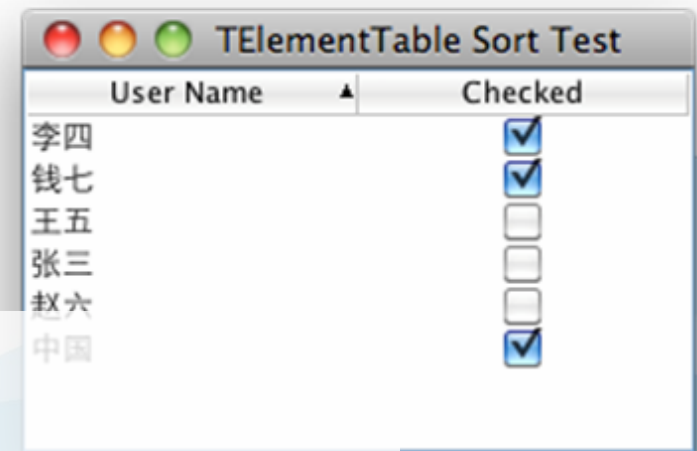
TTableModel#public TTableColumn getColumnByName(String name)

通过TTableModel设置列排序:

TTableModel#public void sortColumn(TTableColumn sortedColumn, int sortMode, boolean multiColumnSort)

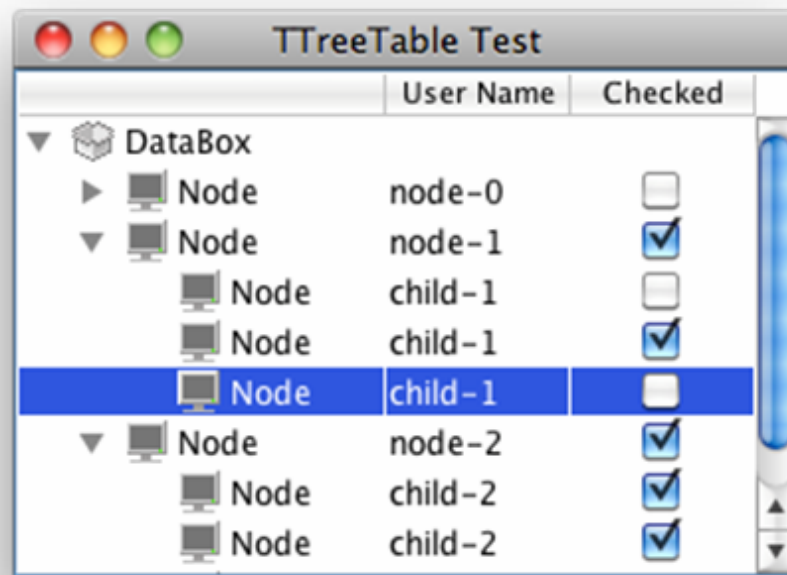
示例, 对userName列定制比较器, 并排序

```
TTableModel tableModel = table.getTableModel();  
TTableColumn column = tableModel.getColumnByName("userName");  
Collator comparator = Collator.getInstance(java.util.Locale.CHINA);  
column.setSortComparator(comparator);  
tableModel.sortColumn("userName", TTableModel.SORT_ASCENDING);
```



# TTreeTable的使用

- TTreeTable继承于TElementTable，增加了一个树列，支持树的展开合并操作，其配置方式与TElementTable相同





# TTreeTable中的Tree

- 获取树：
- `TTree tree = treeTable.getTree();`
- 树表的展开合并、过滤都通过**tree**来操作
- *`table.getTree().expandAll();`*//全部展开
- 获取树列：

`TTableColumn column = treeTable.getTreeColumn();`

# TTreeTable的数据顺序

- 数据显示顺序与树的顺序相同
- 数据排序按树排序使用

```
treeTable.setTreeColumnComparator(new Comparator() {  
    Comparator cmp = Collator.getInstance();  
    public int compare(Object obj1, Object obj2) {  
        Element node1 = (Element) obj1;  
        Element node2 = (Element) obj2;  
        return cmp.compare(node1.getName(), node2.getName());  
    }  
});
```

