

TWaver .NET 培训

2011

Serva Software LLC

TWaver .NET 介绍

TWaver™

- TWaver是什么？
- .NET基础
 - C#.NET之间的关系
 - Silverlight & WPF ...
- TWaver .NET 基础
 - Hello TWaver!
 - MVC 设计模式
 - 数据元素/数据容器/视图
- 与TWaver Java & Flex的比较

TWaver是什么？

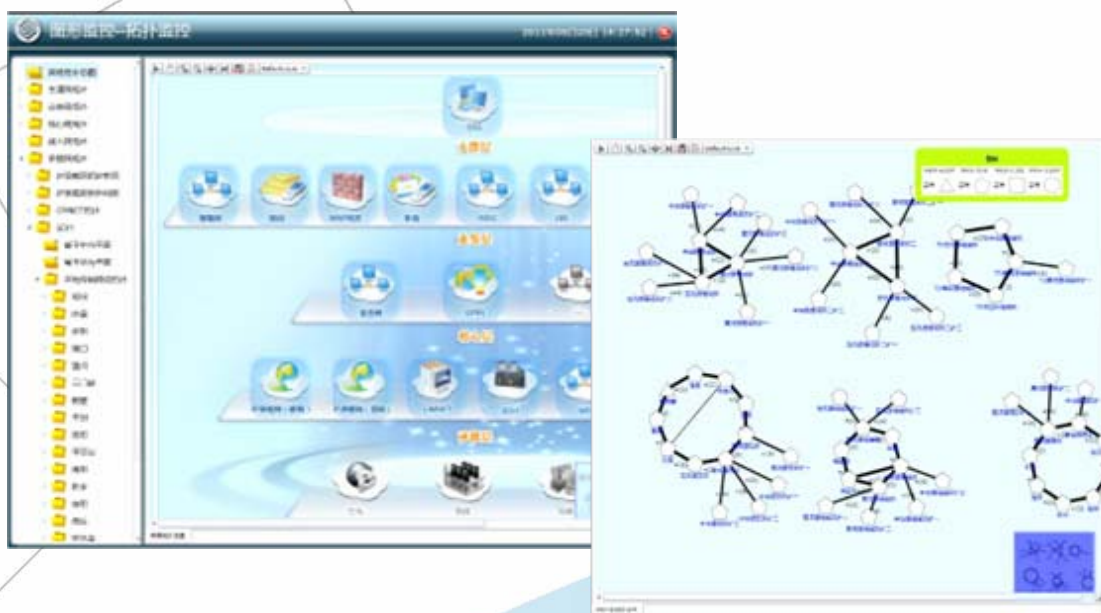
TWaver™

- 高效轻量的图形组件库
- 多平台解决方案
- TWaver .NET 产品包结构
- TWaver 许可

TWaver是一个图形组件库

- TWaver 关注于数据的图形呈现
- 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

TWaver™



- [-] TWaver Silverlight Demos
 - [x] Alarm Demos
 - [x] Alarm Statistics Demo
 - [x] Alarm Mapping Demo
 - [x] Alarm Propagation Demo
 - [-] Network Demos
 - [x] Topology Demos
 - [x] Equipment Demos
 - [x] Tree Demos
 - [x] Tree Layout Demo
 - [x] File Tree Demo

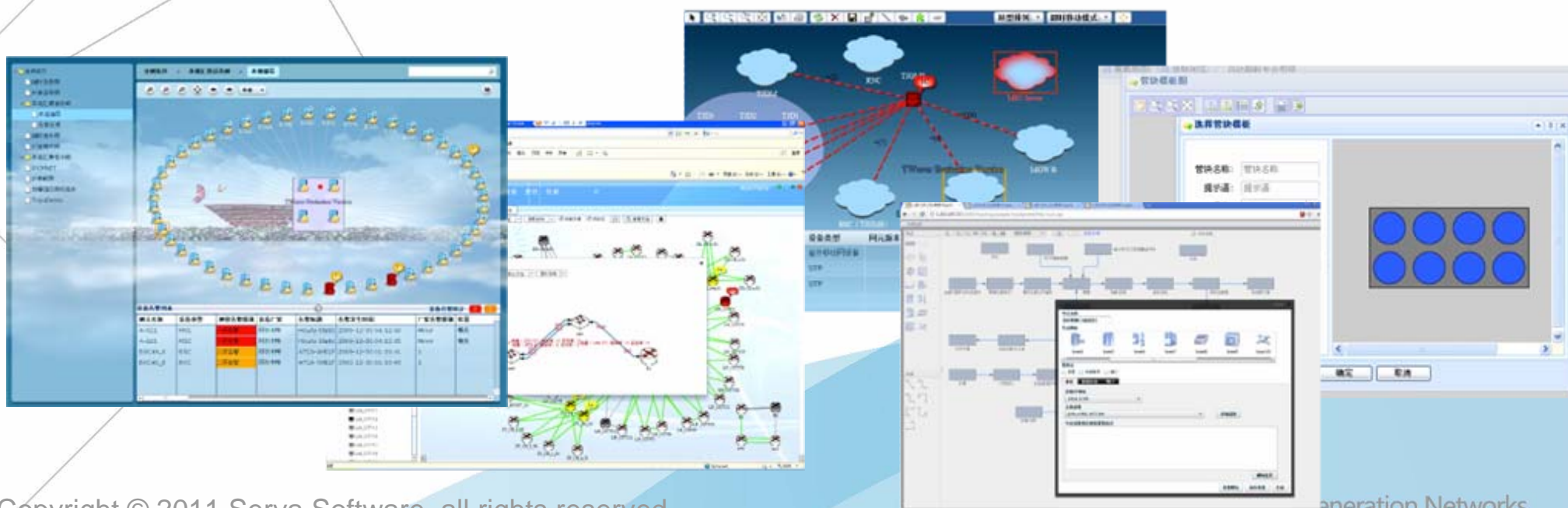


Mapping ID	Severity	Acked	Cleared	Raised Time
1	Indeterminate	<input type="checkbox"/>	<input type="checkbox"/>	7/29/2011 10:25:03 AM
2	Warning	<input type="checkbox"/>	<input type="checkbox"/>	7/29/2011 10:25:03 AM
3	Minor	<input type="checkbox"/>	<input type="checkbox"/>	7/29/2011 10:25:03 AM
4	Major	<input type="checkbox"/>	<input type="checkbox"/>	7/29/2011 10:25:03 AM
5	Critical	<input type="checkbox"/>	<input type="checkbox"/>	7/29/2011 10:25:03 AM
5	Indeterminate	<input checked="" type="checkbox"/>	<input type="checkbox"/>	7/29/2011 10:25:03 AM
4	Warning	<input checked="" type="checkbox"/>	<input type="checkbox"/>	7/29/2011 10:25:03 AM

关注于电信网管行业

- 提供电信行业业务模型，提供设备面板，告警传播等机制
- 拥有大量电信行业应用解决方案
- 但并不局限于电信行业

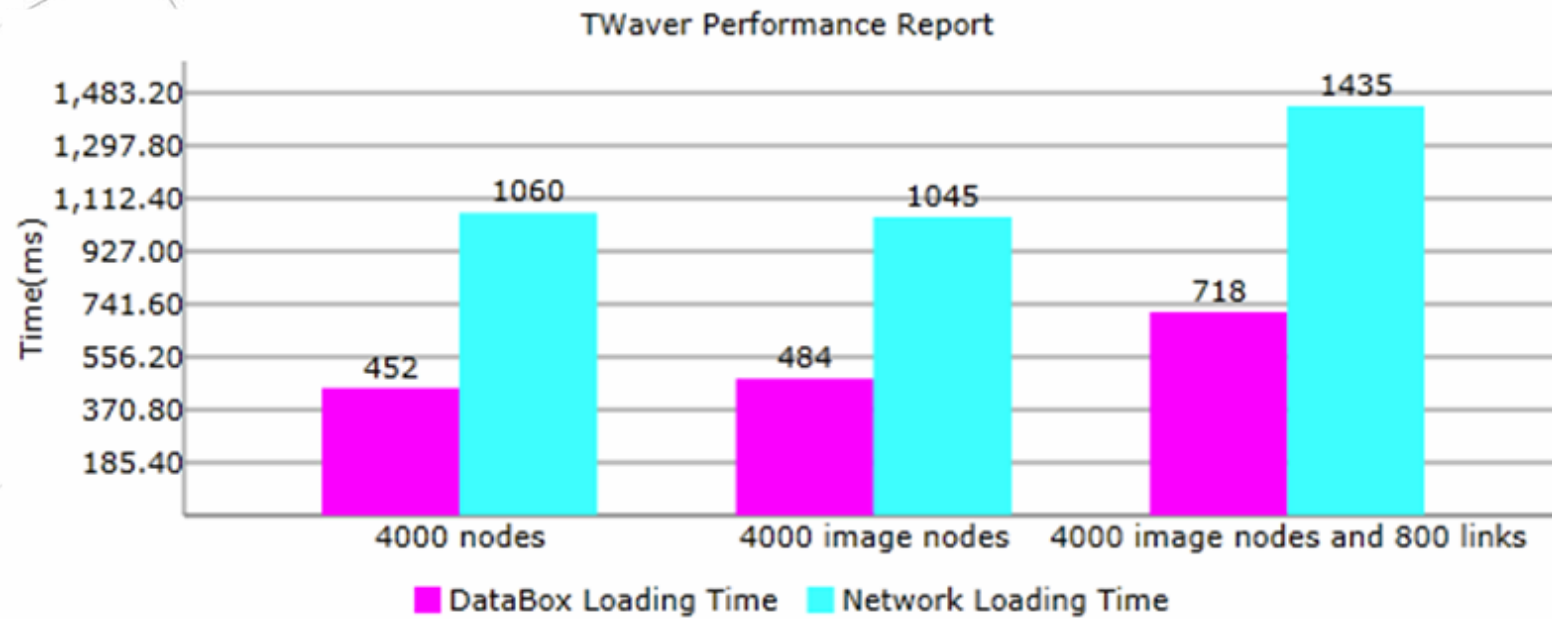
TWaver™



高效轻量

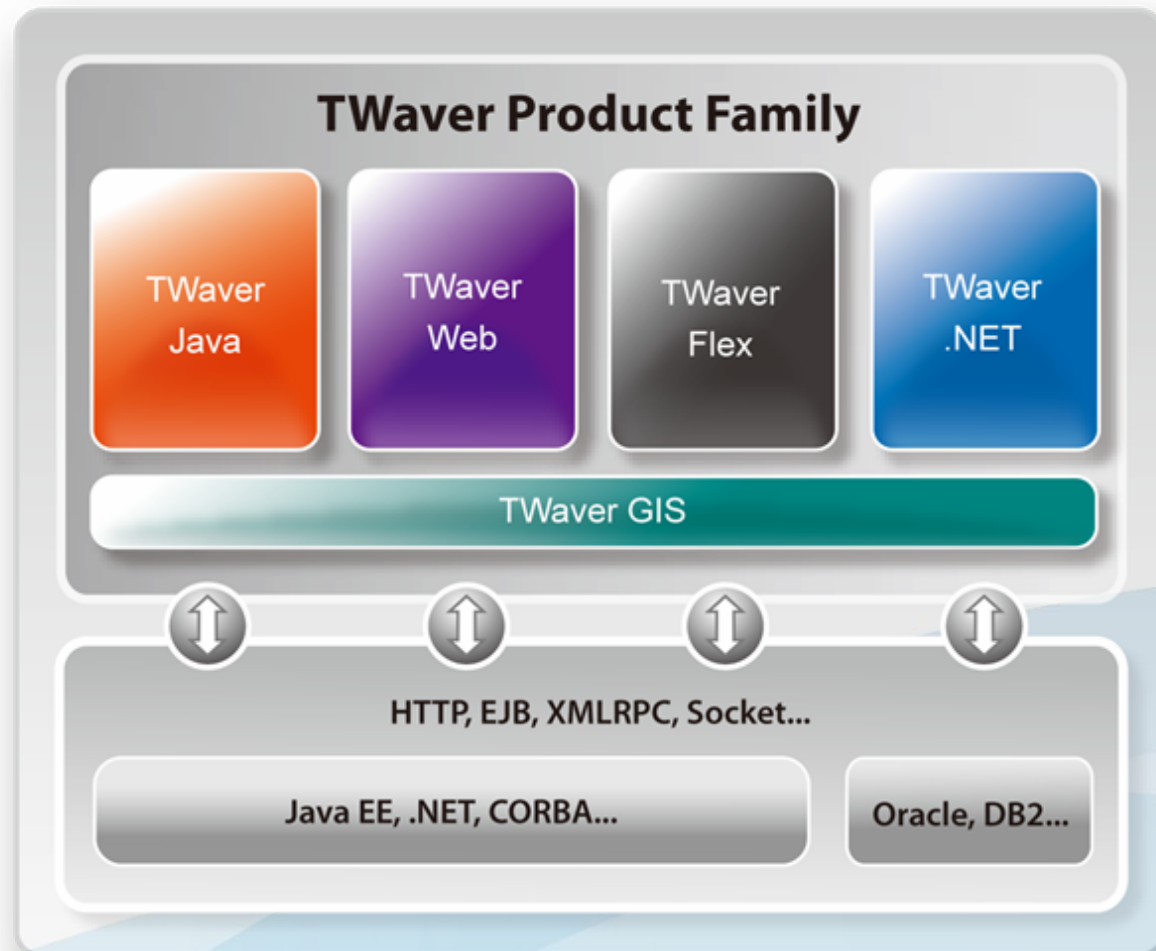
TWaver™

- TWaver.Silverlight.dll 900kb
- 建议网元数量控制在6000以内（节点和连线）
- 加载4000节点和800条连线需1.5秒



多语言平台解决方案

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

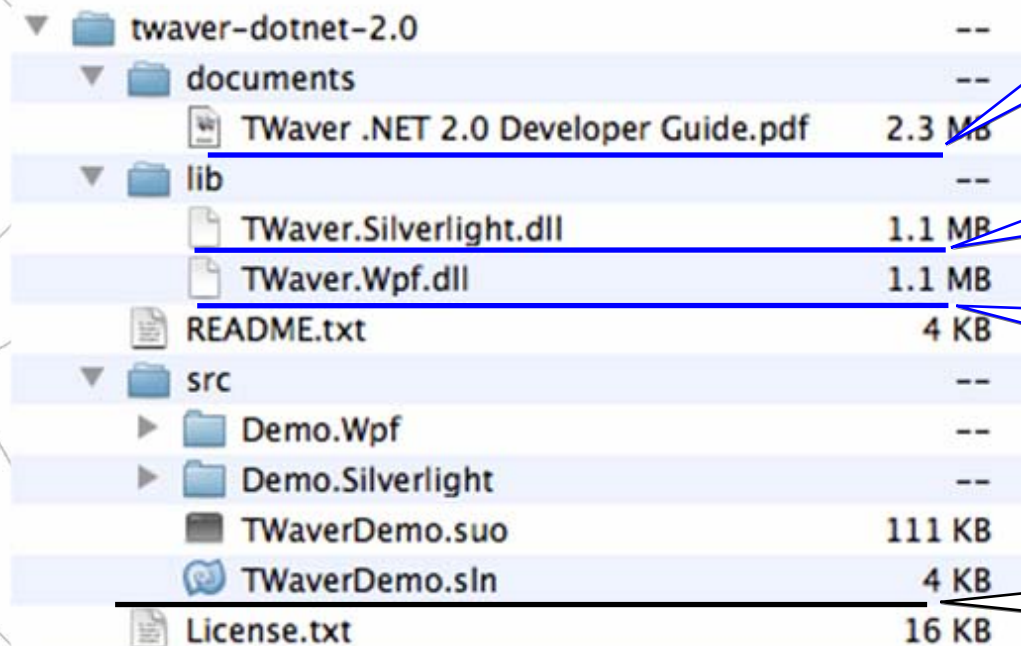


TWaver™



TWaver .NET 产品包内容

TWaver™



▼ twaver-dotnet-2.0	--
▼ documents	--
TWaver .NET 2.0 Developer Guide.pdf	2.3 MB
▼ lib	--
TWaver.Silverlight.dll	1.1 MB
TWaver.Wpf.dll	1.1 MB
README.txt	4 KB
▼ src	--
▶ Demo.Wpf	--
▶ Demo.Silverlight	--
TWaverDemo.suo	111 KB
TWaverDemo.sln	4 KB
License.txt	16 KB

开发手册

TWaver.Silverlight.dll

TWaver.Wpf.dll

Silverlight和WPF
示例源码

TWaver 许可

TWaver™

- TWaver有三种许可：试用许可，开发许可，运行许可，在Silverlight或者WPF的拓扑图界面按快捷键**Ctrl+Shift+L**，会显示许可信息
- **试用许可**：可免费申请，用于前期预言或技术选型阶段，组件界面在五分钟后显示"TWaver Evaluation Version"水印
- **开发许可**：用于项目实际开发阶段，组件界面在两个小时会显示开发版水印
- **运行许可**：用于项目实际运行阶段，无水印

如何使用许可文件

TWaver™

- **license.xml** 是一个纯文本文件，它包含了许可的信息
- 使用方法:

```
TWaver.Utills.ValidateLicense(new Uri(  
    "/PROJECT_NAME;component/license.xml",  
    UriKind.Relative));
```

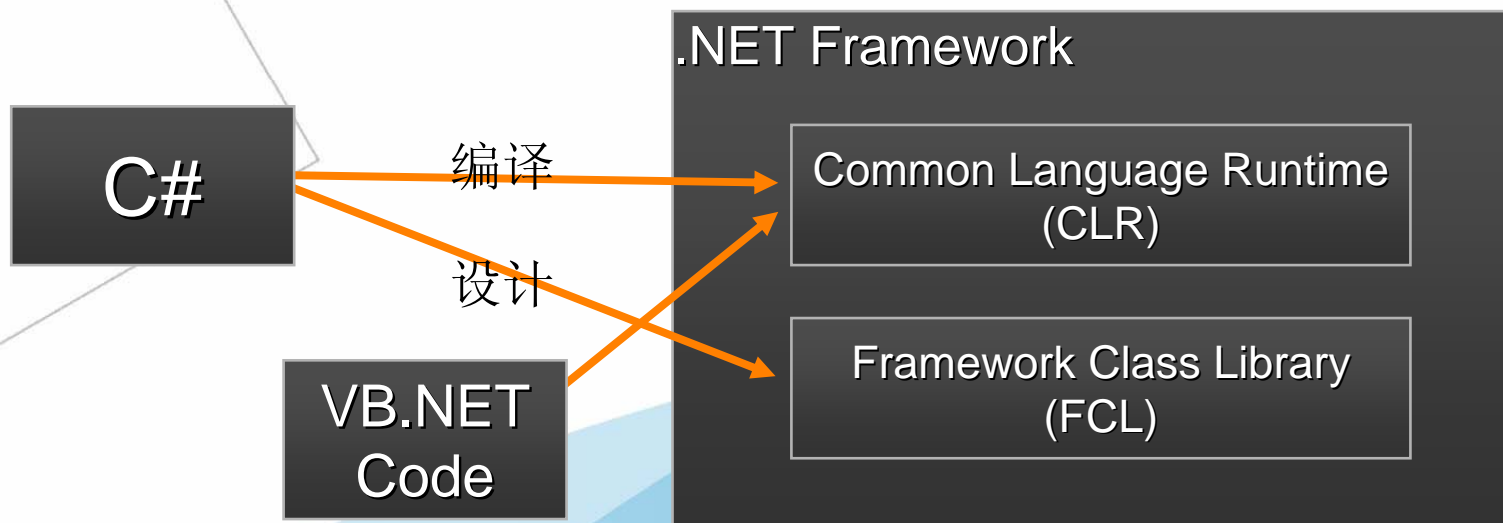
.NET 基础

- C# & .NET
- C# 语言
- Silverlight & WPF
- 其他

Twaver™

C#与.NET的关系

- C# 是专门针对微软.NET架构设计的编程语言
- 但C#本身并不是.NET的一部分，一些.NET功能C#并不支持，同样有些C#语言的特性.NET也不支持



C# 语言特性

“C#是一种多范型的编程语言，包括强类型式，命令式，声明式，函数式，泛型，面向对象和面向组件的编程方式” – 选自wikipedia

- Get & Set
- 泛型
- 委托, Lambdas 和事件

get & set

```
namespace TWaver{  
    public class Data : IData {  
  
    ...  
  
        private string name;  
        public virtual string Name  
        {  
            get { return this.name; }  
            set  
            {  
                object oldValue = this.name;  
                this.name = value;  
                this.DispatchPropertyChangeEvent("Name", oldValue,  
value);  
            }  
        }  
        ...  
    }  
}
```

使用:

```
Data data = new Data();  
data.Name = "001";  
Console.WriteLine(data.Name);
```

TWaver™

泛型

- 泛型类

```
public class List<T> { }
```

```
public delegate TOutput Converter<TInput, TOutput>(TInput  
    from);
```

```
public interface IComparable<in T> {  
    int CompareTo(T other);  
}
```

- 使用泛型

```
List<Node> nodes = new List<Node>();
```

```
DataBox<Element> box = new DataBox<Element>();
```

```
Tree<Element> tree = new Tree<Element>(box);
```

委托, Lambda和事件

定义一个委托

```
delegate TOutput Converter<TInput, TOutput>(TInput from);
```

创建一个委托实例

```
Converter<int, string> g = delegate(int input){  
    return "AA-" + input;  
};
```

通过Lambda表达式创建委托实例

```
Converter<int, string> g2 = (int input) =>{  
    return "AA-" + input;  
};
```

使用Lambda添加事件监听

```
tree.MouseLeftButtonDown += (object sender, MouseButtonEventArgs evt) =>{  
    Console.WriteLine("mouse click");  
};
```

WPF & Silverlight

TM
waver

- WPF - Windows Presentation Foundation, 一个用于优雅客户端应用提供界面组件的程序库
- Silverlight的组成:
 - 一个核心呈现框架（WPF的一个子集），不包含流文档，固定文档和3-D...
 - .NET框架中的Silverlight部分 (.NET 框架的一个子集)
 - 安装器和更新器.