# TPropertySheet的使用

- 属性页组件用于显示选中网元属性信息
- 属性页配置方式与TElementTable相同，两者可以共用配置文件
- 支持属性的分组显示

# 属性页示例

属性表的配置方式与TElementTable相同

```
TDataBox box = new TDataBox();
TPropertySheet sheet = new TPropertySheet(box);

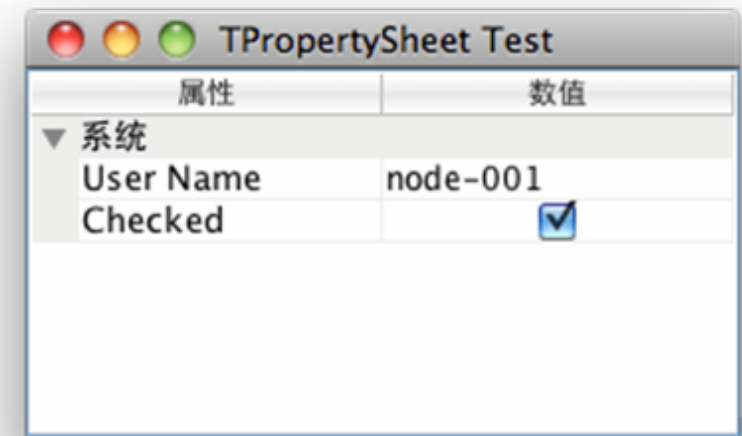
List attributes = new ArrayList();
ElementAttribute attribute = new ElementAttribute();
attribute.setUserPropertyKey("userName");
attribute.setDisplayName("User Name");
attributes.add(attribute);

attribute = new ElementAttribute();
attribute.setUserPropertyKey("checked");
attribute.setDisplayName("Checked");
attributes.add(attribute);

sheet.registerElementClassAttributes(Node.class, attributes);

Node element = new Node();
element.putUserProperty("userName", "node-001");
element.putUserProperty("checked", TWaverUtil.getRandomBool());
box.addElement(element);

box.getSelectionModel().setSelection(element);
showFrame("TPropertySheet Test", new JScrollPane(sheet));
```



# 属性分组显示

- 属性表支持属性的分组显示，默认在“system”分组
- 定制分组两步实现，首先注册分组，然后将分组信息设置给属性

```
Category.registerCategory("user", "User Info", true);
```

```
Category.registerCategory("basic", "Basic", true);
```

```
List categoryNames = new ArrayList();
```

```
categoryNames.add(0, "user");
```

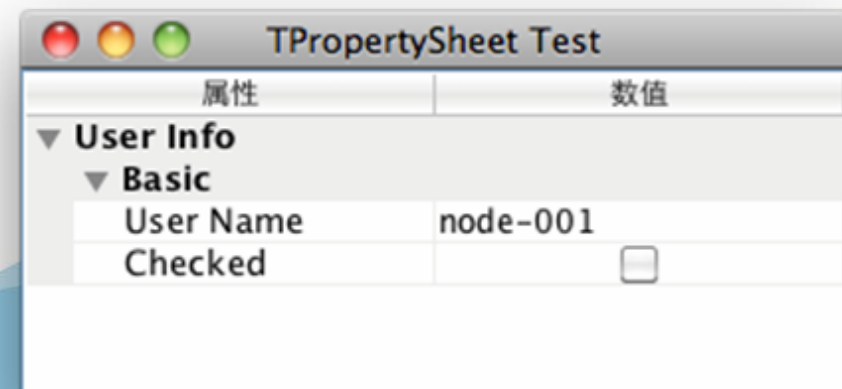
```
categoryNames.add(1, "basic");
```

```
ElementAttribute attribute = new ElementAttribute();
```

```
attribute.setCategoryNames(categoryNames);
```

```
attribute.setUserPropertyKey("userName");
```

```
attribute.setDisplayName("User Name");
```

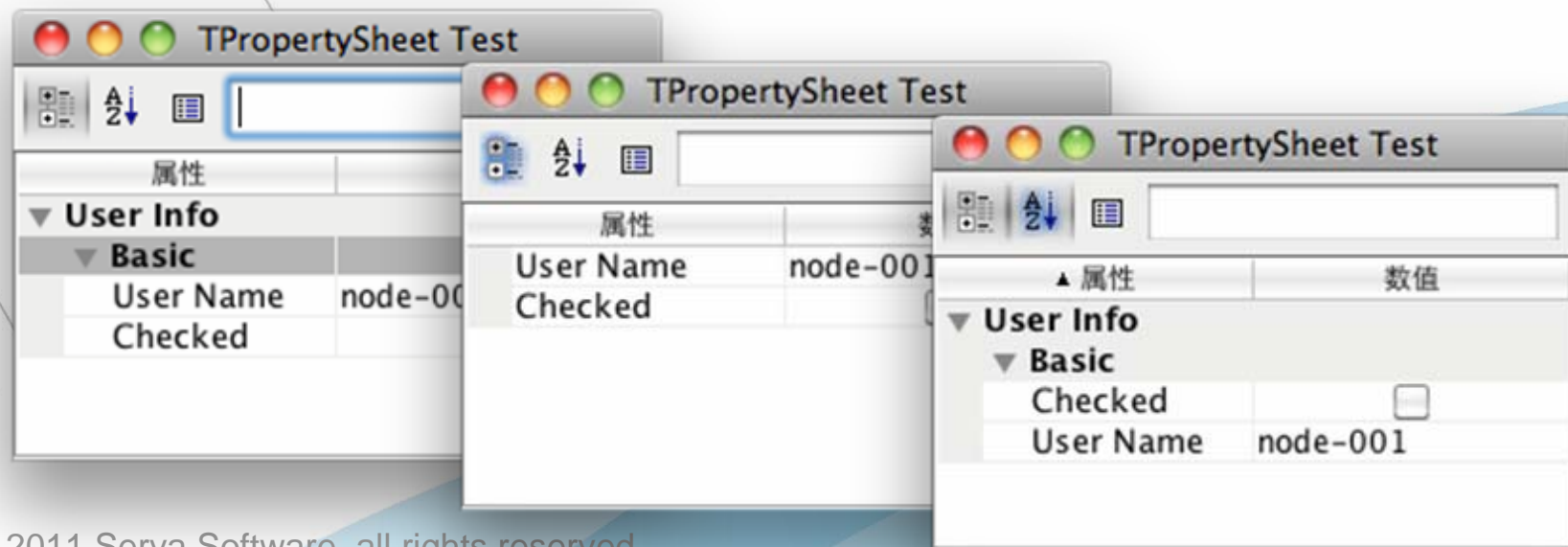




# 属性页面板组件

- 属性页面板（**TPropertySheetPane**）包含工具栏，提供属性页相关的快捷操作，如：切换属性表显示样式，排序，快速查找...

```
TPropertySheetPane sheetPane = new TPropertySheetPane(sheet);  
showFrame("TPropertySheetPane Test", sheetPane);
```



# 过滤器，排序 ...

Twaver™

## ●过滤器

```
public void setElementAttributeVisibleFilter(ElementAttributeVisibleFilter elementAttributeVisibleFilter)
```

## ●排序

### - 属性排序

```
public void setPropertySortingComparator(Comparator propertySortingComparator)
```

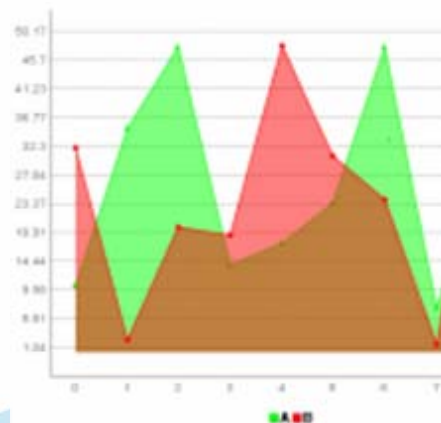
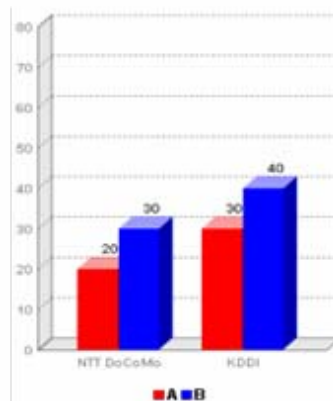
### - 分组排序

```
public void setCategorySortingComparator(Comparator categorySortingComparator)
```

# Chart的使用

- Chart组件以图表的形式展示TDataBox中的数据。

- BarChart
- BubbleChart
- DialChart
- LineChart
- PercentChart
- PieChart
- RadarChart



# Chart Value & List Values

TWaver™

一个网元可以设置一个或一组chart 值，通过下面的方法：

twaver.Element#

//设置单个chart值

```
public void putChartValue(double value);
```

//适用于bubble chart

```
public void addChartBubble(Bubble bubble);
```

//设置一组chart值

```
public void addChartValue(double value);
```

图表根据这些数值来绘制，TWaver chart中称单个图表值为**Chart Value**，称一组图表值为**Chart List Values**，这两种数据完全独立，一个图表可根据自己的类型，选择使用**Chart Value**还是**Chart List Values**做数据源。

例如饼图使用**Chart Value**，线图使用**Chart List Values**，而有的Chart组件对这两种数据源都支持，比如柱状图

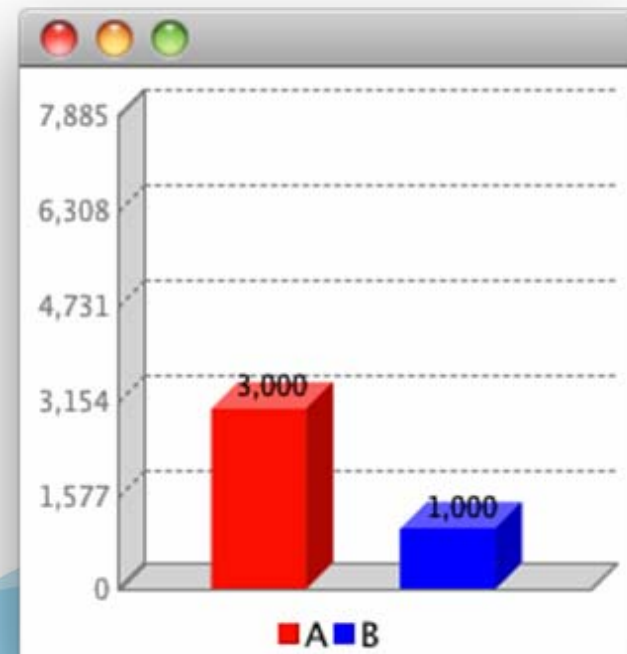
# BarChart示例

```
TDataBox box = new TDataBox();
```

```
BarChart barChart = new BarChart(box);  
barChart.setYScaleTextVisible(true);  
barChart.setYScaleMinTextVisible(true);  
barChart.setUpperLimit(7885);  
barChart.setLowerLimit(0);  
barChart.setYScaleValueGap(1577);
```

```
Element A = new Node("A");  
A.setName("A");  
A.putChartColor(Color.RED);  
A.putChartValue(3000);  
box.addElement(A);
```

```
Element B = new Node("B");  
B.setName("B");  
B.putChartColor(Color.BLUE);  
B.putChartValue(1000);  
box.addElement(B);
```