WPF & Silverlight

TM
waver

- WPF应用程序运行在Windows系统下
- Silverlight使用插件模式寄存在浏览器中
- WPF程序打包成一个二进制格式的可运行程序集
- Silverlight程序会被打包成一个XAP后缀的Zip文件，其中包含一个程序集和配置信息

其他

Twaver™

- XAML & C#
- class & struct
- DispatcherObject, DependencyObject, Visual, UIElement, FrameworkElement, Control, ContentControl, ItemsControl, Panel, Application
- Shape, Path, Geometry, Transform ...

TWaver 基础

TWaver™

- Hello TWaver
- MVC 设计模式
- 数据元素和数据容器
- 视图

Hello TWaver

TWaver™

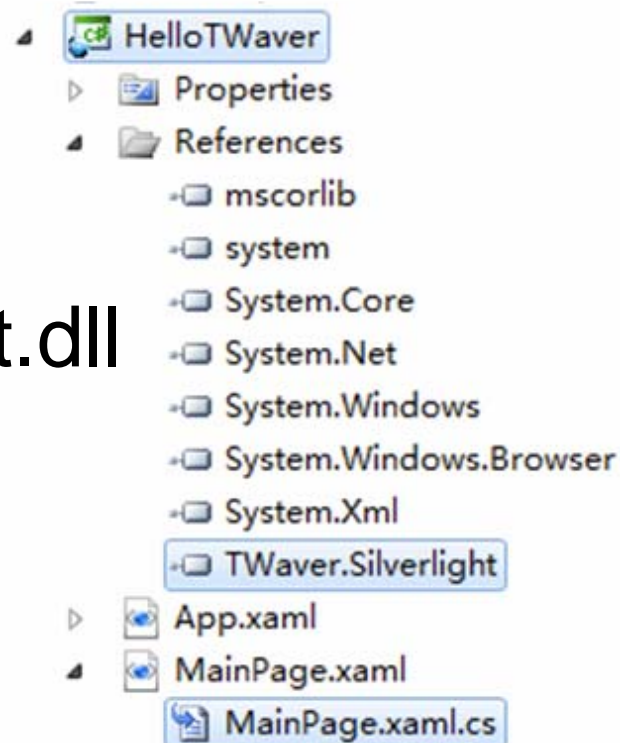
TWaver .NET 开发环境

- TWaver.Silverlight.dll / TWaver.Wpf.dll
- Silverlight 4 Tools for Visual Studio 2010 / .NET Framework 4
- Visual Studio 2010
- Silverlight 4 / .NET Framework 4 Client Profile or .NET Framework 4

Hello TWaver

- 创建一个Silverlight 应用程序
- 选择.NET Framework 4
- 添加引用 - TWaver.Silverlight.dll

参考开发手册中的“开发环境”



TWaver™

Hello TWaver

TWaver™

```
using TWaver;
using TWaver.NETwork;
namespace HelloTWaver{
    public partial class MainPage : UserControl {
        public MainPage() {
            InitializeComponent();

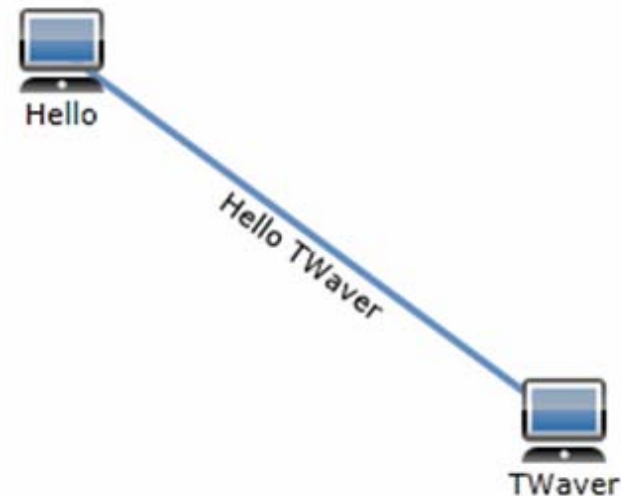
            ElementBox box = new ElementBox();

            Node node = new Node();
            node.Name = "Hello";
            node.SetLocation(10, 10);
            box.Add(node);

            Node node2 = new Node();
            node2.Name = "TWaver";
            node2.SetLocation(200, 150);
            box.Add(node2);

            Link link = new Link(node, node2);
            link.Name = "Hello TWaver";
            link.SetStyle(Styles.LINK_LABEL_ROTATABLE, true);
            box.Add(link);

            Network network = new Network(box);
            LayoutRoot.Children.Add(network);
        }
    }
}
```



配置 XAML

TWaver™

```
<navigation:Page x:Class="TWaverPPT.Demos.HelloTWaverFull"
xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
Title="HelloTWaverFull Page">

  <Grid x:Name="LayoutRoot" Background="White" Margin="20" >
    <Grid.ColumnDefinitions>
      <ColumnDefinition Width="170" />
      <ColumnDefinition Width="*" />
    </Grid.ColumnDefinitions>
    <Grid.RowDefinitions>
      <RowDefinition Height="*" />
      <RowDefinition Height="100" />
    </Grid.RowDefinitions>
  </Grid>
</navigation:Page>
```

添加树和表格

```
Network network = new Network(box);  
network.SetValue(System.Windows.Controls.Grid.ColumnProperty, 1);  
network.SetValue(System.Windows.Controls.Grid.RowProperty, 0);
```

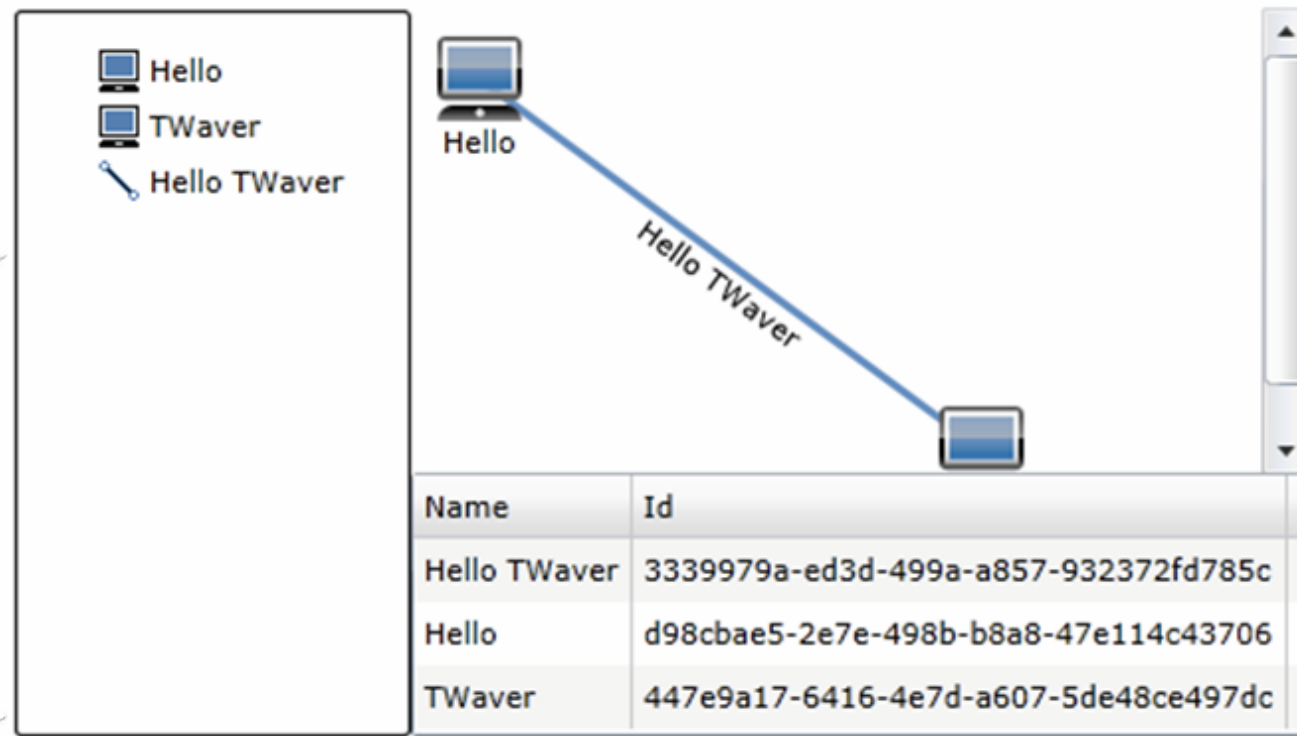
```
Tree<IElement> tree = new Tree<IElement>(box);  
tree.SetValue(System.Windows.Controls.Grid.ColumnProperty, 0);  
tree.SetValue(System.Windows.Controls.Grid.RowProperty, 0);  
tree.SetValue(System.Windows.Controls.Grid.RowSpanProperty, 2);
```

```
TWaver.Controls.Table<IElement> table = new TWaver.Controls.Table<IElement>(box);  
table.Columns.Add(new TableTextColumn(Consts.PROPERTY_TYPE_ACCESSOR,  
"Name", "Name"));  
table.Columns.Add(new TableTextColumn(Consts.PROPERTY_TYPE_ACCESSOR,  
"ID", "Id"));  
table.SetValue(System.Windows.Controls.Grid.ColumnProperty, 1);  
table.SetValue(System.Windows.Controls.Grid.RowProperty, 1);
```

```
LayoutRoot.Children.Add(network);LayoutRoot.Children.Add(tree);  
LayoutRoot.Children.Add(table);
```

Hello TWaver

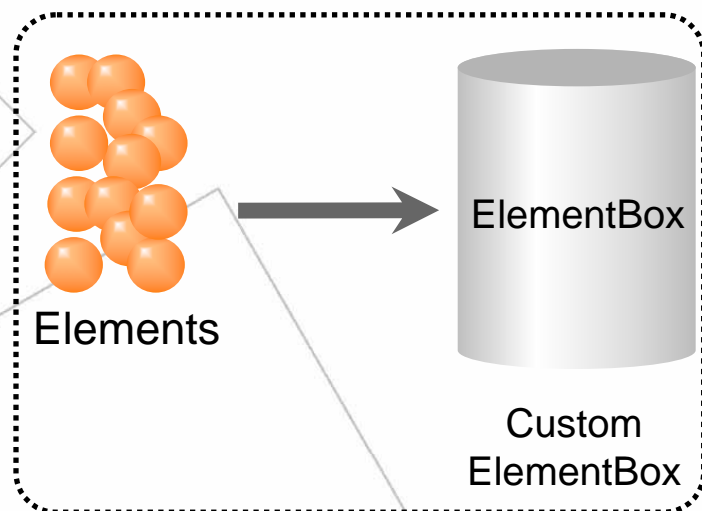
TWaver™



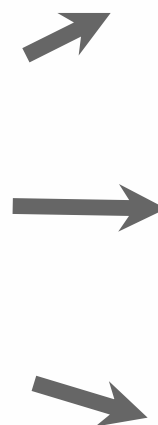
MVC 设计模式

TWaver™

M



V



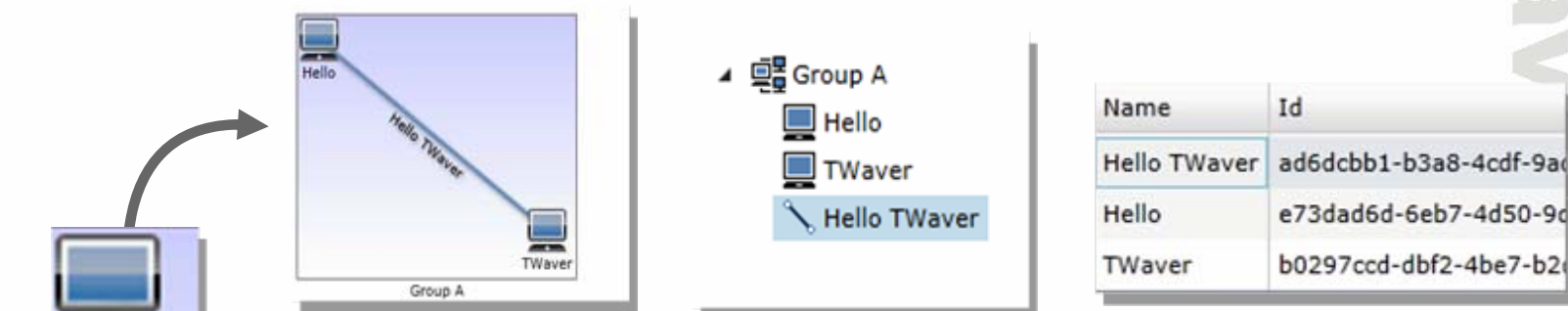
Name	Id
Hello TWaver	ad6dcbb1-b3a8-4cdf-9ad0-c6c5d07563f6
Hello	e73dad6d-6eb7-4d50-9ddd-9dc21dcbdaad
TWaver	b0297ccd-dbf2-4be7-b2d9-19d50d9f583f

interaction

C

Model & Views

Views(V)



Model(M)

Element

Node
Link
Group
...



ElementBo
X

事件驱动

Twaver™

Element

```
node.SetStyle(Styles.INNER_COLOR,  
Utils.CreateColor(0xFFFF0000))
```

```
DispatchPropertyChangeEvent
```

ElementBox

```
HandleDataPropertyChange
```

```
DispatchEvent
```

View

network, tree, chart ...

```
HandleElementPropertyChange
```

```
invalidate element UI
```



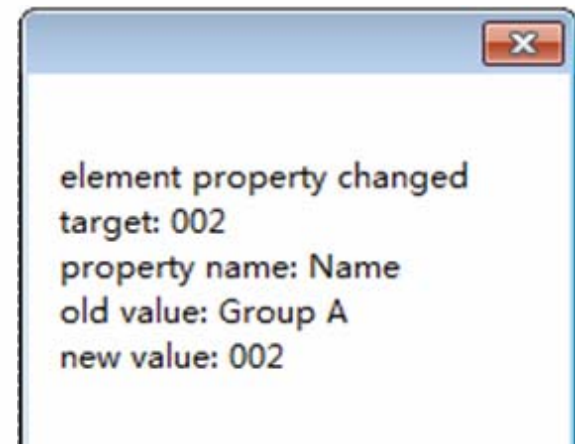
事件监听

TWaver™

```
box.DataPropertyChange += (PropertyChangeEvent<IElement> evt) => {  
    MessageBox.Show("element property changed\ntarget: " + evt.Source +  
        "\nproperty name: " + evt.PropertyName +  
        "\nold value: " + evt.OldValue +  
        "\nnew value: " + evt.NewValue);};
```

```
<Button Content="Change Name" Click="Button_Click" />
```

```
private void Button_Click(object sender, RoutedEventArgs e)  
{  
    box.Datas[0].Name = "002";  
}
```

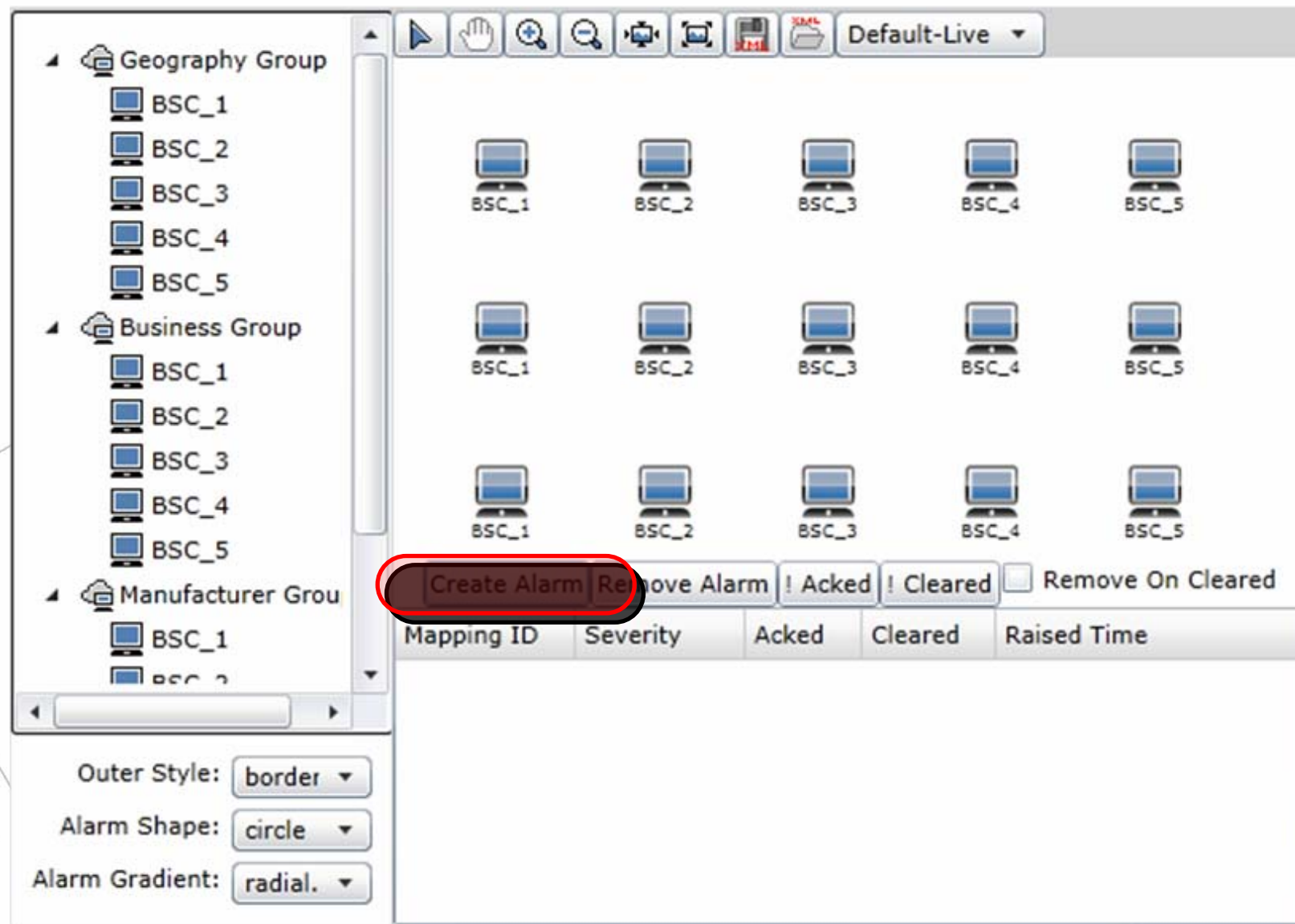


事件监听

Target	Listener
DataBox	PriorDataBoxChange / DataBoxChange / DataPropertyChange / HierarchyChange / PropertyChange
ElementBox	extends DataBox<IElement> IndexChange
AlarmBox	extends DataBox<IAlarm>
LayerBox	extends DataBox<ILayer>
SelectionModel	SelectionChange
Network	Interaction / PropertyChange
AlarmSeverity	AlarmSeverityChange

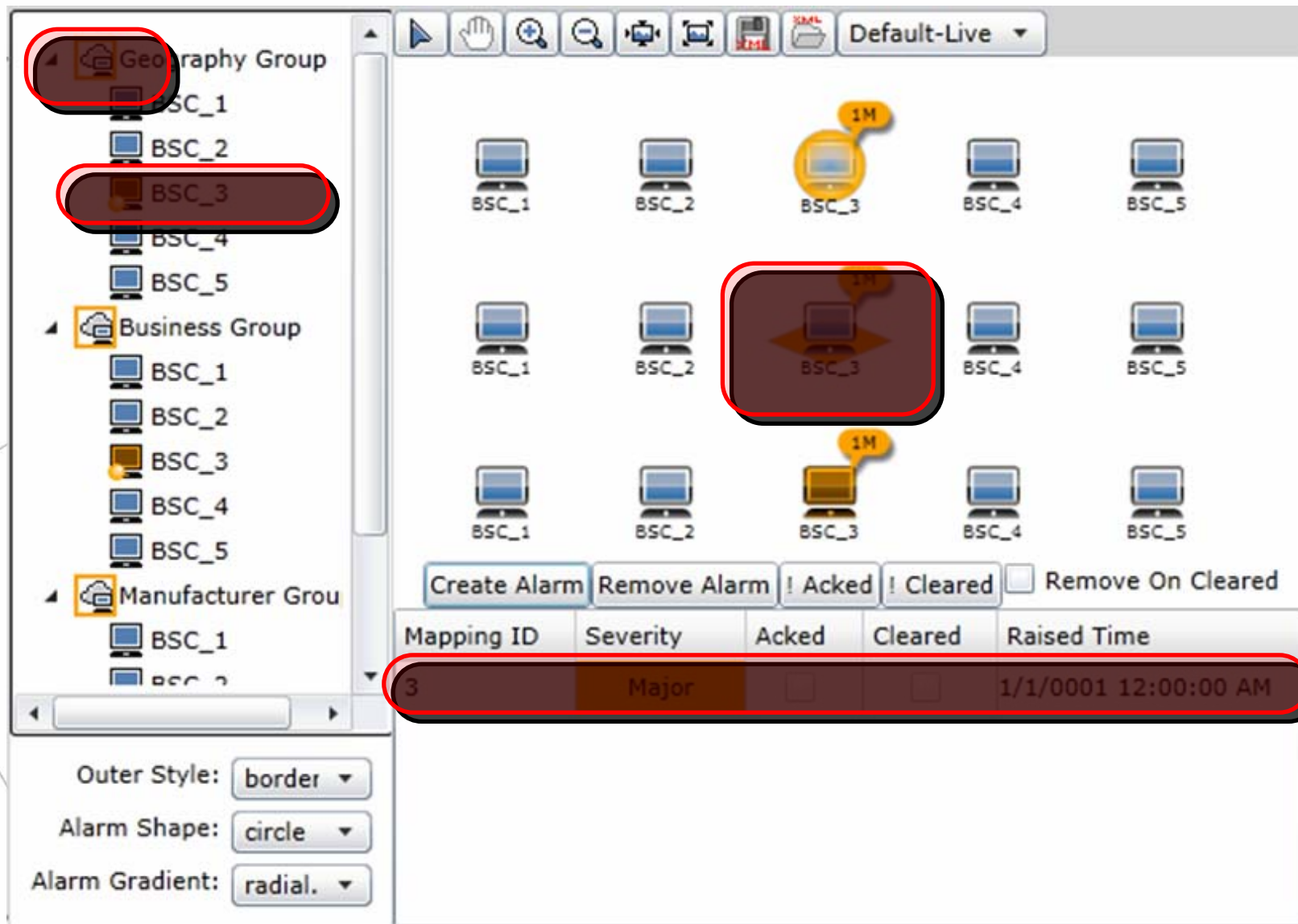
TWaver™

视图同步



TWaver™

视图同步



The screenshot displays the TWaver software interface. On the left, a hierarchical tree shows three groups: 'Geography Group', 'Business Group', and 'Manufacturer Group'. Each group contains five BSC (Base Station Controller) entries, labeled BSC_1 through BSC_5. The 'Geography Group' and 'BSC_3' are highlighted with red boxes. The main area shows a grid of BSC icons, with 'BSC_3' in the center highlighted by a red box. Below the grid, there are buttons for 'Create Alarm', 'Remove Alarm', 'Acked', 'Cleared', and 'Remove On Cleared'. At the bottom, a table displays alarm information.

Mapping ID	Severity	Acked	Cleared	Raised Time
3	Major	<input type="checkbox"/>	<input type="checkbox"/>	1/1/0001 12:00:00 AM

Outer Style: border
Alarm Shape: circle
Alarm Gradient: radial.

TWaver™

数据元素和数据容器

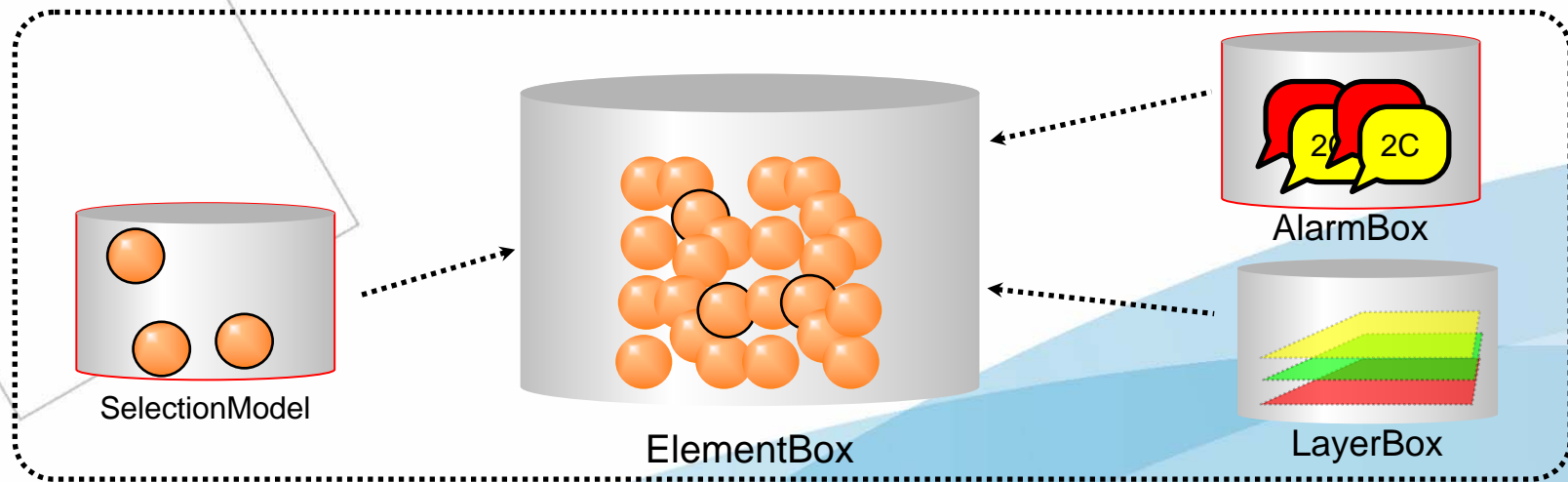
Data Item : 基本的数据单元

Data Model : 数据管理容器，提供数据的增加，删除，清除操作，统一管理数据元素的属性变化事件

Data Items	Data Models
IData	DataBox
IElement	ElementBox
IAalarm	AlarmBox
ILayer	LayerBox

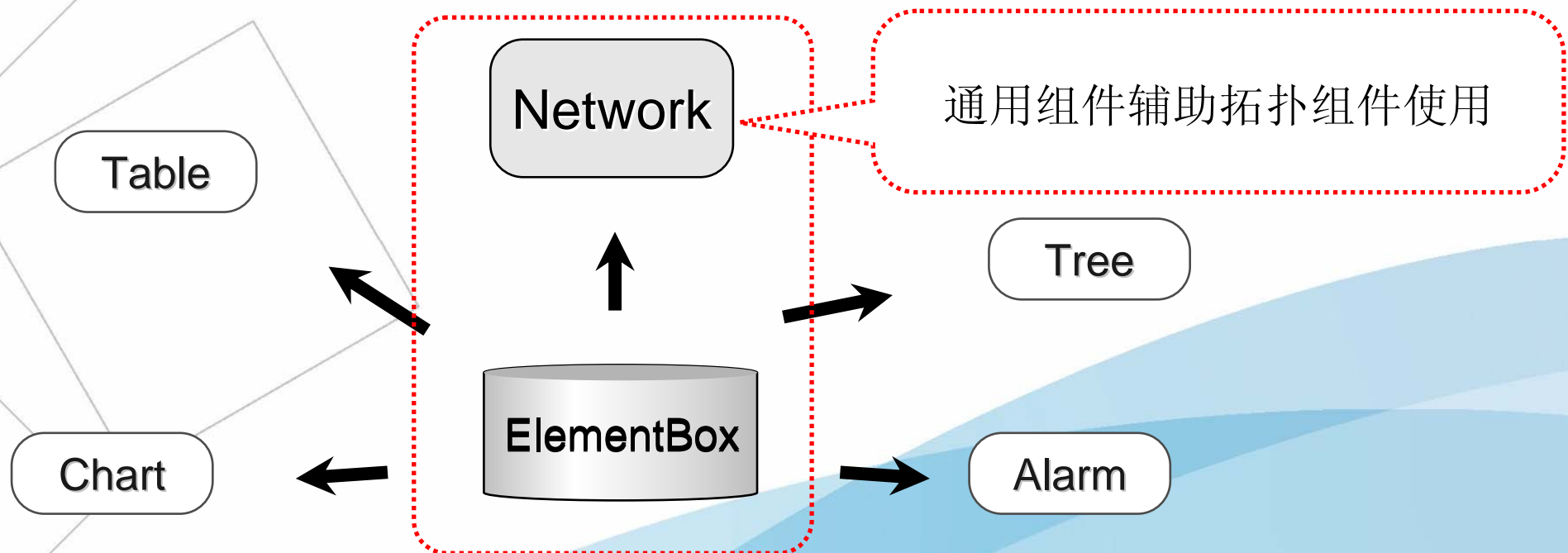
数据容器之间的关系

- **ElementBox**是用于管理拓扑网元的数据容器，它包含了一个告警容器，图层容器和选中模型



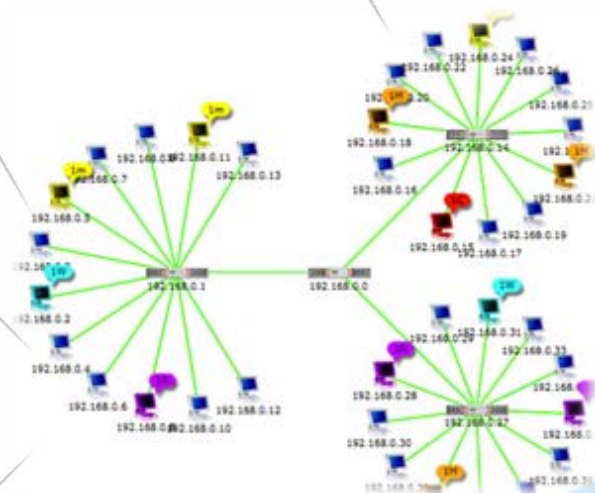
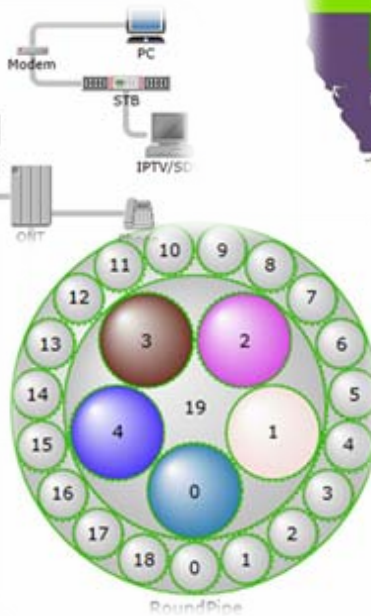
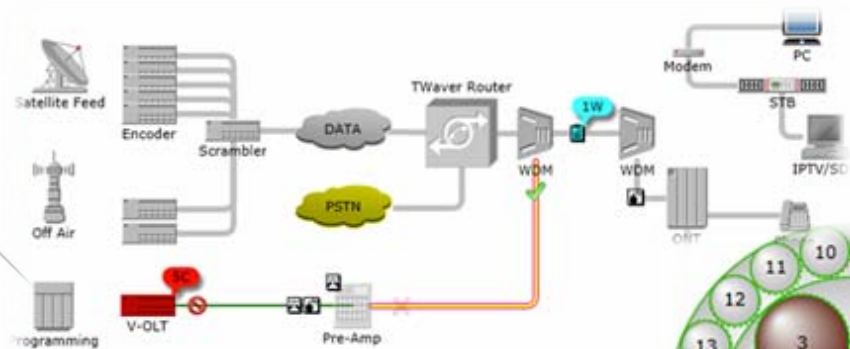
视图类型

- 拓组件件- Network
- 通用组件 - Tree, Table, Charts



Network

Twaver™



Tree

Label Folder

label

Icons Folder



Message Folder

Memory Usage 63%, 190M/512MB

Composite Folder

label message

label message

message label

message label

label message

message label

TWaver Silverlight Demos

Alarm Demos

Alarm Statistics Demo

Alarm Mapping Demo

Alarm Propagation Demo

Network Demos

Topology Demos

Equipment Demos

EMS Demo

Chassis Demo

Tree Demos

Tree Layout Demo

File Tree Demo

TWaver Silverlight Demos

Alarm Demos

Alarm Statistics Demo

Alarm Mapping Demo

Alarm Propagation Demo

Network Demos

Chart Demos

Pie Chart Demo NEW

Bar Chart Demo NEW

Line Chart Demo NEW

Bubble Chart Demo NEW

Radar Chart Demo NEW

Dial Chart Demo NEW

Editor Demos

Grid Editor Demo

Pipe Editor Demo







Topology Editor Demo

TWaver™

Table

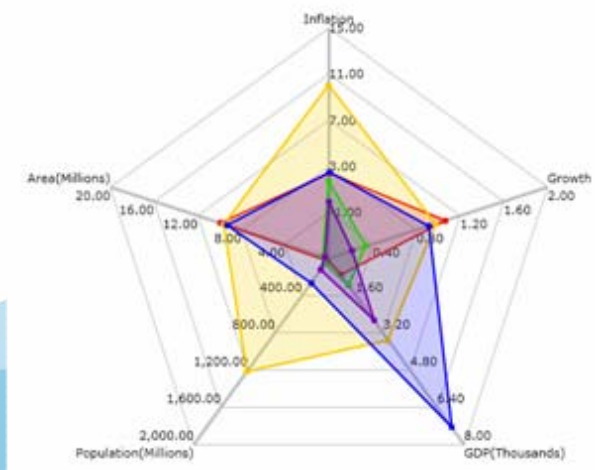
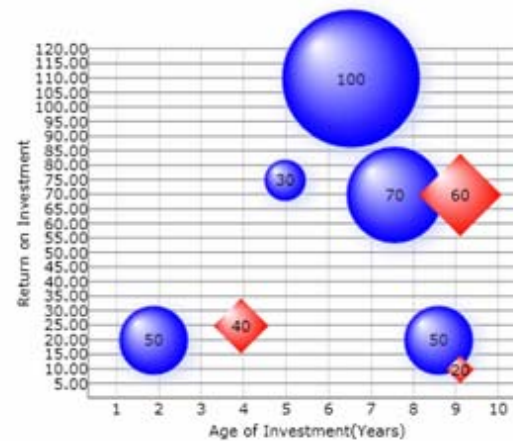
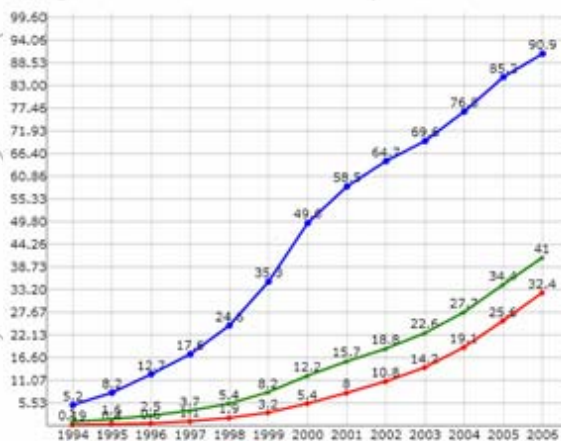
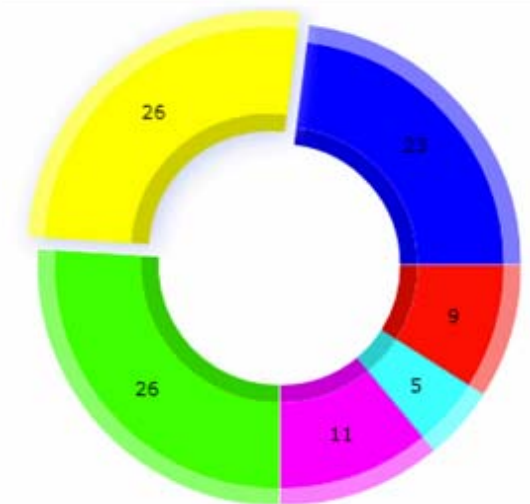
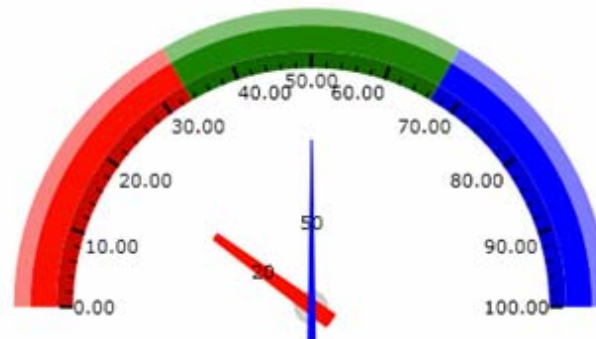
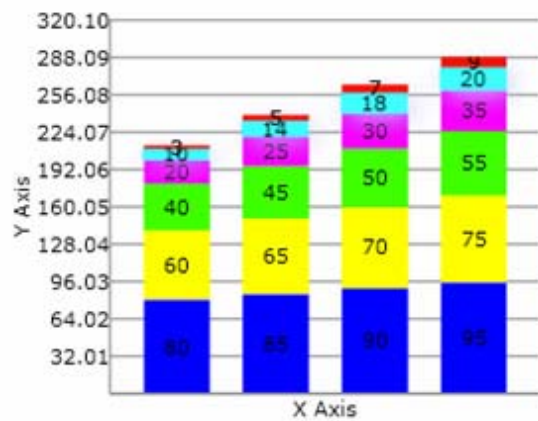
State	Employed	Unemploy	Male	Female
Michigan	4166196	374341	4512781	4782516
Texas	7687338	590269	8433346	8688674
Kentucky	1970934	148125	2195130	2356394
Ohio	4524383	324867	4816445	5164442

Mapping ID	Severity	Acked	Cleared	Raised Time	627	14862394
1	Indeterminate	<input type="checkbox"/>	<input type="checkbox"/>	8/22/2011 3:04:47 PM	3	5878369
2	Warning	<input type="checkbox"/>	<input type="checkbox"/>	8/22/2011 3:04:47 PM	4	2503774
3	Minor	<input type="checkbox"/>	<input type="checkbox"/>	8/22/2011 3:04:47 PM		
4	Major	<input type="checkbox"/>	<input type="checkbox"/>	8/22/2011 3:04:47 PM		
5	Critical	<input type="checkbox"/>	<input type="checkbox"/>	8/22/2011 3:04:47 PM		

Icon	ID	Alarm Color	Tree Label	Network Label
	13f965c5-0f35-48ef	#FF000000	boy1	Sam
	233b4926-c8a0-4cc	#FF000000	boy2	Paul
	8013ae51-7c54-427	#FF000000	girl1	
	cafa8abb-74c1-45c	#FF000000	girl2	
	cb57a4a0-3375-471	#FF000000	girl3	
	d7d3e832-aa04-4af	#FF000000	boy3	Eric

TWaver™

Chart



TM
waver

与TWaver Java的比较

TWaver™

- TWaver .NET与TWaver Java有很多相似之处
- 主要类比较
- 网元属性比较
- 拓扑组件比较

主要类比较

TWaver™

TWaver .NET	TWaver Java	Description
ElementBox	TDataBox	网元容器
AlarmBox	AlarmModel	告警容器
LayerBox	LayerModel	图层容器
Network	TNetwork	拓扑组件
Tree	TTree	树组件
Table	TElementTable	表格组件

网元属性比较

TWaver™

TWaver .NET	TWaver Java	描述
node.Name = "001"	node.setName("001")	属性
node.SetStyle	node.putClientProperty	样式
node.SetClient	node.putUserProperty	业务属性

拓扑图使用比较

TWaver .NET	TWaver Java	Description
AlarmLabelFunction	alarmLabelGenerator	alarm bubble label
VisibleFunction	visibleFilter	visible filter
MovableFunction	movableFilter	movable filter
EditableFunction	elementLabelEditableFilter	label editable filter
LabelFunction	elementLabelGenerator	label text generator
ToolTipFunction	elementToolTipTextGenerator	tooltip text generator
InnerColorFunction	elementBodyColorGenerator	body color generator
OuterColorFunction	elementOutlineColorGenerator	outline color generator
SelectColorFunction	elementSelectColorGenerator	selection color generator
AlarmFillColorFunction	alarmColorGenerator	alarm bubble color