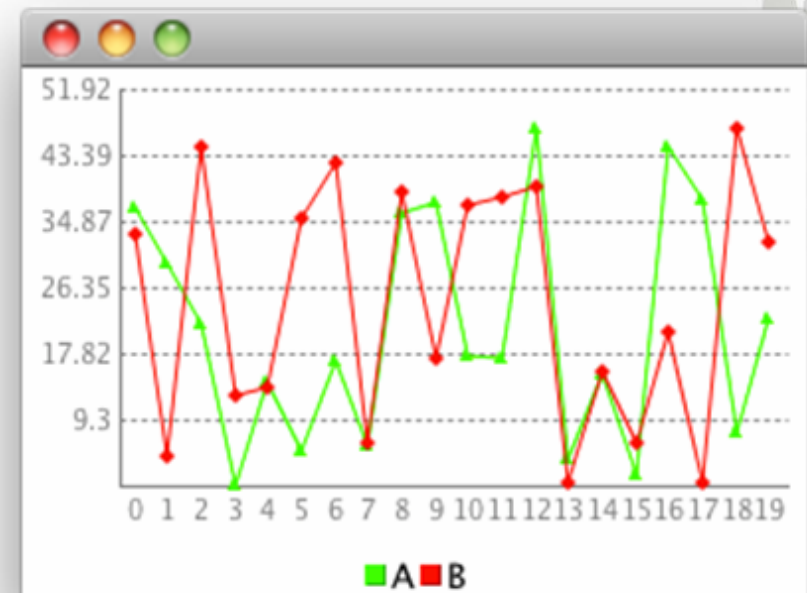
# LineChart示例

```

TDataBox box = new TDataBox();
LineChart lineChart = new LineChart(box);
lineChart.setYAxisVisible(true);
lineChart.setYScaleTextVisible(true);
lineChart.setXAxisVisible(true);
lineChart.setXScaleTextVisible(true);
// Display markers for each data value.
lineChart.setInflexionVisible(true);

Element a = new Node();
a.setName("A");
a.putChartColor(Color.GREEN);
a.putChartInflexionStyle(TWaverConst.INFLEXION_STYLE_TRIANGLE);
box.addElement(a);
Element b = new Node();
b.setName("B");
b.putChartColor(Color.RED);
b.putChartInflexionStyle(TWaverConst.INFLEXION_STYLE_DIAMOND);
box.addElement(b);
for (int i = 0; i < 20; i++) {
    lineChart.addXScaleText("" + i);
    a.addChartValue(Math.random() * 50);
    b.addChartValue(Math.random() * 50);
}

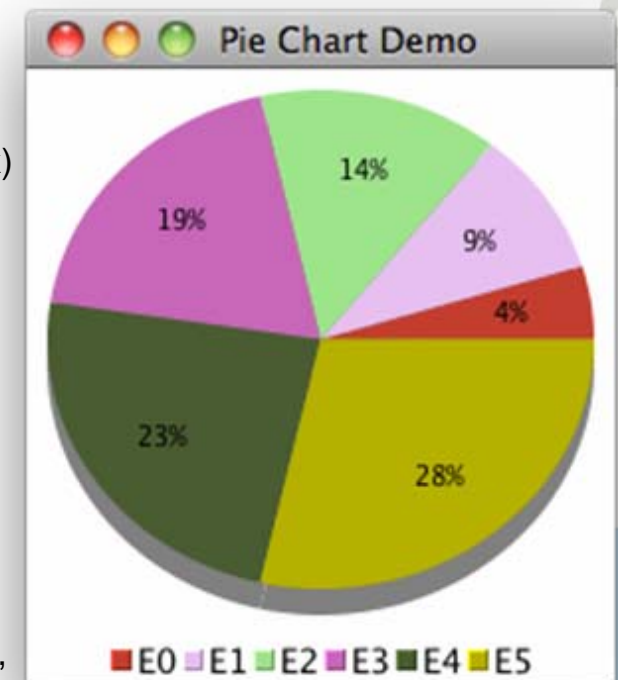
```