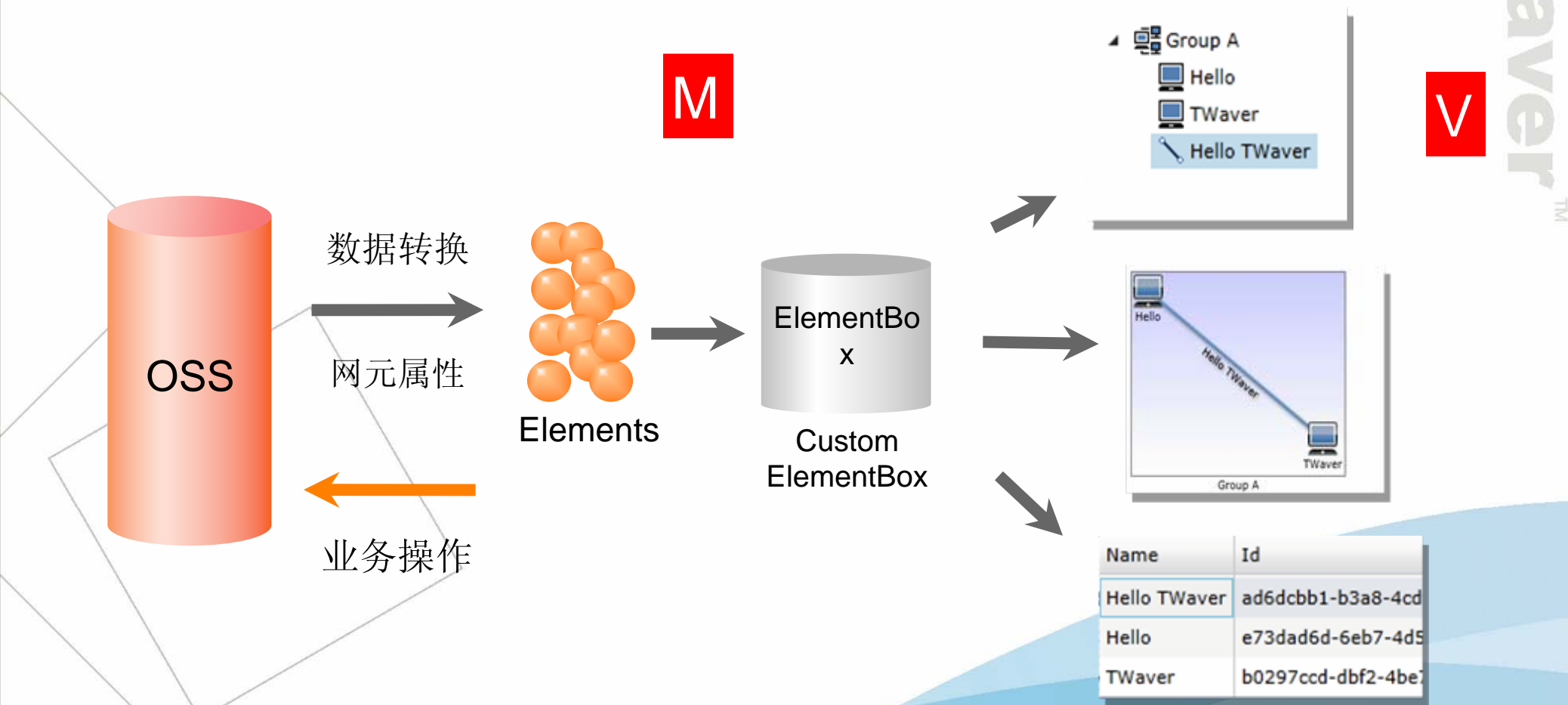
- 论坛- twaver.servasoft.com/forum
- Email - tw-service@servasoft.com

TWaver .NET 核心组件

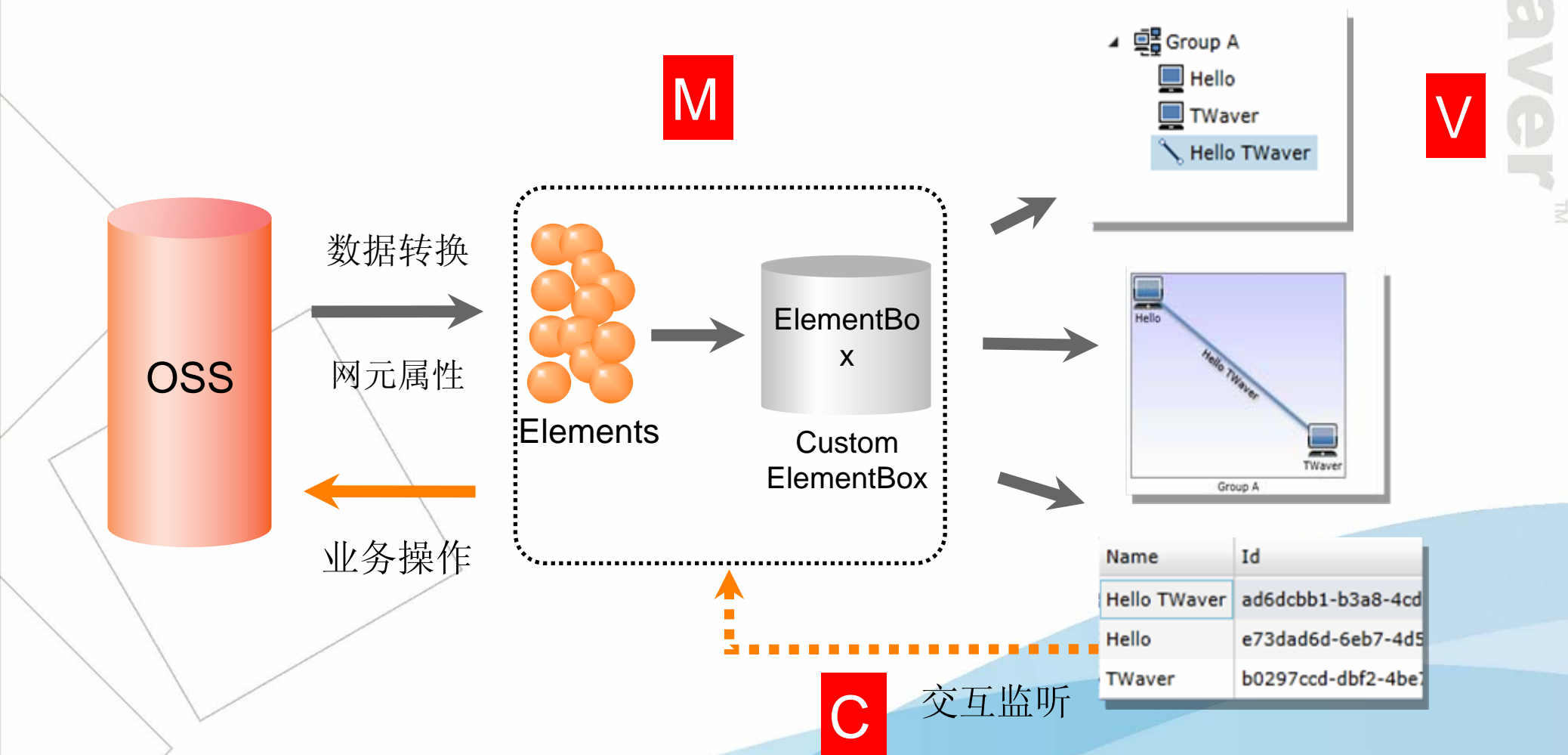
TWaver™

- 开发流程
- 使用DataBox & ElementBox
- 预定义网元类型
- 使用Network

TWaver .NET 开发流程



TWaver .NET 开发流程



TWaver .NET 开发流程

TWaver™

```
ElementBox box = new ElementBox();  
//data acquisitionArray  
devices = GetDevicesFromOSS();  
Array relationships = GetDevicesRelationshipFromOSS();  
//data translate  
TranslateToTWaverNode(box, devices, relationships);  
Network network = new Network(box);  
this.LayoutRoot.Children.Add(network);  
AutoLayouter layouter = new AutoLayouter(network);  
layouter.IsAnimated = false;  
layouter.DoLayout(Constants.LAYOUT_SYMMETRY);  
//add interaction  
network.Interaction += (InteractionEvent evt) =>{  
    if (evt.Kind != InteractionEvent.DOUBLE_CLICK_ELEMENT) {  
        return;  
    }  
    IElement element = evt.Element;  
    ShowInputDialog("Rename", element.Name, (string result) => {  
        element.Name = result;  
    });  
};
```

TWaver .NET 开发流程

```
private Array GetDevicesFromOSS() {
    return new object[] { new { name = "R1", type = "router" },
        new { name = "R2", type = "router" },
        new { name = "S1", type = "device" },
        new { name = "S2", type = "device" },
        new { name = "S3", type = "device" },
        new { name = "S4", type = "device" },
        new { name = "S5", type = "device" } };
}

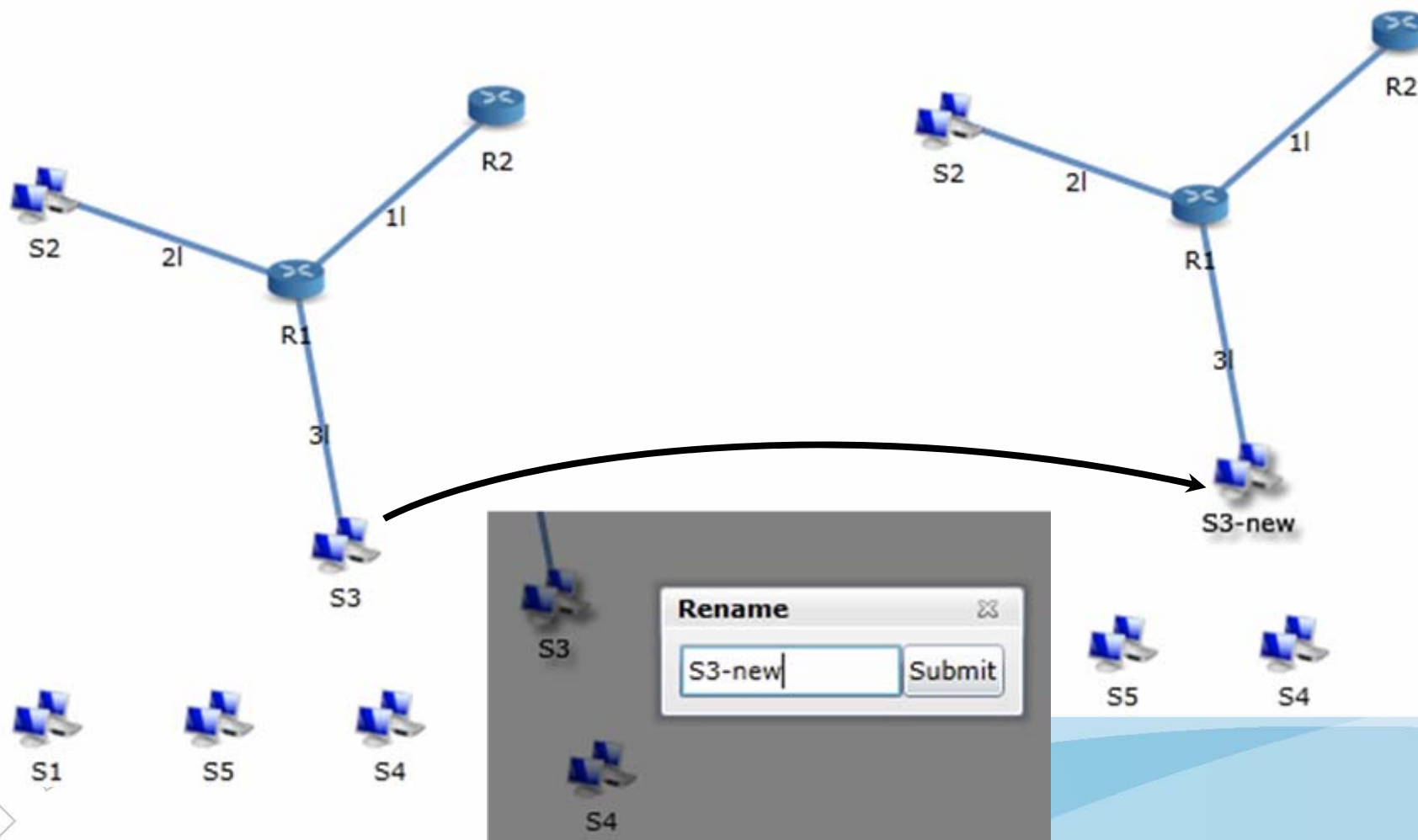
private Array GetDevicesRelationshipFromOSS() {
    return new object[] { new { name = "1I", from = "R1", to = "R2" },
        new { name = "2I", from = "R1", to = "S2" },
        new { name = "3I", from = "R1", to = "S3" } };
}

private void TranslateToTWaverNode(ElementBox box, Array devices, Array relationships) {
    foreach (dynamic device in devices) {
        Node node = new Node(device.name);
        node.Name = device.name;
        node.Image = device.type;
        box.Add(node);
    }
    foreach (dynamic relationship in relationships) {
        Link link = new Link((Node)box.GetDataByID(relationship.from), (Node)box.GetDataByID(relationship.to));
        link.Name = relationship.name;
        box.Add(link);
    }
}
```

TWaver™

TWaver .NET 开发流程

TWaver™

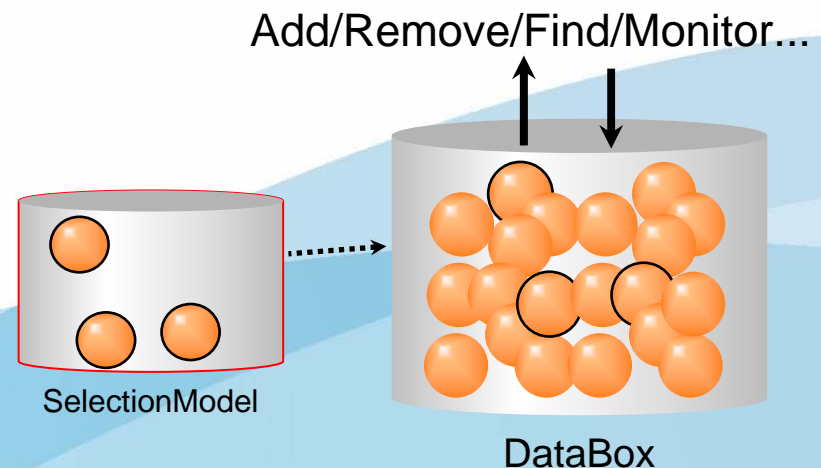
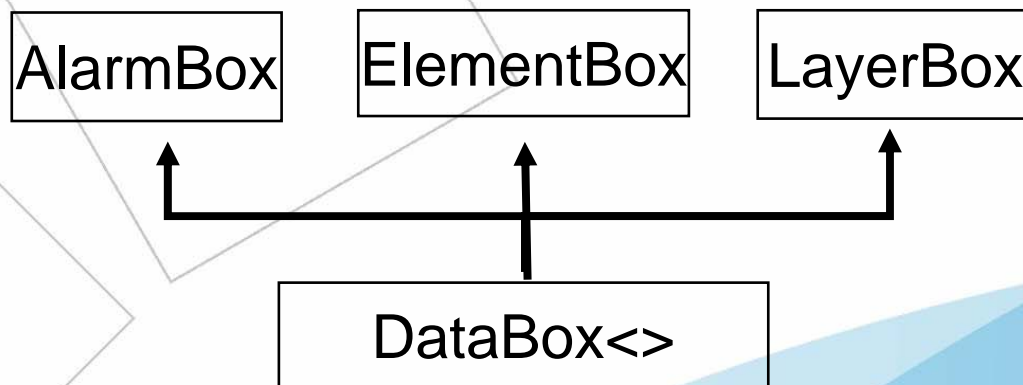


使用DataBox

DataBox – 数据管理容器

- 提供基本的数据操作，如：add, delete, edit
- 监听数据元素的属性变化事件
- 是ElementBox, AlarmBox, LayerBox的基类
- 包含选中模型

DataBox#**public** SelectionModel SelectionModel



基本操作

- Add data

public virtual void Add(TData data)

public virtual void Add(TData data, int index)

- Delete data

public virtual void Remove(TData data)

public virtual void RemoveSelection()

public virtual void RemoveByID(Object id)

- Clear data

public virtual void Clear()

Twaver™

获取元素

TWaver™

- Get data

public virtual TData GetDataByID(Object id)

public virtual IList<TData> Datas()

public virtual IList<TData> ToDatas(Predicate<TData> match)

public virtual IList<TData> Roots()

public virtual IList<TData> GetSiblings(TData data)

- Traverse data

public virtual void ForEach(Action<TData> callbackFunction):

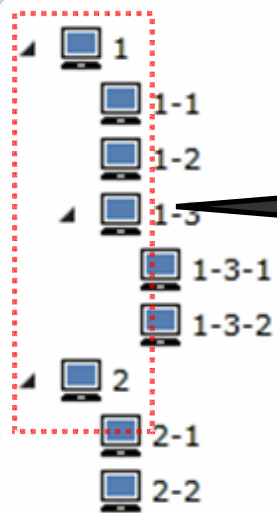
public virtual void ForEachByDepthFirst(Action<TData> callbackFunction,
TData data = **null**):

public virtual void ForEachByBreadthFirst(Action<TData>
callbackFunction, TData data = **null**)

元素的层次关系

- 元素的层次关系由父子关系决定
- DataBox的Move***方法可用于调节元素的顺序关系

TWaver™



node 1 ,2 at the root
level (Roots)

- MoveDown(TData)
- MoveSelectionDown()
- MoveSelectionDown(TWaver.SelectionModel<TData>)
- MoveSelectionToBottom()
- MoveSelectionToBottom(TWaver.SelectionModel<TData>)
- MoveSelectionToTop()
- MoveSelectionToTop(TWaver.SelectionModel<TData>)
- MoveSelectionUp()
- MoveSelectionUp(TWaver.SelectionModel<TData>)
- MoveTo(TData, int)
- MoveToBottom(TData)
- MoveToTop(TData)
- MoveUp(TData)

Quick Finder

QuickFinder:

根据元素的属性实现快速查找，如查找所有名称为“PC”的所有元素
示例：

```
QuickFinder<IData> finder = new QuickFinder<IData>(box, "age");  
IList<IData> elements = finder.Find("PC");
```

Create a quick finder:

```
public QuickFinder(DataBox<TData> dataBox, string propertyName,  
    string propertyType,  
    Func<TData, object> valueFunction,  
    Predicate<TData> filterFunction)
```

Quick finder types:

Consts.PROPERTY_TYPE_ACCESSOR — find by accessor

propertyConsts.PROPERTY_TYPE_CLIENT — find by client

propertyConsts.PROPERTY_TYPE_STYLE — find by style property

Quick Finder 示例

```
DataBox<IData> box = new DataBox<IData>();
for (int i = 0; i < 30; i++)
{
    Node node = new Node();
    node.Name = "Node-" + i;
    node.SetClient("age", i % 10);
    box.Add(node);
}
QuickFinder<IData> finder = new QuickFinder<IData>(box, "age",
Consts.PROPERTY_TYPE_CLIENT);
IList<IData> elements = finder.Find(8);
foreach (IData node in elements)
{
    Trace(node.Name + "'s age is " + node.GetClient("age"));
}
```

trace:

Node-8's age is 8

Node-18's age is 8

Node-28's age is 8

Twaver™

DataBox中的监听器

TWaver™

Add listener - DataBox.DataPropertyChange += listener

Delete listener - DataBox.DataPropertyChange -= listener

box.DataPropertyChange += new

```
EventListener<PropertyChangeEvent<IData>>((PropertyChangeEvent<IData> evt) =>{  
    MessageBox.Show(evt.Source.Name + "'s property changed");  
});
```

Listener	Description
DataBoxChange	detect element's add, remove, clear
PriorDataBoxChange	before dataBox change
PropertyChange	detect dataBox's property change
DataPropertyChange	detect element's property change
HierarchyChange	detect dataBox's sequence change

监听器示例

```
DataBox<IData> box = new DataBox<IData>();
```

```
box.DataPropertyChange += new  
EventListener<PropertyChangeEvent<IData>>((PropertyChangeEvent<IData> evt) =>{  
    Trace(evt.Source.Name + "'s property changed");
```

```
});  
box.DataBoxChange += new  
EventListener<DataBoxChangeEvent<IData>>((DataBoxChangeEvent<IData> evt) => {  
    Trace("data " + evt.Kind + ": " + evt.Data.Name);  
});
```

```
Data data = new Data();  
data.Name = "001";  
//add data  
box.Add(data);  
//modified data property  
data.Name = "002";  
//remove data  
box.Remove(data);
```

```
output:  
data add: 001  
002's property changed  
data remove: 002
```

DataBox的导入导出

TM
waver

- DataBox可以导入导出成xml
- 导出的XML文件包含DataBox自身的属性以及其内数据元素的信息
- 使用XMLSerializer实现导入导出

DataBox的导入导出

TWaver™

● XMLSerializer - import & export

- **public** XMLSerializer(DataBox<TData> dataBox, SerializationSettings settings, Predicate<TData> filterFunction))
- **public** String Serialize():
- **public void** Deserialize(string xmlString, IData rootParent)

● SerializationSettings - import & export setting

- **public void** RegisterProperty(string property, string type, bool cdata)
- **public void** RegisterStyle(string styleProp, string type, bool cdata)
- **public void** RegisterClient(string clientProp, string type, bool cdata)

DataBox 导出示例

```
ElementBox box = network.ElementBox;
```

```
SubNetwork parent = new SubNetwork();  
parent.Name = "Parent";  
network.ElementBox.Add(parent);
```

```
Node node1 = new Node();  
node1.Location = new Point(50, 50);  
node1.Name = "Node1";  
node1.SetClient("age", 27);  
node1.Parent = parent;  
network.ElementBox.Add(node1);
```

```
SerializationSettings settings = new SerializationSettings();  
settings.RegisterProperty("ID", null, false);  
settings.RegisterClient("age", Consts.TYPE_INT, false);  
settings.RegisterProperty("name", Consts.TYPE_STRING, true);  
settings.RegisterProperty("parent", Consts.TYPE_DATA, false);  
serializer = new XMLSerializer<IElement>(box, settings);
```

```
textBox.Text = serializer.Serialize();
```

```
<?xml version="1.0" encoding="utf-16"?>  
<TWaver V="2.0" P="SILVERLIGHT">  
  <DataBox Type="TWaver.ElementBox">  
    <LayerBox>  
      <Layer Name="default" IsVisible="True"  
IsEditable="True" IsMovable="True" />  
    </LayerBox>  
  </DataBox>  
  <Data Type="TWaver.SubNetwork" Ref="0">  
    <P N="Name"><![CDATA[Parent]]></P>  
  </Data>  
  <Data Type="TWaver.Node" Ref="1">  
    <C N="age">27</C>  
    <P N="Name"><![CDATA[Node1]]></P>  
    <P N="Parent" Ref="0" />  
    <P N="Location" X="50" Y="50" />  
  </Data>  
</TWaver>
```

DataBox导入示例

TWaver™

```
SerializationSettings settings = new SerializationSettings();  
settings.RegisterProperty("ID", null, false);  
settings.RegisterClient("age", Consts.TYPE_INT, false);  
settings.RegisterProperty("name", Consts.TYPE_STRING, true);  
settings.RegisterProperty("parent", Consts.TYPE_DATA, false);  
serializer = new XMLSerializer<IElement>(box, settings);
```

```
textBox.Text = serializer.Serialize();
```

```
string xml = textBox.Text;  
box.Clear();  
serializer.Deserialize(xml);
```

```
textBox.Text = "box count: " + box.Count;  
textBox.Text += "\nthe first node's name: " + box.Datas[0].Name;  
textBox.Text += "\nthe second node's name: " + box.Datas[1].Name;
```

```
output:  
box count: 2  
the first node's name: Parent  
the second node's name: Node1
```

使用ElementBox

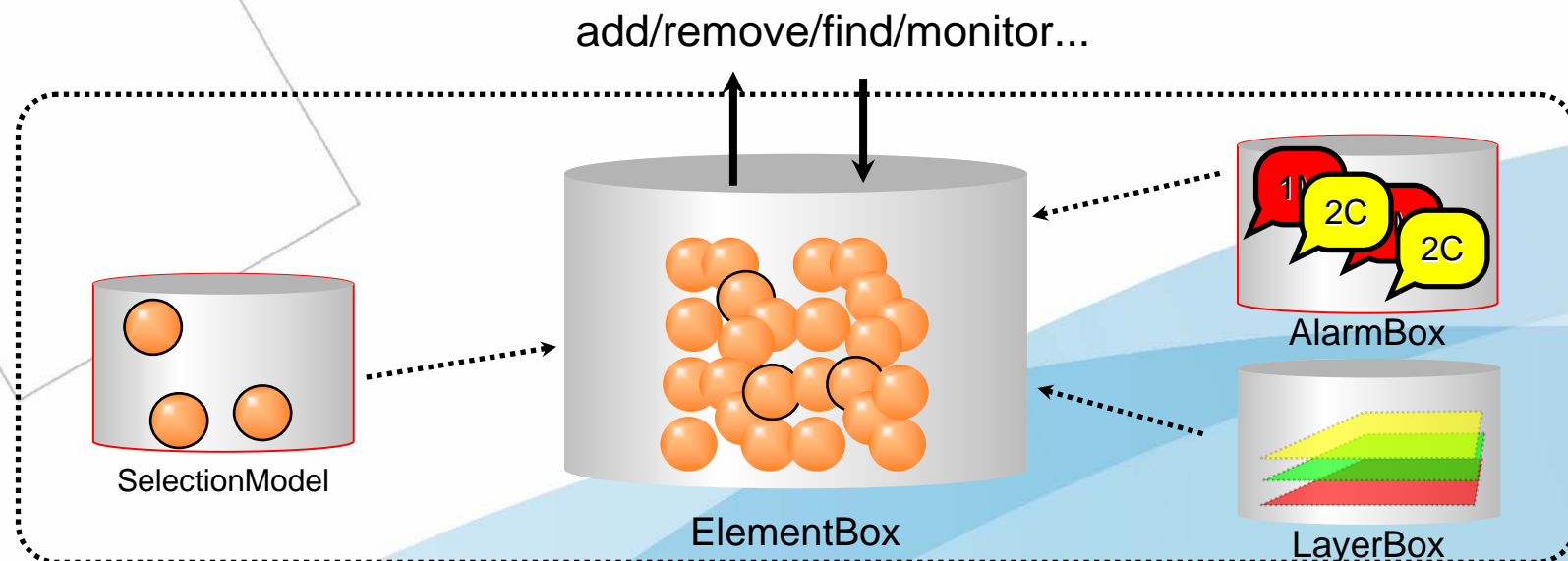
ElementBox – 用于管理拓扑网元的数据容器

继承于DataBox<IElement>, 保留了DataBox的功能, 增加了
图层管理和告警管理

ElementBox#

public LayerBox LayerBox

public AlarmBox AlarmBox



使用ElementBox

- 增加了LayerBox、AlarmBox

```
public LayerBox LayerBox
```

```
public AlarmBox AlarmBox
```

- 按图层遍历网元

```
public void ForEachByLayer(Action<IElement> callbackFunction, ILayer layer)
```

```
public void ForEachByLayerReverse(Action<IElement> callbackFunction, ILayer layer)
```

- 增加了图层变化监听

```
public event EventListener<IndexChangeEvent> IndexChange
```

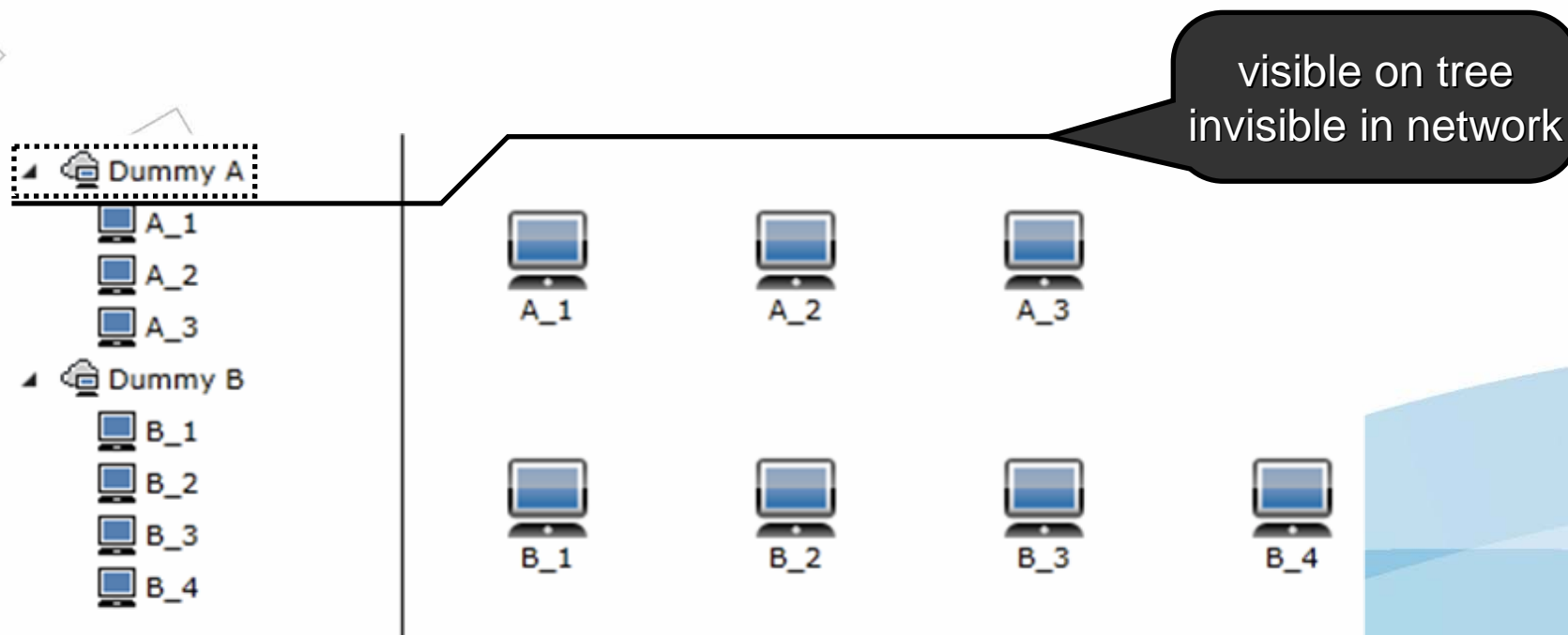
Elements

Data

- | -- Alarm
- | -- Layer
- | -- Element
 - | -- Dummy
 - | -- Node
 - | -- Follower
 - | -- Group
 - | -- Grid
 - | -- ShapeNode
 - | -- Bus
 - | -- SubNetwork
 - | -- Link
 - | -- ShapeLink
 - | -- LinkSubNetwork

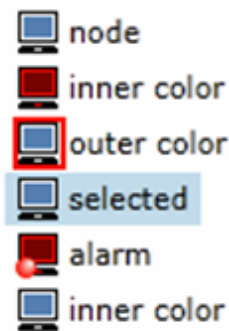
Dummy

Dummy在Network中不可见，在树、表格中都可见
Dummy对象用来组织树结构，不影响Network



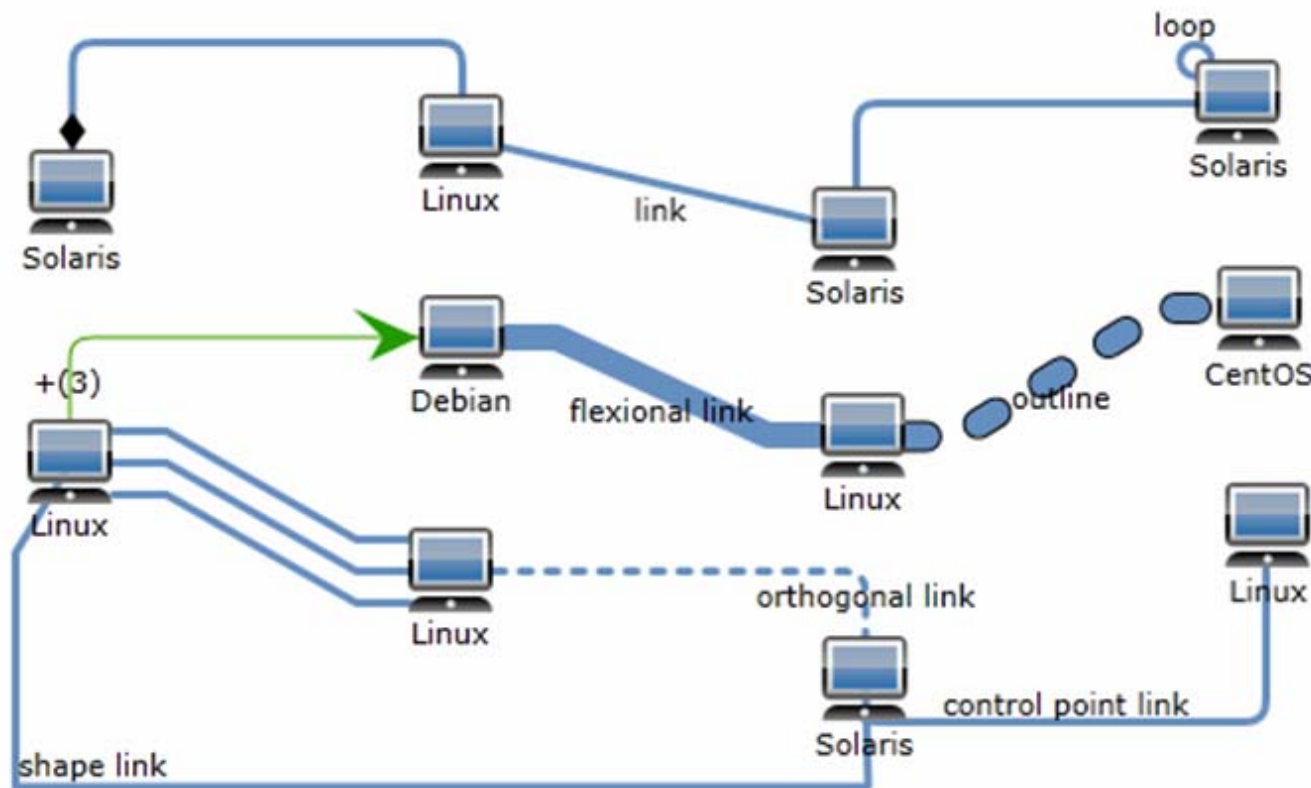
Node

- 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，虚线效果等

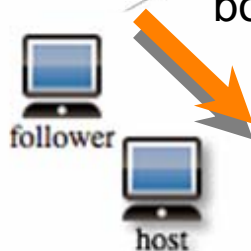


Follower

- 跟随者，可设置host节点，host移动，follower网元跟随移动

```
Follower node = new Follower();  
node.Location = new Point(100, 100);  
node.Name = "follower";  
box.Add(node);
```

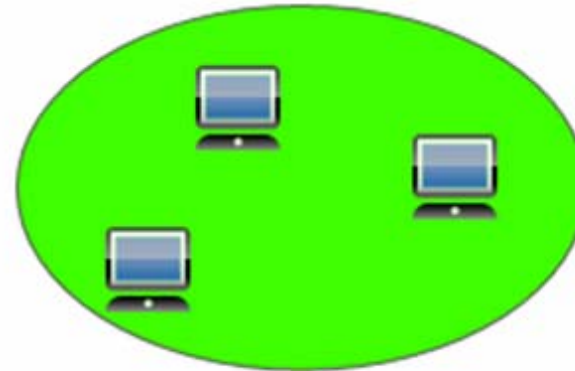
```
Node host = new Node();  
host.Location = new Point(140, 140);  
node.Host = host;  
host.Name = "host";  
box.Add(host);
```



host move, and follower move, too.

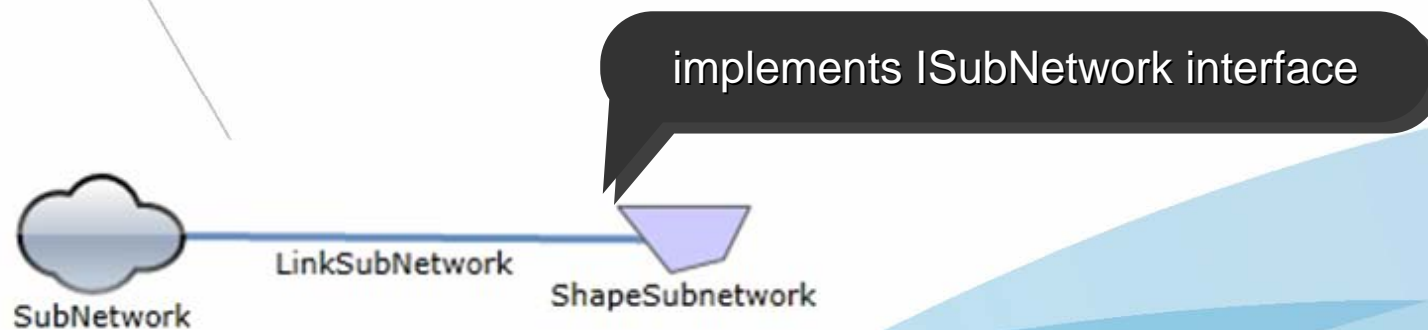
Group

- 分组，可以设置椭圆，矩形等形状，支持渐变填充，边框颜色等渲染效果



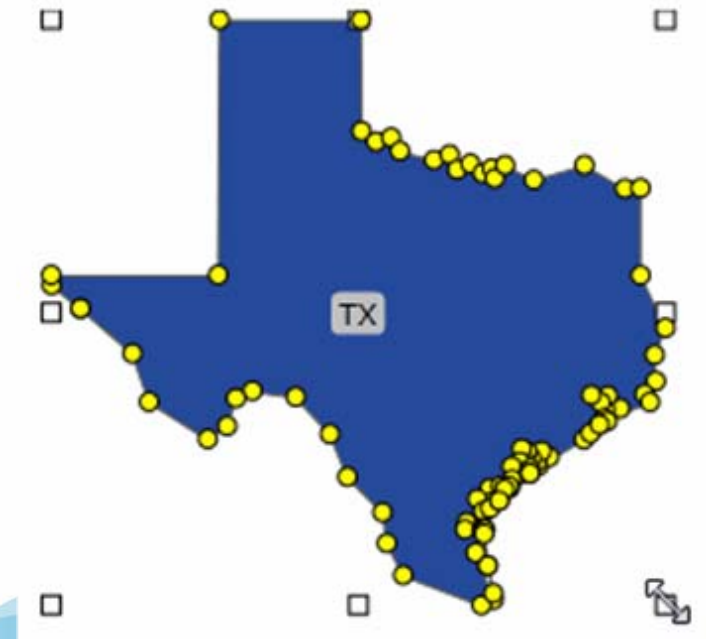
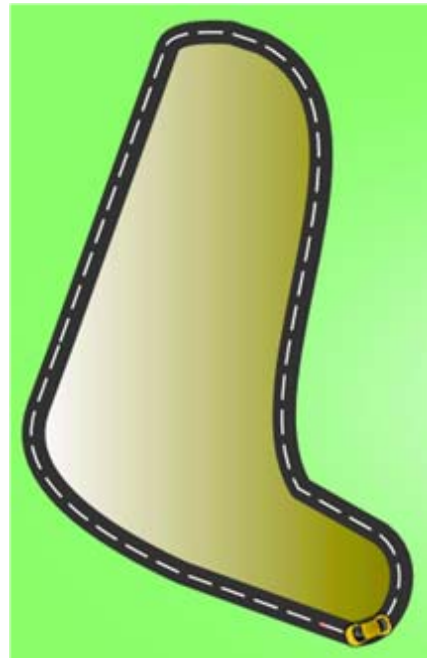
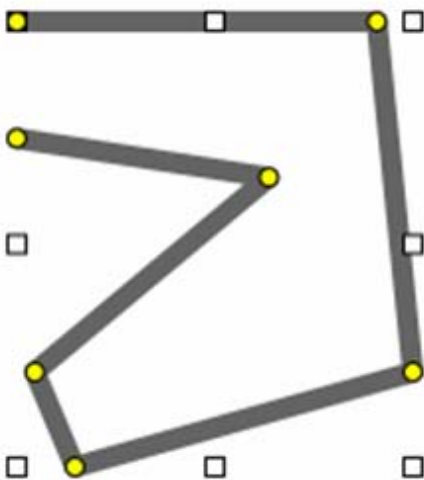
SubNetwork

- 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



ShapeNode

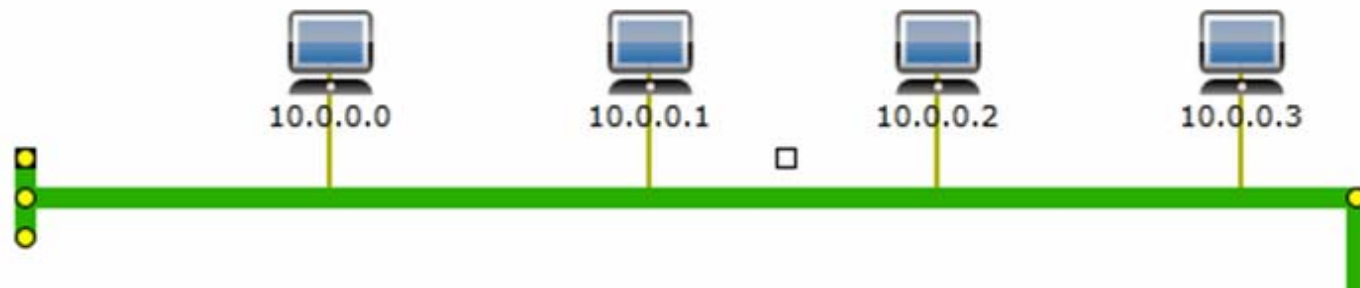
- 多边形，由一系列控制点围成的多边形或线段



Bus

TWaver™

- 继承于ShapeNode，通过一组控制点围成的线段，与其相连的Link呈垂直链接分布



Grid

- 网格，行列分布，可以组合使用，制作设备面板图

```
TWaver.Grid grid = new TWaver.Grid();  
grid.Name = "Grid B";  
grid.SetStyle(Styles.GRID_BORDER, 5);  
grid.SetStyle(Styles.GRID_COLUMN_COUNT, 4);  
grid.SetStyle(Styles.GRID_COLUMN_PERCENTS, new double[]  
{ 0.15, 0.2, 0.25, 0.4 });  
grid.SetStyle(Styles.GRID_ROW_COUNT, 2);  
grid.SetStyle(Styles.GRID_ROW_PERCENTS, new double[]  
{ 0.6, 0.4 });  
grid.SetStyle(Styles.GRID_FILL_COLOR,  
Utils.CreateColor(0xFFCC6600));  
grid.SetStyle(Styles.GRID_DEEP, 11);  
grid.SetStyle(Styles.GRID_CELL_DEEP, -4);  
grid.SetStyle(Styles.GRID_PADDING, 0);  
grid.Width = 250;  
grid.Height = 106;
```