# PieChart示例

```
TDataBox box = new TDataBox();
PieChart pieChart = new PieChart(box){
    protected String getFormattedText(Element element, double value, int index)
    {
        double proportion=element.getChartValue()/sum;
        return (int) (proportion * 100) + "%";
    }
};
Random random = new Random();
for (int i = 0; i < 6; i++) {
    Element element = new Node();
    element.setName("E" + i);
    element.putChartValue(10 * (i + 1));
    element.putChartColor(new Color(random.nextInt(255), random.nextInt(255),
    random.nextInt(255)));
    box.addElement(element);
}

TestUtil.showFrame("Pie Chart Demo", pieChart);
```



# 告警的使用

- 告警：反映网元或设备运行状态的业务元素
- 告警对象 / 告警级别
- 告警容器
- 告警状态 / 告警统计
- 告警呈现

# 告警级别

- `twaver. AlarmSeverity`
- 告警级别：反映告警的紧急程度  
包含：`name`, `nickName`, `value`, `color`等属性
- 默认提供六级告警级别  
**CRITICAL / MAJOR / MINOR / WARNING /  
INDETERMINATE / CLEARED**
- 可以扩展告警级别  
`AlarmSeverity.clearAlarmSeverity();`  
`AlarmSeverity.registerAlarmSeverity("a", "a", 1, Color.RED, "AAA");`

# 定制告警级别示例

```
public static void main(String[] args) throws Exception {
```

```
    AlarmSeverity.clearAlarmSeverity();
```

```
    AlarmSeverity a = AlarmSeverity.registerAlarmSeverity("a", "a", 1, Color.RED, "AAA");
```

```
    AlarmSeverity b = AlarmSeverity.registerAlarmSeverity("b", "b", 2, Color.BLUE, "BBB");
```

```
    AlarmSeverity c = AlarmSeverity.registerAlarmSeverity("c", "c", 3, Color.GREEN, "CCC");
```

```
    TDataBox box = new TDataBox();
```

```
    Group node1 = new Group();
```

```
    node1.setExpand(true);
```

```
    Node node2 = new Node();
```

```
    node2.setLocation(100, 100);
```

```
    Node node3 = new Node();
```

```
    node3.setLocation(200, 150);
```

```
    node3.setParent(node1);
```

```
    node2.setParent(node1);
```

```
    box.addElementWithDescendant(node1);
```

```
    addAlarm(box.getAlarmModel(), node1, a);
```

```
    addAlarm(box.getAlarmModel(), node2, b);
```

```
    addAlarm(box.getAlarmModel(), node3, c);
```

```
    TestUtil.showFrame("Custom AlarmSeverity Demo", new TNetwork(box));
```

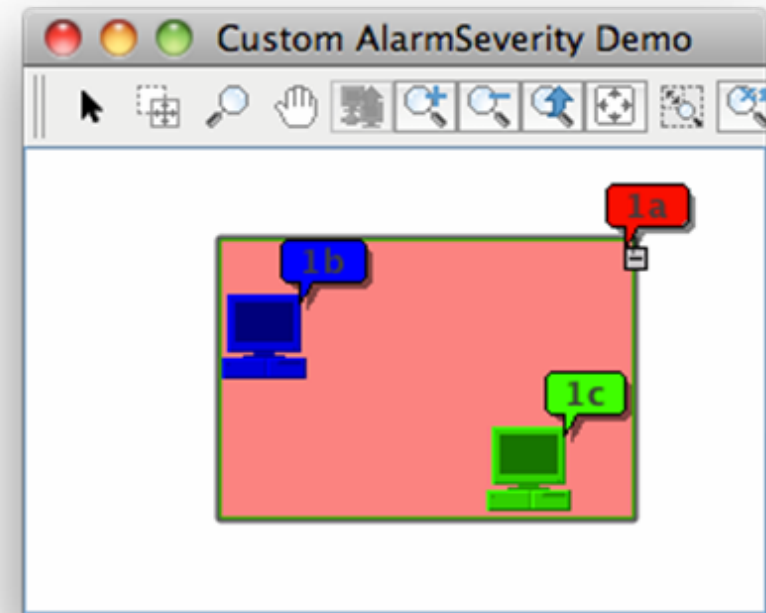
```
}

private static void addAlarm(AlarmModel alarmModel, Element element, AlarmSeverity severity){
```

```
    Alarm alarm = new Alarm(element.getID(), severity);
```

```
    alarmModel.addAlarm(alarm);
```

```
}
```



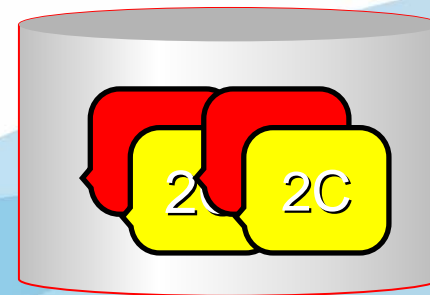


# 告警对象

- **twaver.Alarm:** 描述设备故障或者网络异常的数据元素
- 每个告警对象中包含：  
elementID、alarmSeverity、alarmID、acked、cleared、alarmState等  
.putClientProperty(key, value)添加属性

# 告警容器

- AlarmModel: 管理告警对象的容器
- 提供增加删除清空告警函数  
addAlarm/addAlarms  
removeAlarm\*\*\*
- 快速查找器 / 监听器...