Grid A



Grid B



网元属性

TWaver™

twaver.Consts#

```
public const string PROPERTY_TYPE_ACCESSOR =  
"accessor"; public const string PROPERTY_TYPE_CLIENT = "client";  
public const string PROPERTY_TYPE_STYLE = "style";
```

Element

accessor - name, id ...

node.Name = "001";

style - style properties

node.SetStyle(Styles.INNER_COLOR,
Utils.CreateColor(0xFFCC6600));

client - client properties

node.SetClient("age", 27);

Network的使用

TM
waver

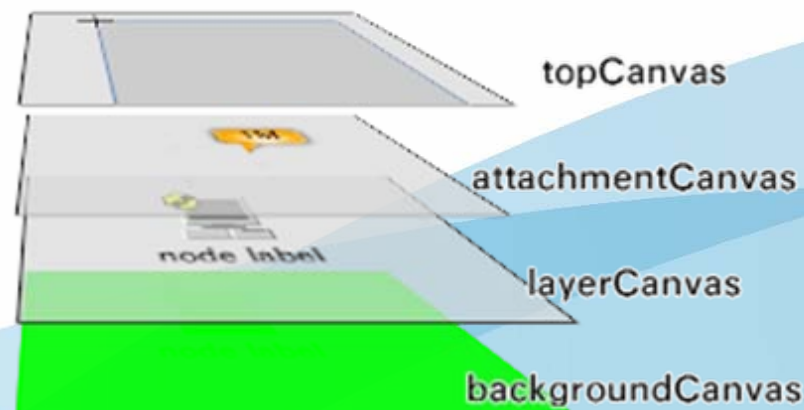
- Network的层次结构
- Network添加背景
- Network的交互处理
- Network的过滤器
- Network样式规则函数
- Network自动布局

Network的层次结构

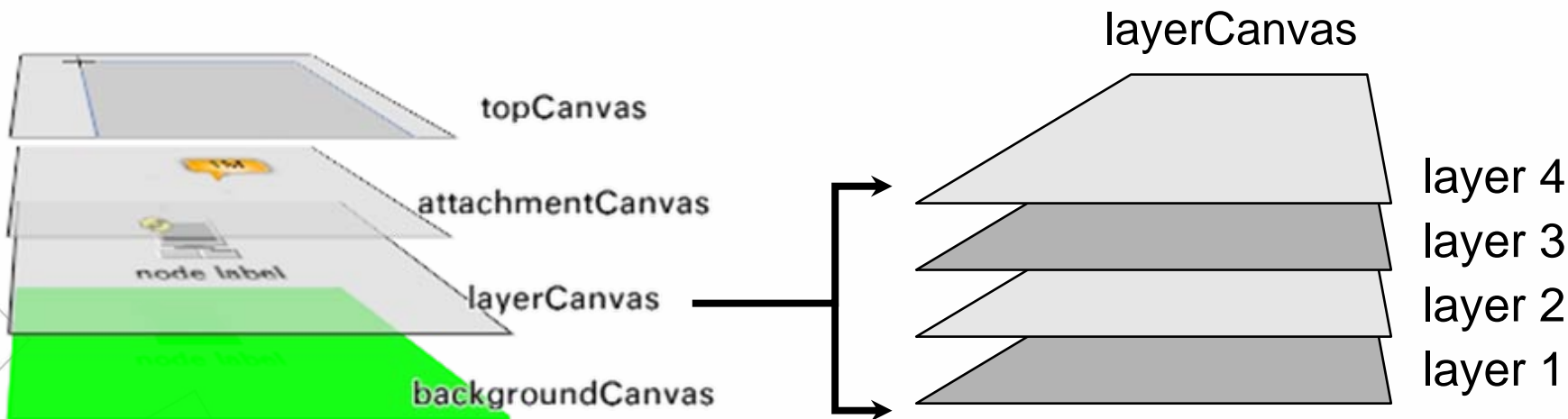
ScrollView

NetworkCanvas

- RootCanvas — 主画布，所有拓扑组件都在其中
- TopCanvas — 顶层画布，可用于绘制交互框选框，调整大小时的拖拽块
- AttachmentCanvas — 放置顶层附件组件
- LayerCanvas — 所有网元视图所在层
- Layer n
- Layer ...
- Default layer
- BottomCanvas — 底层画布，可用于在背景图之上绘制底纹
- BackgroundCanvas — 背景画布，包括颜色，图片等背景



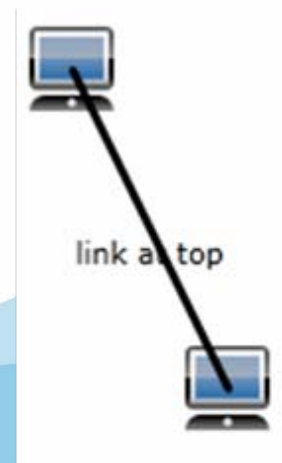
图层管理



```
ElementBox box = network.ElementBox;
LayerBox layerBox = box.LayerBox;
```

```
Layer layerTop = new Layer("top", "top");
layerBox.Add(layerTop, layerBox.Count);
```

```
Node node = new Node();
box.Add(node);
Node node2 = new Node();
box.Add(node2);
Link link = new Link(node, node2);
link.LayerID = layerTop.ID;
link.Name = "link at top";
link.SetStyle(Styles.LINK_COLOR, Utils.CreateColor(0xFF000000));
box.Add(link);
```



Twaver™

添加背景

TWaver™

- 可以对**ElementBox**和**Subnetwork**设置背景，支持栅格图片，颜色，颜色渐变等；

```
Utils.RegisterPNGImage("background", new  
Uri("/TWaverPPT;component/images/shanghai.png", UriKind.Relative));  
box.SetStyle(Styles.BACKGROUND_TYPE, Consts.BACKGROUND_TYPE_IMAGE);  
box.SetStyle(Styles.BACKGROUND_IMAGE, "background");
```

- 也可以使用**.NET**组件默认支持的背景样式，如：

```
network.Background = new SolidColorBrush(Color.FromArgb(255, 0,  
255, 0));
```

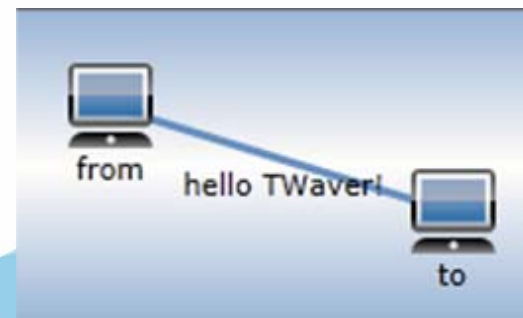
Background 示例

```
Network network = new Network();network.Background = new  
SolidColorBrush(Color.FromArgb(255, 0, 255, 0));
```

```
SubNetwork subnetworkA = AddSubnetwork(box, "subnetwork A");  
subnetworkA.Location = new Point(100, 100);  
subnetworkA.SetStyle(Styles.BACKGROUND_COLOR, Utils.CreateColor(0xFF9CB5D8));  
subnetworkA.SetStyle(Styles.BACKGROUND_GRADIENT_COLOR,  
Utils.CreateColor(0xFFFFFFFF));  
subnetworkA.SetStyle(Styles.BACKGROUND_GRADIENT,  
Consts.GRADIENT_SPREAD_VERTICAL);
```



drop in



Network交互模式

- **InteractionHandler** - 单个监听器
- **Interaction Mode** - 一组监听器组成一种交互模式

Network#

```
public IList<InteractionHandler> InteractionHandlers
```

- **Network**预定义了多种交互模式，如默认模式，编辑模式，缩放模式...

Network#

```
public void SetDefaultInteractionHandlers(bool lazyMode)
```

```
public void SetEditInteractionHandlers(bool lazyMode)
```

```
public void SetCreateLinkInteractionHandlers(Type type)
```

...

交互定制示例

Interaction handler class

```
public class HighlightInputHandler : BasicInteractionHandler {  
    public HighlightInputHandler(Network network) : base(network) { }  
    override public void InstallListeners(){  
        this.NETwork.NETworkCanvas.MouseMove += HandleMouseMove;  
    }  
    override public void UninstallListeners(){  
        this.NETwork.NETworkCanvas.MouseMove -= HandleMouseMove;  
    }  
    ...  
}
```

Setting interaction mode

```
network.InteractionHandlers = new InteractionHandler[] {  
    new SelectInteractionHandler(network),  
    new MoveInteractionHandler(network, false),  
    new DefaultInteractionHandler(network),  
    new HighlightInputHandler(network) };
```



mouse over, node
highlight

TWaver™

```
public class HighlightInputHandler : BasicInteractionHandler {
    public HighlightInputHandler(Network network) : base(network){ }
    override public void InstallListeners(){
        this.NETwork.NETworkCanvas.MouseMove += HandleMouseMove;
    }
    override public void UninstallListeners(){
        this.NETwork.NETworkCanvas.MouseMove -= HandleMouseMove;
    }
    private void HandleMouseMove(object sender, MouseEventArgs e){
        IElement element = network.GetElement(e);
        if (element != null){
            highlight(element);
        } else{
            reset();
        }
    }
    private IElement highlightElement;
    private Color? highLightColor = Utils.CreateColor(0xFFFF8888);
    private Color? oldColor = null;

    private void highlight(IElement element){
        if (element == null || element == highlightElement) {
            return;
        }
        reset();
        highlightElement = element;
        oldColor = (Color?)element.GetStyle(Styles.INNER_COLOR);
        network.Cursor = Cursors.Hand;
        element.SetStyle(Styles.INNER_COLOR, highLightColor);
    }
    private void reset(){
        if (highlightElement != null) {
            if (oldColor != null) {
                highlightElement.SetStyle(Styles.INNER_COLOR, oldColor);
            } else{
                highlightElement.SetStyle(Styles.INNER_COLOR, null);
            }
            highlightElement = null;
            network.Cursor = null;
        }
    }
}
```

Network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

Network#

//可见过滤器

```
public Predicate<IElement> VisibleFunction
```

//可移动过滤器

```
public Predicate<IElement> MovableFunction
```

//可编辑过滤器

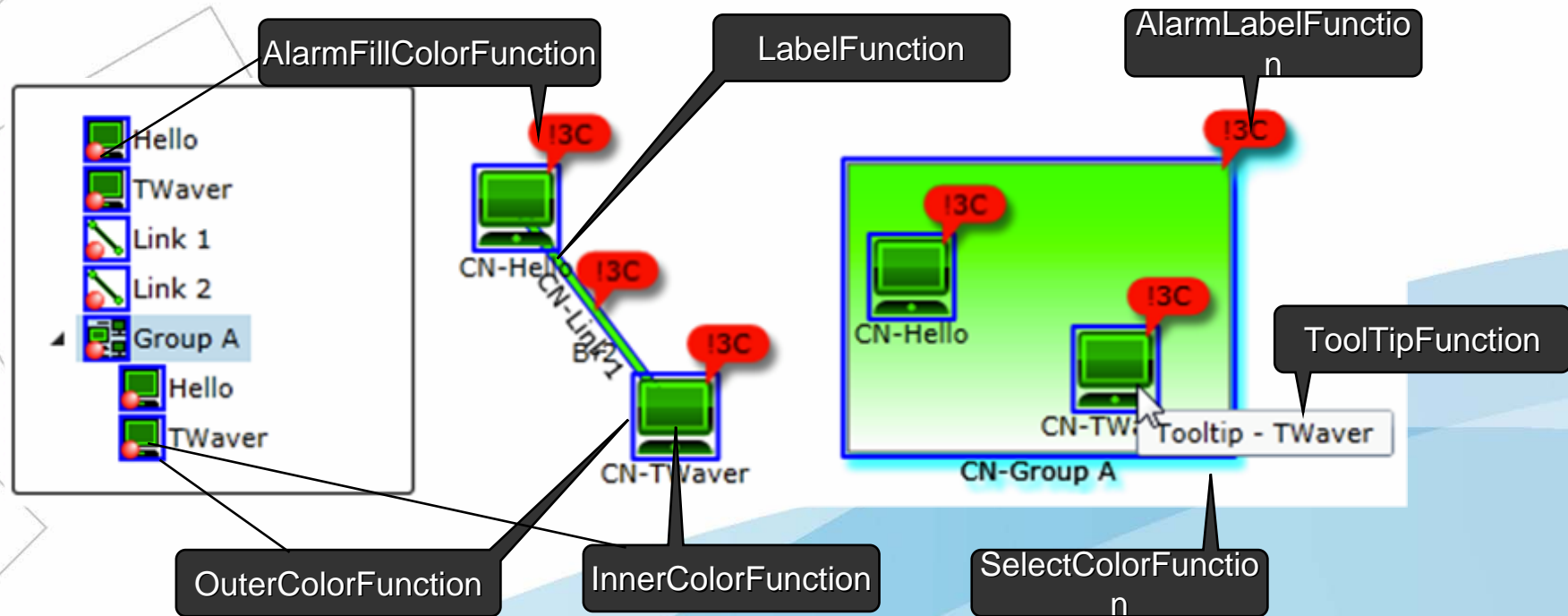
```
public Predicate<IElement> EditableFunction
```

```
network.VisibleFunction = (IElement element) =>{  
    return !(element is Link);  
};
```

Network样式规则函数

***Function

样式规则函数用于全局设置拓扑图中网元显示样式，而不用修改每个网元的属性

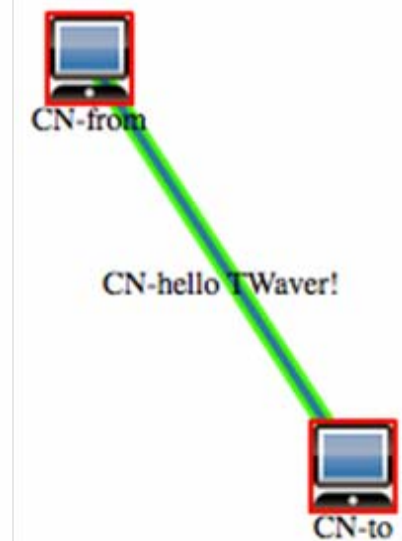


Style Function 示例

对网元文本标签增加前缀，定制外边框颜色规则的示例：

```
network.LabelFunction = (IElement element) => {  
    return "CN-" + element.Name;  
};
```

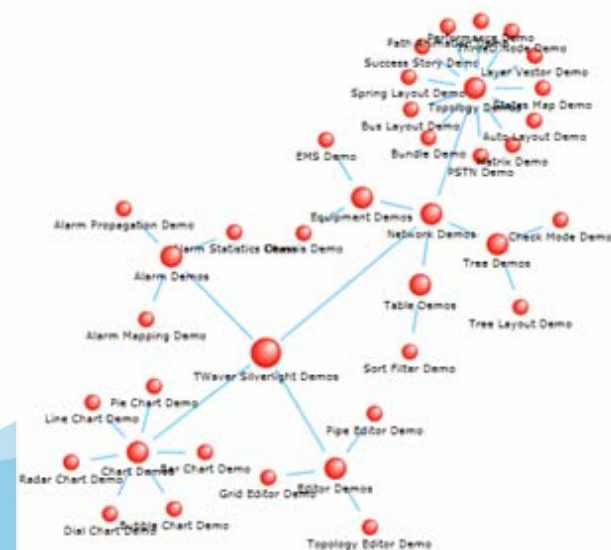
```
network.OuterColorFunction = (IData data) => {  
    return Color.FromArgb(255, 0, 0, 255);  
};
```



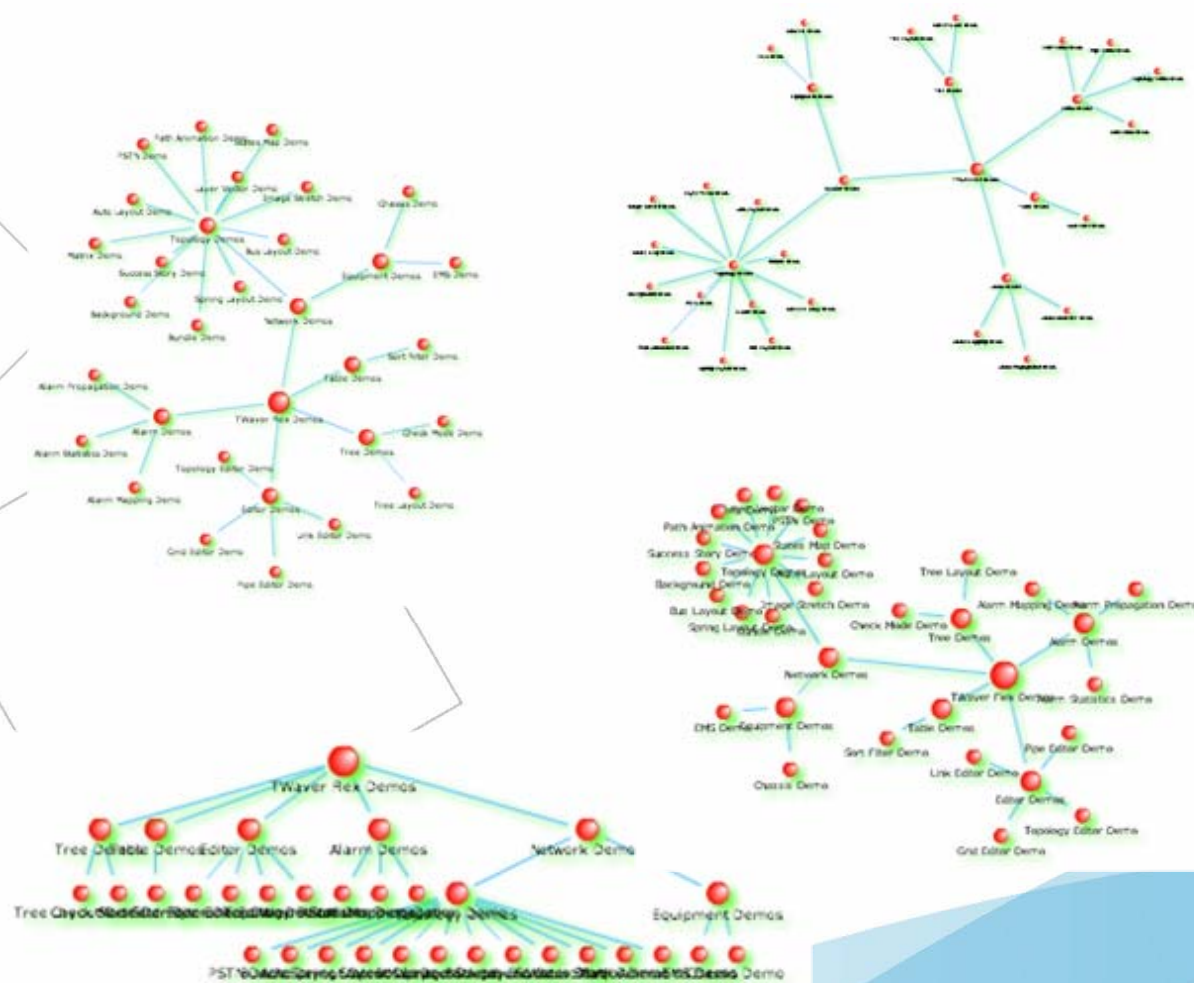
Network自动布局

TWaver™

- 自动对网元连线定位，使其分散开，分布呈现好的效果


TWaver icon

默认自动布局





- 论坛: twaver.servasoft.com/forum
- Email - tw-service@servasoft.com

通用组件的使用

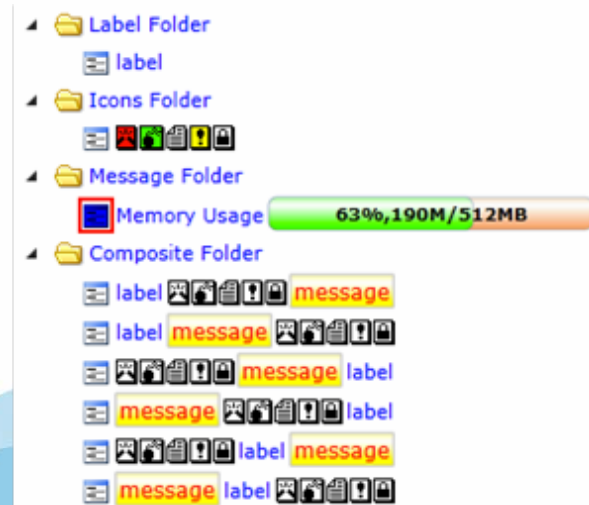
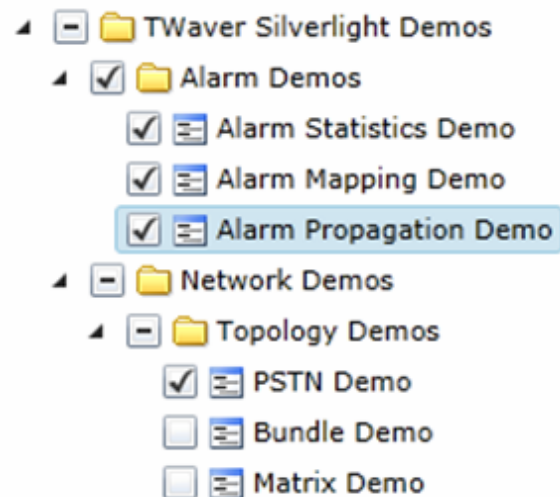
- 通用组件的使用
 - Tree / Table / Chart
- 告警的使用
 - 告警对象 / 告警级别 / 告警容器 / 告警状态 / 告警统计 / 告警呈现