AlarmModel

# 告警状态

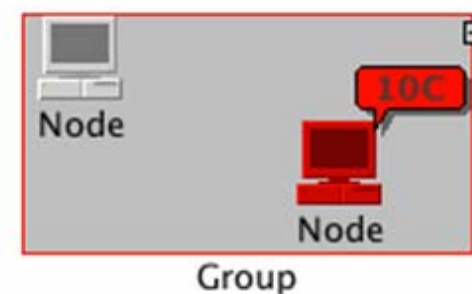
- **twaver.AlarmState**: 描述网元或设备当前的告警状态
- 包含网元的新发、确认、传递告警的统计信息
- 不包含具体的告警 (**twaver.Alarm**) 对象

# 告警状态统计

- **twaver.AlarmStateStatistics**告警状态统计：  
对整个dataBox中所有网元告警状态进行统计
- 包含各种级别告警的数量
- 包含新发、确认告警的数量
- 支持过滤器，可指定对哪些网元做统计

# 告警的呈现

- 告警冒泡
- 告警渲染（染色，边框）
- 告警表格
- 告警统计图表



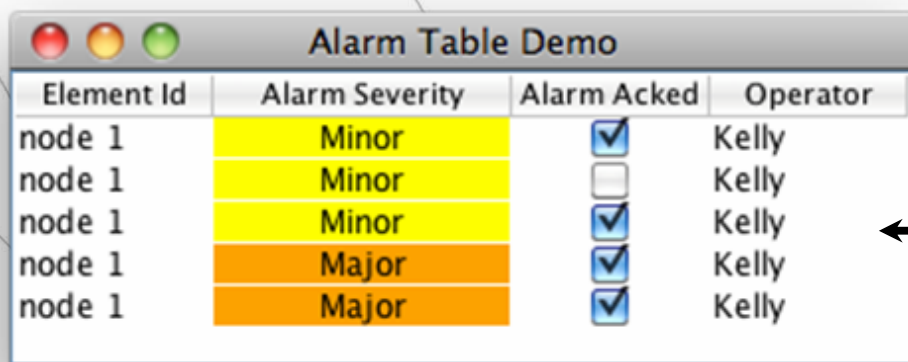
Alarm Table Demo			
Element Id	Alarm Severity	Alarm Acked	Operator
node 1	Minor	<input checked="" type="checkbox"/>	Kelly
node 1	Minor	<input type="checkbox"/>	Kelly
node 1	Minor	<input checked="" type="checkbox"/>	Kelly
node 1	Major	<input checked="" type="checkbox"/>	Kelly
node 1	Major	<input checked="" type="checkbox"/>	Kelly

Severity	New	Acked	Total
Critical	0	0	0
Major	0	1	1
Minor	1	1	2
Warning	2	0	2
Indetermin...	0	1	1
Cleared	0	0	0
Total	3	3	6



# TAlarmTable的使用

- TAlarmTable用于呈现AlarmModel中告警的信息，与告警容器构成了MV的关系
- 与TElementTable类似，也具有排序，过滤等功能



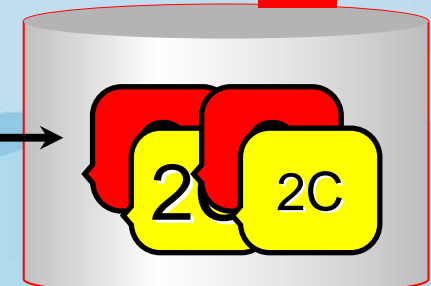
Alarm Table Demo

Element Id	Alarm Severity	Alarm Acked	Operator
node 1	Minor	<input checked="" type="checkbox"/>	Kelly
node 1	Minor	<input type="checkbox"/>	Kelly
node 1	Minor	<input checked="" type="checkbox"/>	Kelly
node 1	Major	<input checked="" type="checkbox"/>	Kelly
node 1	Major	<input checked="" type="checkbox"/>	Kelly

V

每行代表一个告警对象

M



AlarmModel

# TAlarmTable示例

```
TDataBox box=new TDataBox();
Node node = new Node("node 1");
box.addElement(node);

TAlarmTable alarmTable = new TAlarmTable(box, new TTableColumn[]{
    new TTableColumn(Alarm.PROPERTY_ELEMENTID, "Element Id"),
    new TTableColumn(Alarm.PROPERTY_ALARMSEVERITY, "Alarm Severity"),
    new TTableColumn(Alarm.PROPERTY_ACKED, "Alarm Acked"),
    new TTableColumn("operator", "Operator"),
    new TTableColumn(Alarm.PROPERTY_RAISEDTIME, "Raised Time")
});

alarmTable.setConverseIncreaseOrder(true);

AlarmModel alarmModel = box.getAlarmModel();
for (int i = 0; i < 5; i++) {
    AlarmSeverity severity = TWaverUtil.getRandomNonClearedSeverity();
    Alarm alarm = new Alarm(node.getID(), severity);
    alarm.setAcked(TWaverUtil.getRandomBool());
    alarm.setRaisedTime(new Date(System.currentTimeMillis() + i*3600));
    alarm.putClientProperty("operator", TWaverUtil.getRandomBool() ? "Sam" : "Kelly");
    box.getAlarmModel().addAlarm(alarm);
}

TestUtil.showFrame("Alarm Table Demo", new JScrollPane(alarmTable));
```

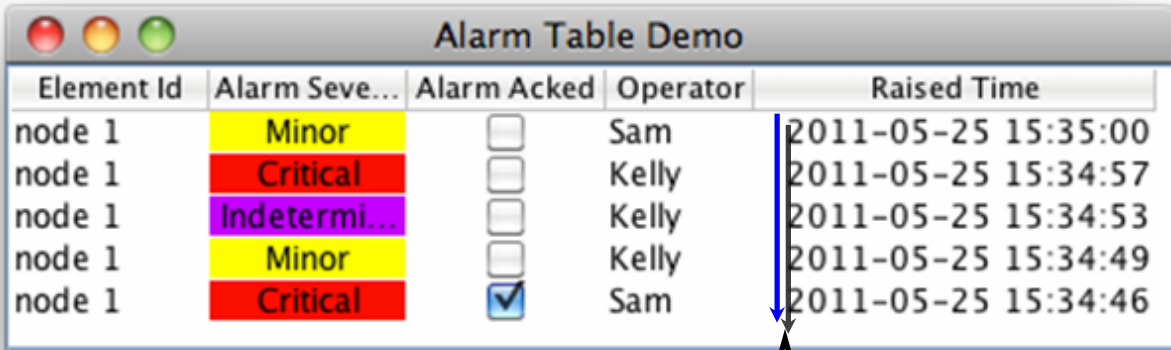
定制显示哪些列

设置反序，这样后加入的显示在上方

添加告警对象到 AlarmModel

# TAlarmTable示例

运行界面:



The image shows a window titled "Alarm Table Demo" with a table of alarm data. The table has five columns: "Element Id", "Alarm Seve...", "Alarm Acked", "Operator", and "Raised Time". The data is as follows:

Element Id	Alarm Seve...	Alarm Acked	Operator	Raised Time
node 1	Minor	<input type="checkbox"/>	Sam	2011-05-25 15:35:00
node 1	Critical	<input type="checkbox"/>	Kelly	2011-05-25 15:34:57
node 1	Indetermi...	<input type="checkbox"/>	Kelly	2011-05-25 15:34:53
node 1	Minor	<input type="checkbox"/>	Kelly	2011-05-25 15:34:49
node 1	Critical	<input checked="" type="checkbox"/>	Sam	2011-05-25 15:34:46

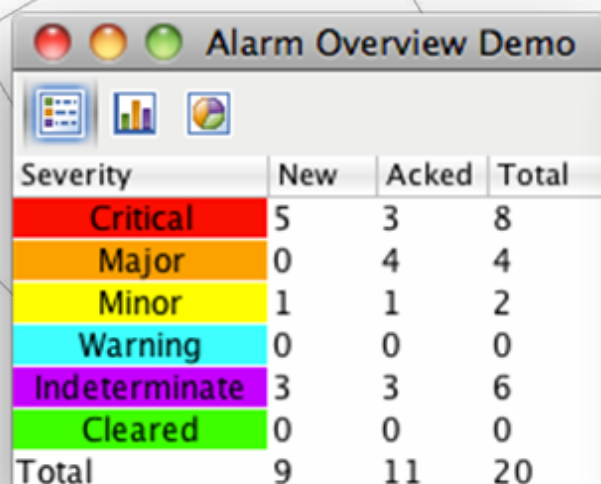
A blue arrow points from the "Raised Time" column to a callout box below the table.