Tree的使用

TWaver™

twaver.controls.Tree -用于展示DataBox中元素父子层次关系的组件

```
Tree<IElement> tree = new Tree<IElement>(box);
```



Tree的创建

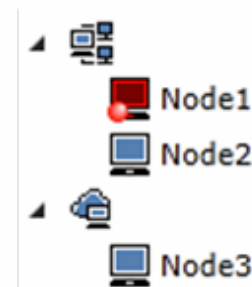
```
DataBox<Data> box = new DataBox<Data>();
```

```
Group group = new Group();  
box.Add(group);  
SubNetwork subnetwork = new SubNetwork();  
box.Add(subnetwork);
```

```
Node node = new Node();  
node.Name = "Node1";  
node.Parent = group;  
node.AlarmState.IncreaseNewAlarm(AlarmSeverity.CRITICAL);  
box.Add(node);  
node = new Node();  
node.Name = "Node2";  
node.Parent = group;  
box.Add(node);
```

```
node = new Node();  
node.Name = "Node3";  
node.Parent = subnetwork;  
box.Add(node);
```

```
tree = new TWaver.Controls.Tree<Data>();  
this.LayoutRoot.Children.Add(tree);  
tree.DataBox = box;  
tree.ExpandAll();
```



网元的父子关系决定树的层次关系

定制树节点

- Icon

`data.Icon = "iconName";`

- Label

`data.Name = "001";`

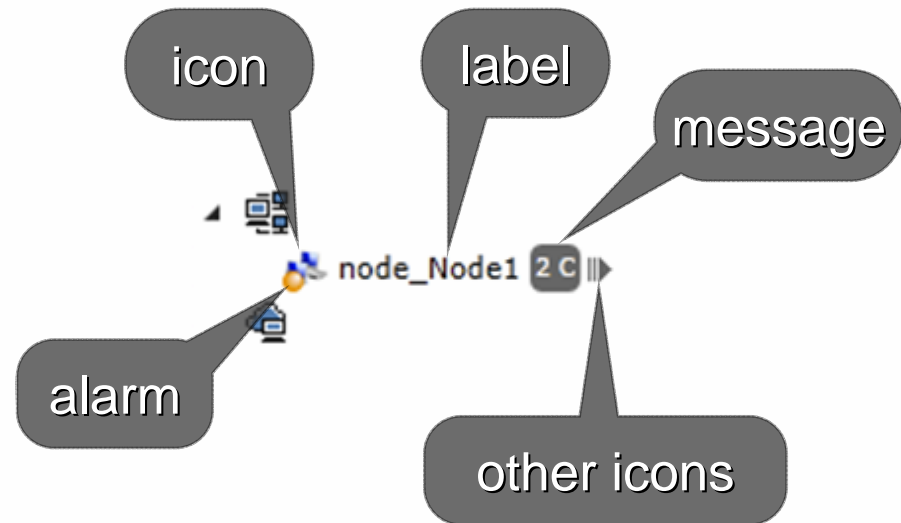
`element.SetStyle(Styles.TREE_LABEL, "002");`

- Other styles

`element.SetStyle(Styles.TREE_LABEL_***, ***);`

`element.SetStyle(Style.TREE_MESSAGE_***, ***);`

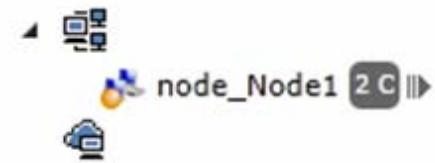
`element.SetStyle(Style.TREE_ICONS_***, ***);`



树节点样式示例

```
ImageSource image = new BitmapImage(new  
Uri("/TWaverPPT;component/images/out.png", UriKind.Relative));  
Utils.RegisterImageByImageSource("outIcon", image, 9, 9);  
Utils.RegisterPNGImage("deviceIcon", new  
Uri("/TWaverPPT;component/images/device.png", UriKind.Relative));  
...
```

```
Node node = new Node();  
node.Name = "Node1";  
node.Icon = "deviceIcon";  
node.SetStyle(Styles.TREE_LABEL, "node_" + node.Name);  
node.SetStyle(Styles.TREE_ICONS_NAMES, new string[] { "outIcon" });  
node.SetStyle(Styles.TREE_LAYOUT_GAP, 3);  
node.SetStyle(Styles.TREE_MESSAGE, "2 C");  
node.SetStyle(Styles.TREE_MESSAGE_FILL_COLOR, Utils.CreateColor(0xFF666666));  
node.SetStyle(Styles.TREE_MESSAGE_GRADIENT, Consts.GRADIENT_NONE);  
node.SetStyle(Styles.TREE_MESSAGE_SIZE, 9);  
node.SetStyle(Styles.TREE_MESSAGE_COLOR, Utils.CreateColor(0xFFFFFFFF));  
node.SetStyle(Styles.TREE_LAYOUT,  
Consts.TREE_LAYOUT_LABEL_MESSAGE_ICONS);  
node.SetStyle(Styles.TREE_ALARM_FILL_COLOR, Utils.CreateColor(0xFFE08000));
```



树节点层次与顺序

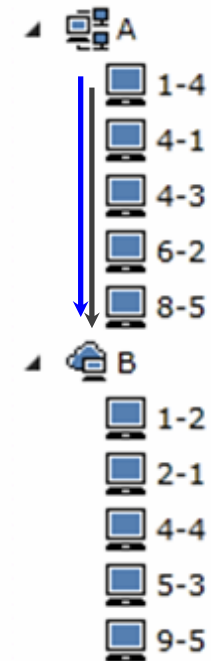
TM
waver

- 层次按网元父子关系
`node.Parent = parent;`
`node.AddChild(child);`
- 同层节点顺序
`box.Move**(node);`
- 可以设置比较器进行排序
`tree.CompareFunction = compareFunction;`

树节点排序示例

```
int i = 0;
while (i++ < 5)
{
    Node node = new Node();
    node.Name = "" + Utils.RandomInt(10) + "-" + i;
    node.Parent = group;
    box.Add(node);
}

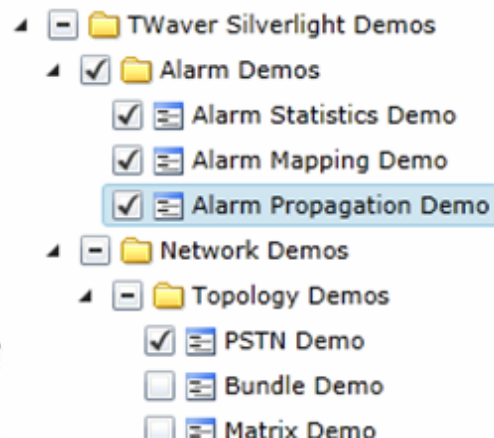
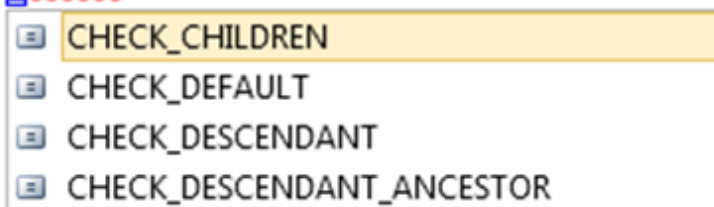
tree.CompareFunction = (Data data1, Data data2) =>
{
    return Comparer<string>.Default.Compare(data1.Name, data2.Name);
};
```



Tree的框选模式

- Tree提供框选模式，可通过下面的方法切换：
*tree.CheckMode = Tree<Data>.CHECK_****

```
tree.CheckMode = Tree<Data>.check_
```



Tree有四种框选模式，见：
CheckModeDemo

Table的使用

TM
T
W
a
v
e
r

- 与DataBox绑定，用以展示容器内元素属性信息，每行对应一个数据元素
- 支持数据元素的三类属性的显示：accessor、style、client
- 数据属性变化，同步更新，可编辑
- 继承于System.Windows.Controls.DataGrid.

Table的使用

TWaver™

TWaver.Controls.Table<> -用于展示DataBox中元素属性的组件，每行对应一个数据元素

```
Table<Data> table = new Table<Data>(box);
```

Name	Is Single	Color
1	<input type="checkbox"/>	#FF8EC80E
2	<input checked="" type="checkbox"/>	#FF9BF5D4
3	<input checked="" type="checkbox"/>	#FFDE7CD3
4	<input type="checkbox"/>	#FF9FD331
5	<input checked="" type="checkbox"/>	#FF5875B6

Mapping ID	Severity	Acked	Cleared
1	Indeterminate	<input type="checkbox"/>	<input checked="" type="checkbox"/>
2	Warning	<input type="checkbox"/>	<input type="checkbox"/>
3	Minor	<input type="checkbox"/>	<input type="checkbox"/>
4	Major	<input type="checkbox"/>	<input type="checkbox"/>
5	Critical	<input type="checkbox"/>	<input type="checkbox"/>

创建表格

```
DataBox<Data> box = new DataBox<Data>();
```

```
int i = 0;
while (i++ < 10)
{
    Data data = new Data();
    data.Name = "" + i;
    data.SetClient("age", Utils.RandomInt(30));
    data.SetClient("isSingle", Utils.RandomBool());
    data.SetClient("color", Utils.RandomColor());
    box.Add(data);
}
```

```
Table<Data> table = new Table<Data>(box);
```

```
DataGridColumn column = new TableTextColumn(Consts.PROPERTY_TYPE_ACCESSOR,
"Name", "Name");
table.Columns.Add(column);
column = new TableCheckBoxColumn(Consts.PROPERTY_TYPE_CLIENT, "isSingle", "Is
Single");
table.Columns.Add(column);
column = new TableColorColumn(Consts.PROPERTY_TYPE_CLIENT, "color", "Color");
table.Columns.Add(column);
```

Name	Is Single	Color
1	<input type="checkbox"/>	#FF8EC80E
2	<input checked="" type="checkbox"/>	#FF9BF5D4
3	<input checked="" type="checkbox"/>	#FFDE7CD3
4	<input type="checkbox"/>	#FF9FD331
5	<input checked="" type="checkbox"/>	#FF5875B6

表格数据顺序

- **Table**中数据存在三种顺序（正序，反序，根层）

```
table.IterateMode = Consts. ITERATE_MODE_***;
```

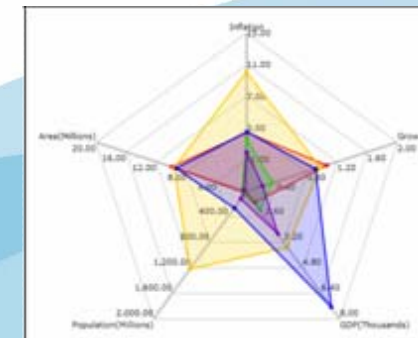
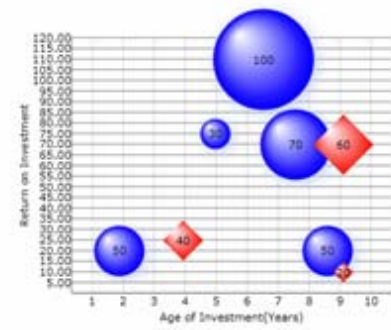
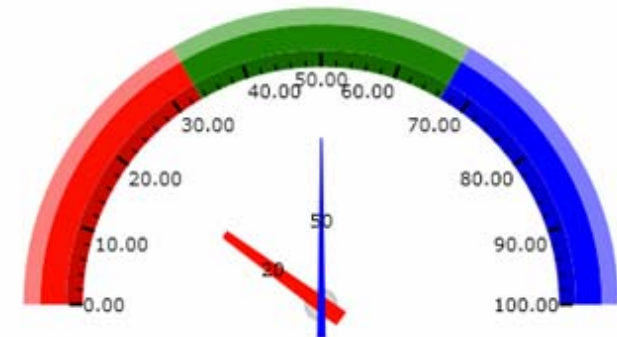
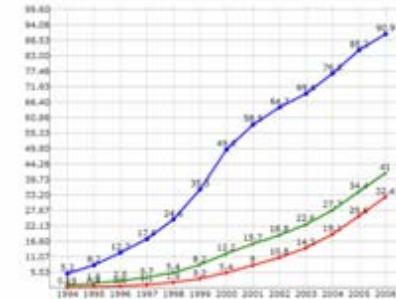
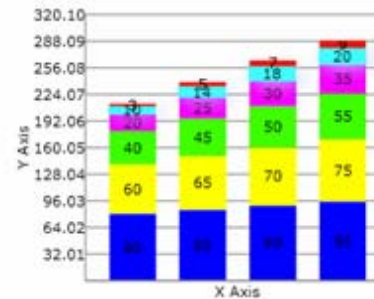
- Consts.ITERATE_MODE_REVERSE
- **Consts.ITERATE_MODE_FORWARD**
- Consts.ITERATE_MODE_ROOTS

- 通过**dataBox.Move***(element)**调整行的顺序

Chart的使用

TM
Twaiver

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



ChartValue & ChartValues

TWaver™

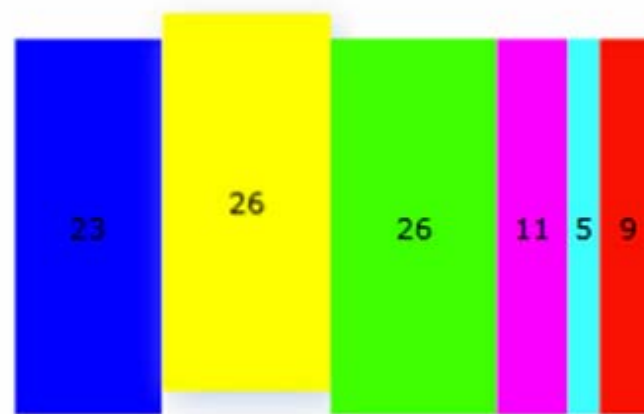
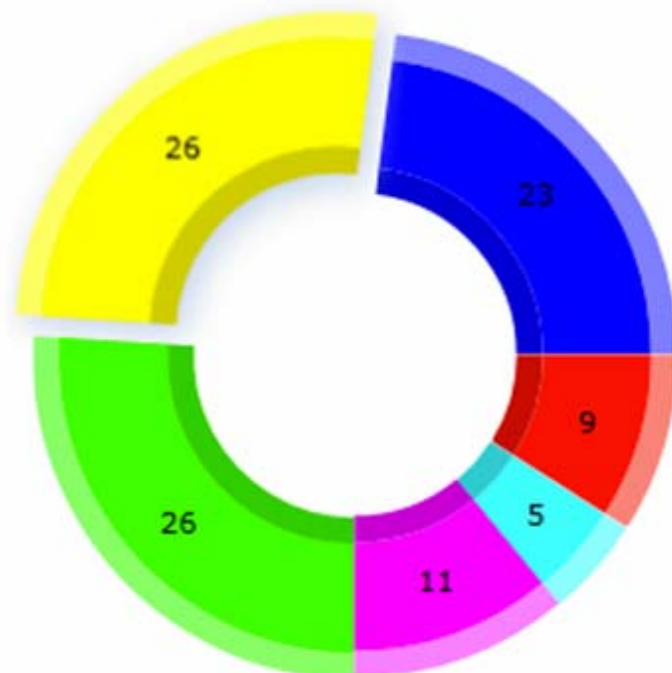
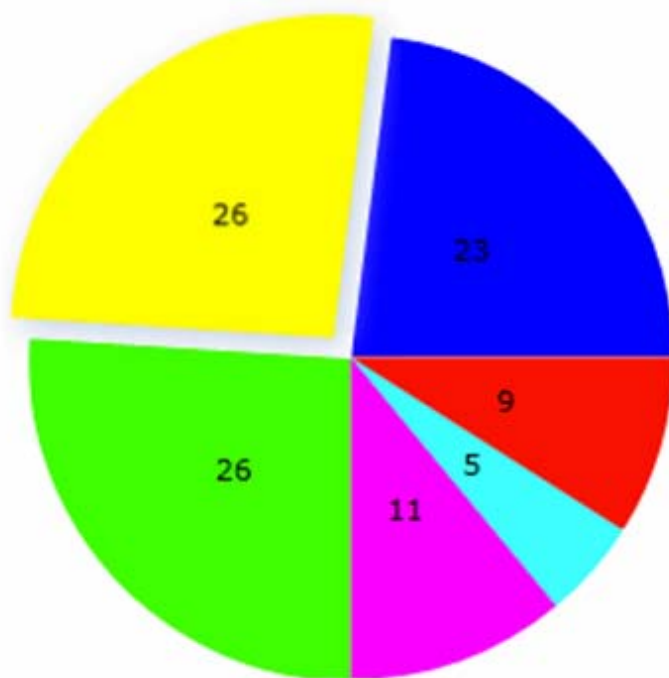
- Chart与DataBox关联，以图表的形式表现dataBox中数据元素的Chart数值属性
- Chart数值属性有两种：CHART_VALUE和CHART_VALUES，如下设置

```
element.SetStyle(Styles.CHART_VALUE, 27d);
```

```
element.SetStyle(Styles.CHART_VALUES, new double[]{27, 37, 48});
```

PieChart

TM
waver



PieChart

```
public PieChartTest() {  
    InitializeComponent();  
  
    DataBox<Data> box = new DataBox<Data>();  
    PieChart<Data> chart = new PieChart<Data>(box);  
  
    createData(box, "A", 10, Utils.CreateColor(0xFF4CB743))  
    createData(box, "B", 20, Utils.CreateColor(0xFFD6EAC9))  
    createData(box, "C", 40, Utils.CreateColor(0xFF77DD77))  
  
    chart.Type = Consts.PIECHART_TYPE_OVALDONUT;  
    chart.ValueTextFunction = (IData item, double value) =>  
    {  
        return item.Name + " - " + (int)(value / chart.Sum * 100) + "%";  
    };  
  
    this.LayoutRoot.Children.Add(chart);  
}  
  
private void createData(DataBox<Data> box, string name, double value, Color color){  
    Data data = new Data();  
    data.SetClient(Styles.CHART_VALUE, value);  
    data.SetClient(Styles.CHART_COLOR, color);  
    data.SetClient(Styles.CHART_VALUE_COLOR, Utils.CreateColor(0xFF555555));  
    data.SetClient(Styles.CHART_VALUE_FONT, new FontFamily("heivetica"));  
    data.Name = name;  
    box.Add(data);  
}
```