设置反序显示

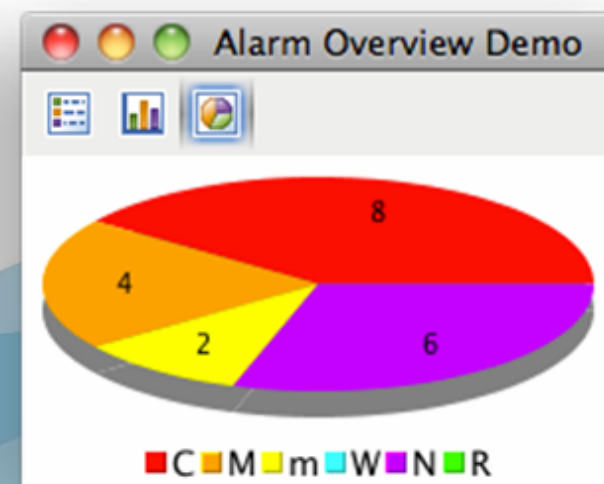
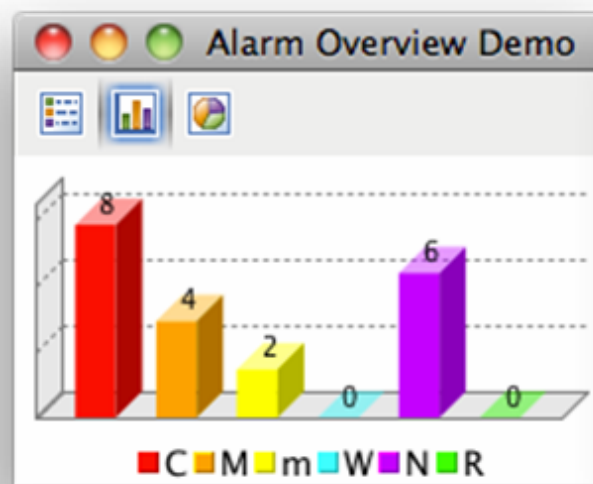
```
alarmTable.setConverseIncreaseOrder(true);
```

# TAlarmOverview

- TAlarmOverview组件用于呈现告警统计信息，包含三种展现方式：表格、柱状图、饼图



Severity	New	Acked	Total
Critical	5	3	8
Major	0	4	4
Minor	1	1	2
Warning	0	0	0
Indeterminate	3	3	6
Cleared	0	0	0
Total	9	11	20

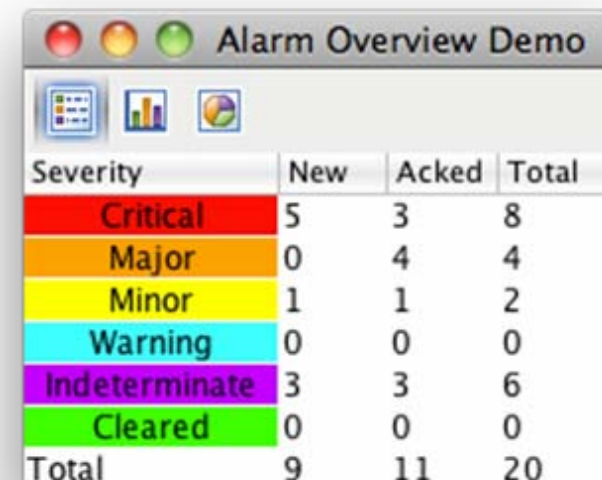


# TAlarmOverview示例

```
TDataBox box=new TDataBox();
Node node = new Node("node 1");
box.addElement(node);
Node node2 = new Node("node 2");
box.addElement(node2);

for (int i = 0; i < 20; i++) {
    AlarmSeverity severity = TWaverUtil.getRandomNonClearedSeverity();
    Alarm alarm = new Alarm((TWaverUtil.getRandomBool() ? node : node2).getID(), severity);
    alarm.setAcked(TWaverUtil.getRandomBool());
    alarm.putClientProperty("operator", TWaverUtil.getRandomBool() ? "Sam" : "Kelly");
    box.getAlarmModel().addAlarm(alarm);
}

TAlarmOverview overview = new TAlarmOverview(box);
TestUtil.showFrame("Alarm Overview Demo", overview);
```

A screenshot of a software window titled "Alarm Overview Demo". It contains a table with alarm statistics. The table has four columns: Severity, New, Acked, and Total. The rows are: Critical (5 New, 3 Acked, 8 Total), Major (0 New, 4 Acked, 4 Total), Minor (1 New, 1 Acked, 2 Total), Warning (0 New, 0 Acked, 0 Total), Indeterminate (3 New, 3 Acked, 6 Total), Cleared (0 New, 0 Acked, 0 Total), and a Total row (9 New, 11 Acked, 20 Total). The rows are color-coded: Critical is red, Major is orange, Minor is yellow, Warning is cyan, Indeterminate is purple, and Cleared is green.

Severity	New	Acked	Total
Critical	5	3	8
Major	0	4	4
Minor	1	1	2
Warning	0	0	0
Indeterminate	3	3	6
Cleared	0	0	0
Total	9	11	20



# Thank you

- 论坛: [twaver.servasoft.com/forum](http://twaver.servasoft.com/forum)
- MSN: [twavercn@hotmail.com](mailto:twavercn@hotmail.com)