TWaver™

LineChart

```

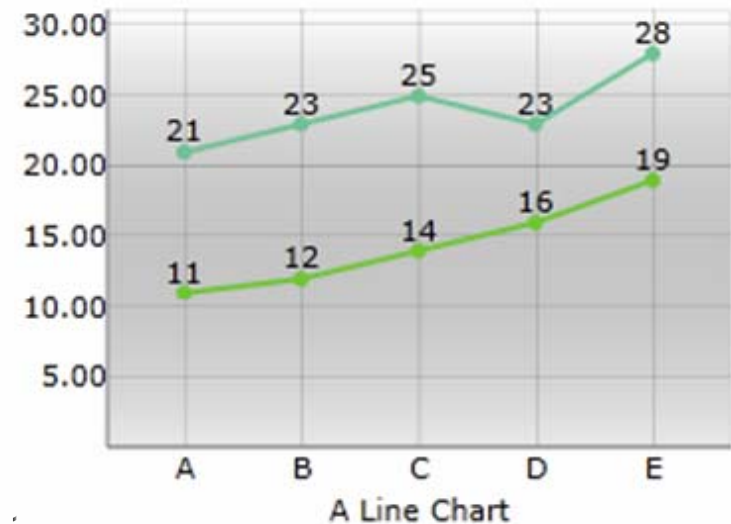
public LineChartTest()
{
    InitializeComponent();

    DataBox<Data> box = new DataBox<Data>();
    LineChart<Data> chart = new LineChart<Data>(box);

    Element data = new Element();
    data.SetStyle(Styles.CHART_VALUES, new double[] {
19d });
    box.Add(data);
    data = new Element();
    data.SetStyle(Styles.CHART_VALUES, new double[] { 21d, 23d, 25d, 23d,
28d });
    box.Add(data);

    chart.XScaleTexts = new string[] { "A", "B", "C", "D", "E" };
    chart.LowerLimit = 0d;
    chart.YScaleValueGap = 5d;
    chart.XAxisText = "A Line Chart";
    chart.BackgroundVisible = true;
    this.LayoutRoot.Children.Add(chart);
}

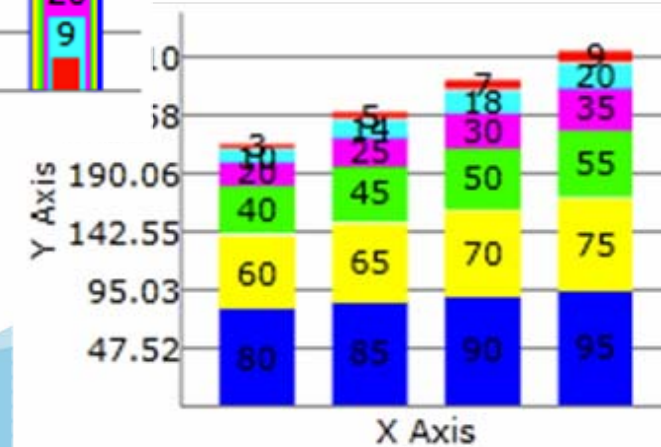
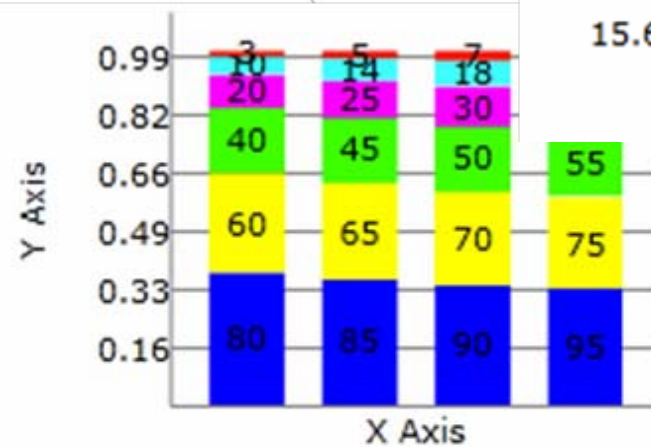
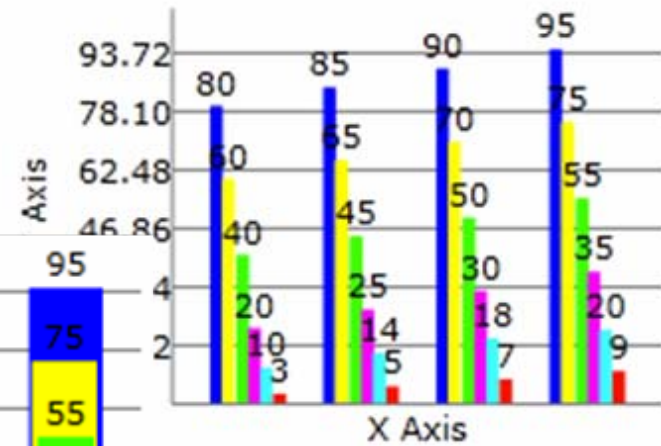
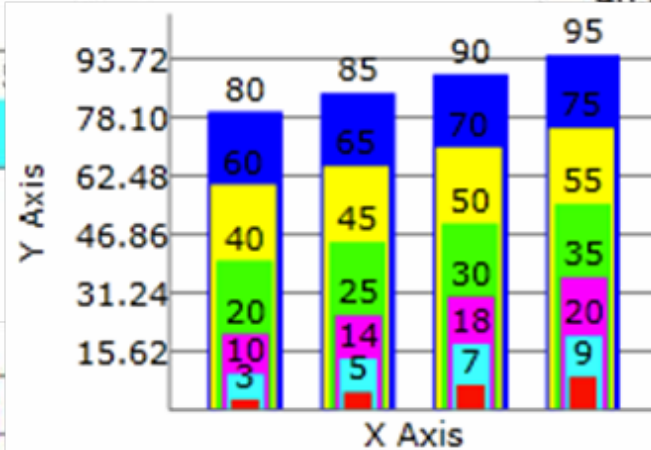
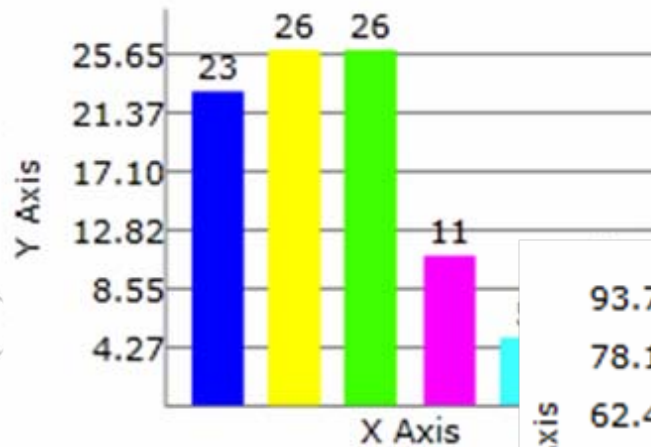
```



TWaver™

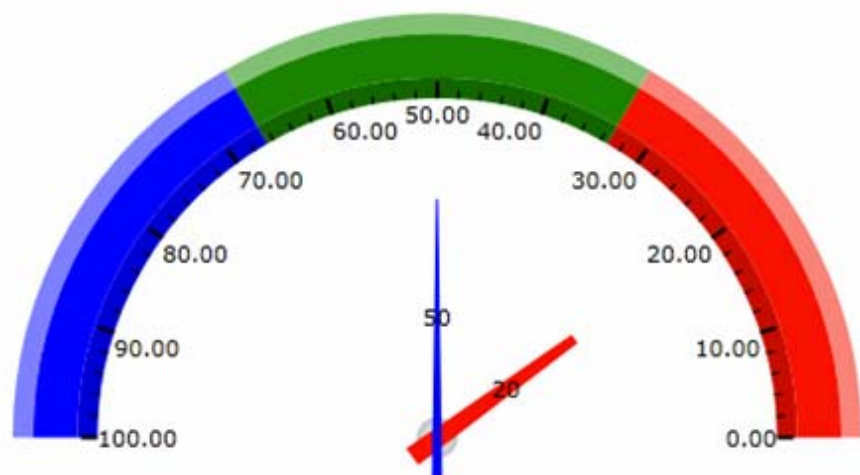
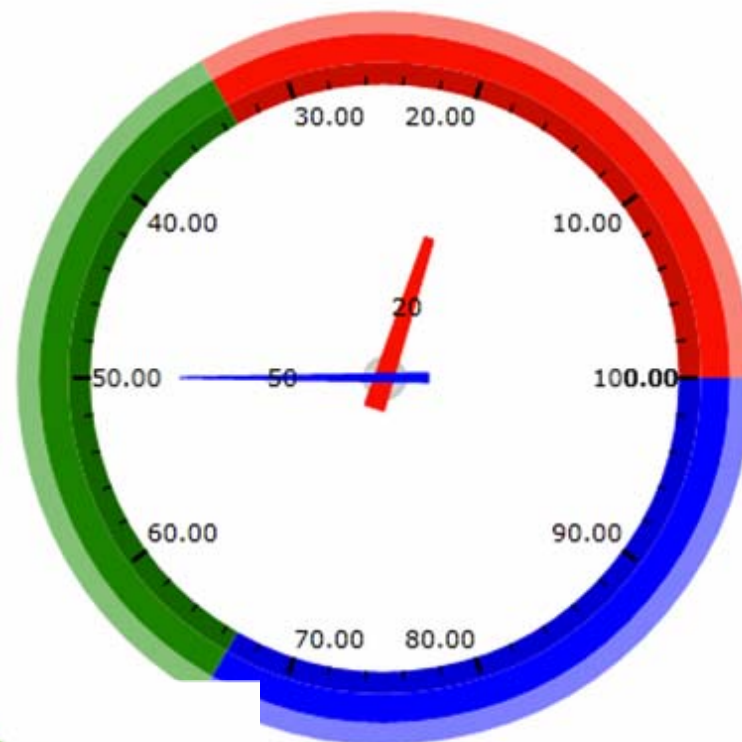
BarChart

TM
Twaiver



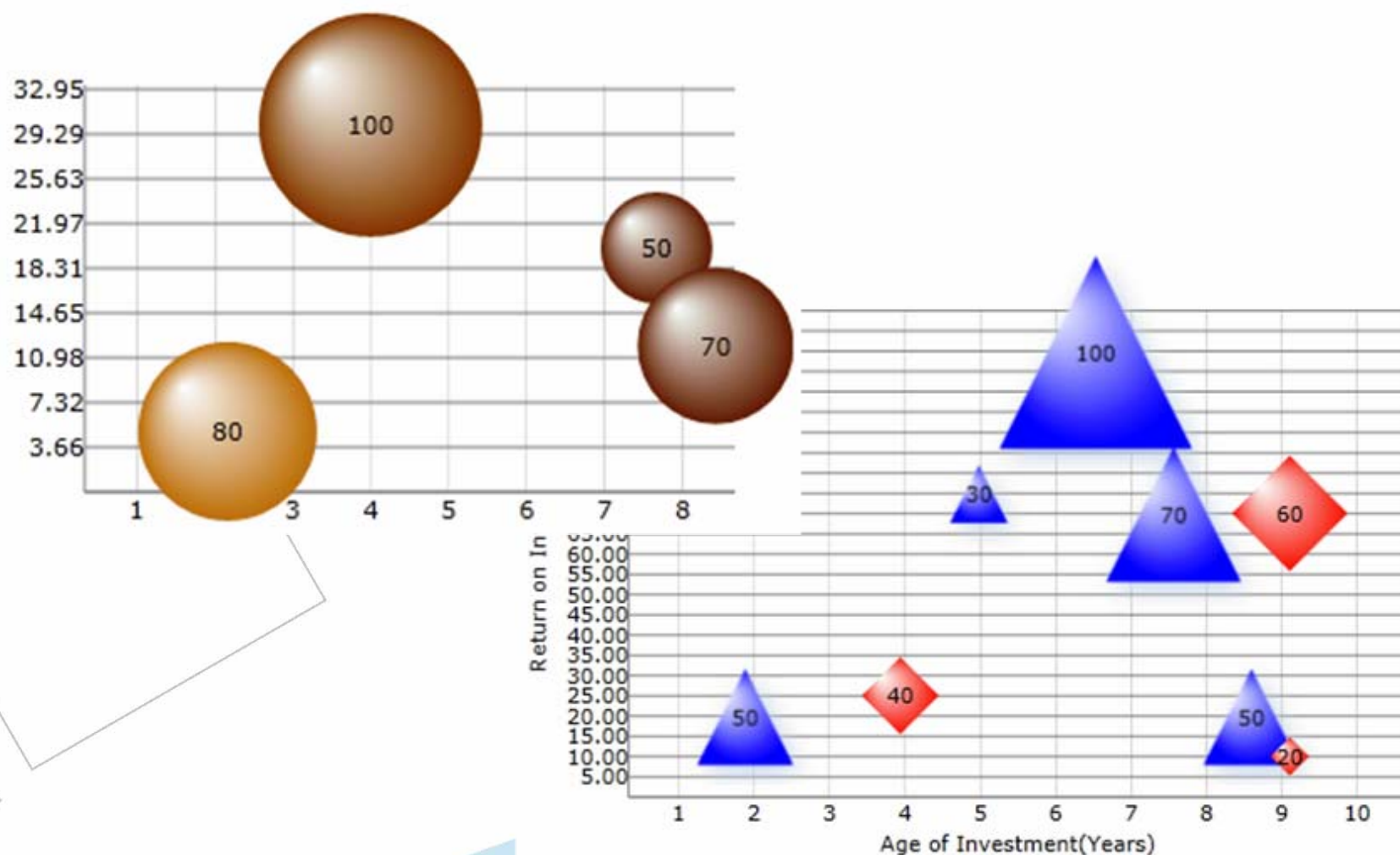
DialChart

Twaver™



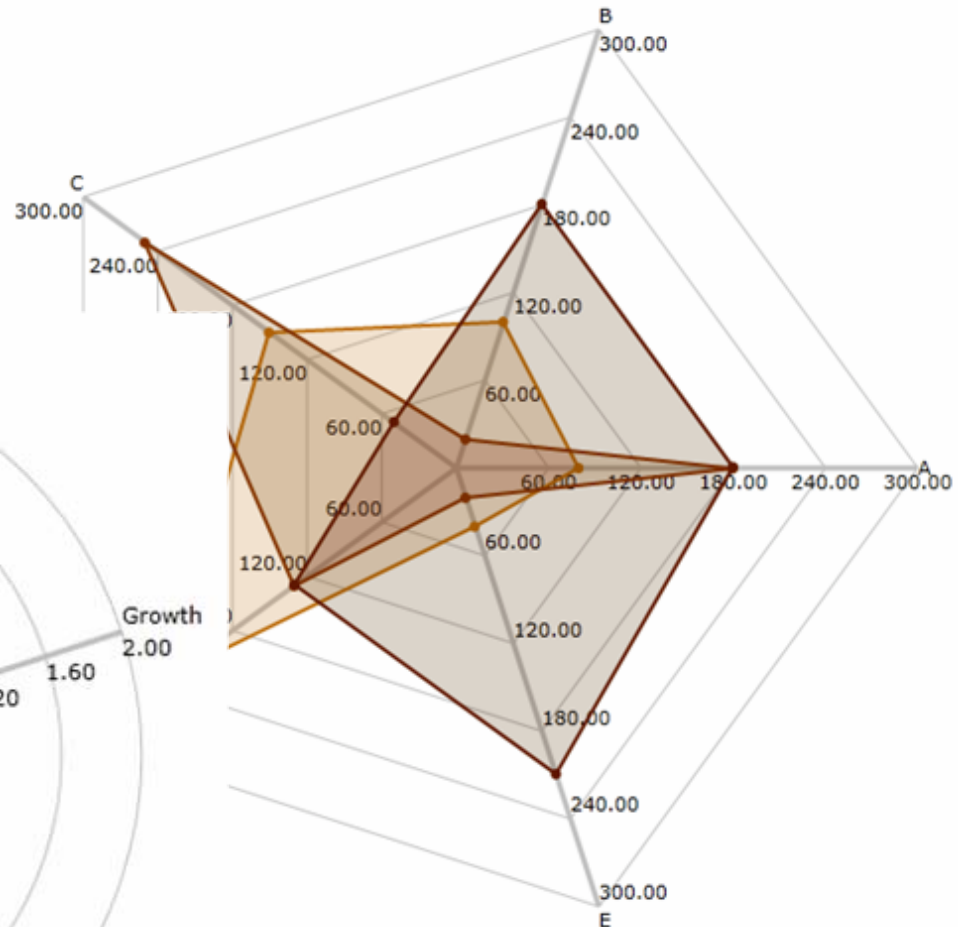
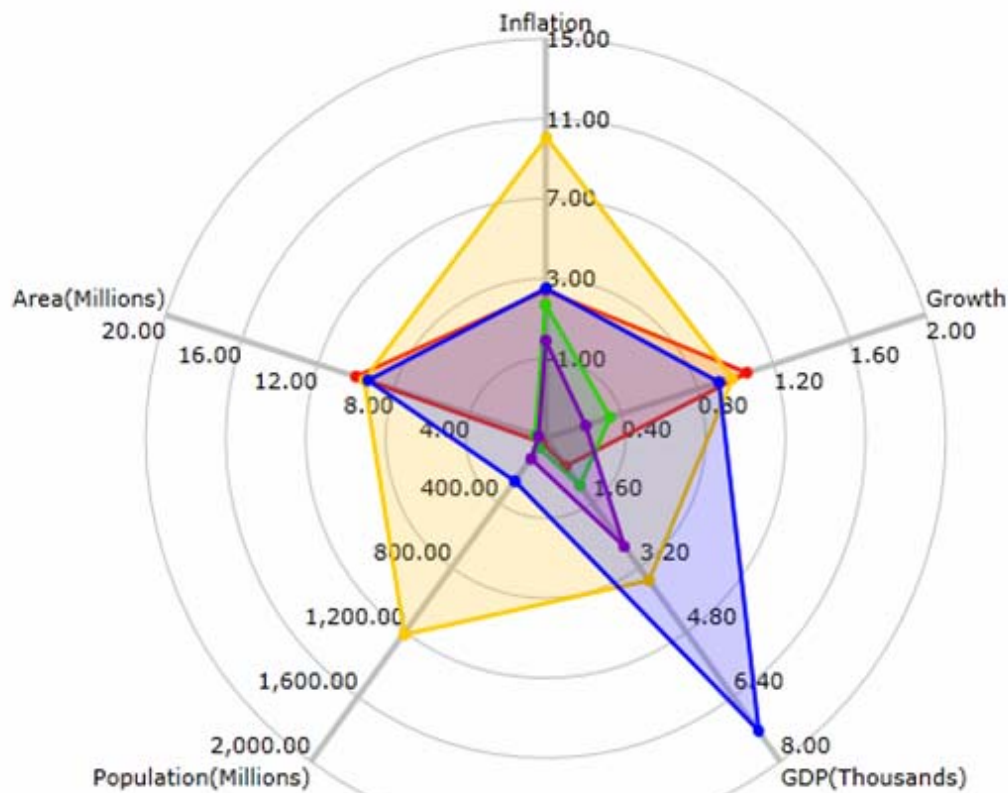
BubbleChart

TWaver™



RadarChart

Twaver™



告警的使用

TM
waver

告警：反映网元或设备运行状态的业务元素

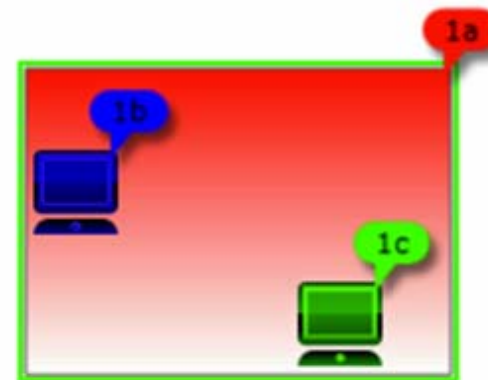
- 告警对象 / 告警级别
- 告警容器
- 告警状态 / 告警统计
- 告警呈现

告警级别

- *TWaver. AlarmSeverity*
- **Alarm severity** -告警级别：反映告警的紧急程度 包含：name, nickName, value, color 等属性
- 默认告警级别 -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.Clear();
AlarmSeverity.Register(1, "a", "a", Utils.CreateColor(0xFFFF0000),
"AAA");

定制告警级别示例

```
public CustomAlarmSeverityTest() {  
    ...  
    AlarmSeverity.Clear();  
    AlarmSeverity a = AlarmSeverity.Register(1, "a", "a", Utils.CreateColor(0xFFFF0000), "AAA");  
    AlarmSeverity b = AlarmSeverity.Register(2, "b", "b", Utils.CreateColor(0xFF0000FF), "BBB");  
    AlarmSeverity c = AlarmSeverity.Register(3, "c", "c", Utils.CreateColor(0xFF00FF00), "CCC");  
  
    ElementBox box = new ElementBox();  
  
    Group node1 = new Group();  
    node1.IsExpanded = true;  
    Node node2 = new Node();  
    node2.SetLocation(100, 100);  
    Node node3 = new Node();  
    node3.SetLocation(200, 150);  
    node3.Parent = node1;  
    node2.Parent = node1;  
    box.Add(node1);  
    box.Add(node2);  
    box.Add(node3);  
  
    AddAlarm(box.AlarmBox, node1, a);  
    AddAlarm(box.AlarmBox, node2, b);  
    AddAlarm(box.AlarmBox, node3, c);  
    ...  
}  
  
private void AddAlarm(AlarmBox alarmBox, Element element, AlarmSeverity severity) {  
    Alarm alarm = new Alarm(null, element.ID, severity);  
    alarmBox.Add(alarm);  
}
```



告警对象

TWaver™

- TWaver.Alarm - 描述设备故障或者网络异常的数据元素
- 告警属性:
ID, ElementID, AlarmSeverity, IsAcked, IsCleared
添加其他属性: alarm.SetClient(key, value).

告警容器

- AlarmBox -管理告警对象的容器
- 基本操作
Add/Remove/Clear ...
- 快速查找器 / 监听器



AlarmBox

告警状态

TWaver™

- **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

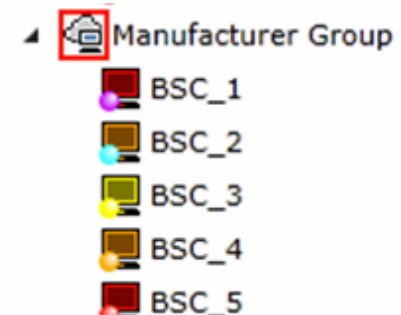
告警状态统计

TM
waver

- **AlarmStateStatistics** -告警状态统计：对整个elementBox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



Mapping ID	Severity	Acked	Cleared	Raised Time
1	Indeterminate	<input type="checkbox"/>	<input type="checkbox"/>	2011-08-30
2	Warning	<input type="checkbox"/>	<input type="checkbox"/>	2011-08-30
3	Minor	<input type="checkbox"/>	<input type="checkbox"/>	2011-08-30
4	Major	<input type="checkbox"/>	<input type="checkbox"/>	2011-08-30
5	Critical	<input type="checkbox"/>	<input type="checkbox"/>	2011-08-30

Name	New	Acked	Total
Critical	3	2	5
Major	5	1	6
Minor	1	3	4
Warning	8	1	9
Indeterminate	0	7	7
Cleared	2	1	3
Total	19	15	34

Thank you

- 论坛: twaver.servasoft.com/forum
- Email - tw-service@servasoft.com