WEBOS.面向积木编程.分形架构

--ToyBricks技术体系

淘宝网.通用产品部　九章

SINA微博:琴上的日月

百家讲坛

讲稿提纲

1.TB-WEBOS概述

2.设计原则

3.组件模型

4.ToyBricks技术体系

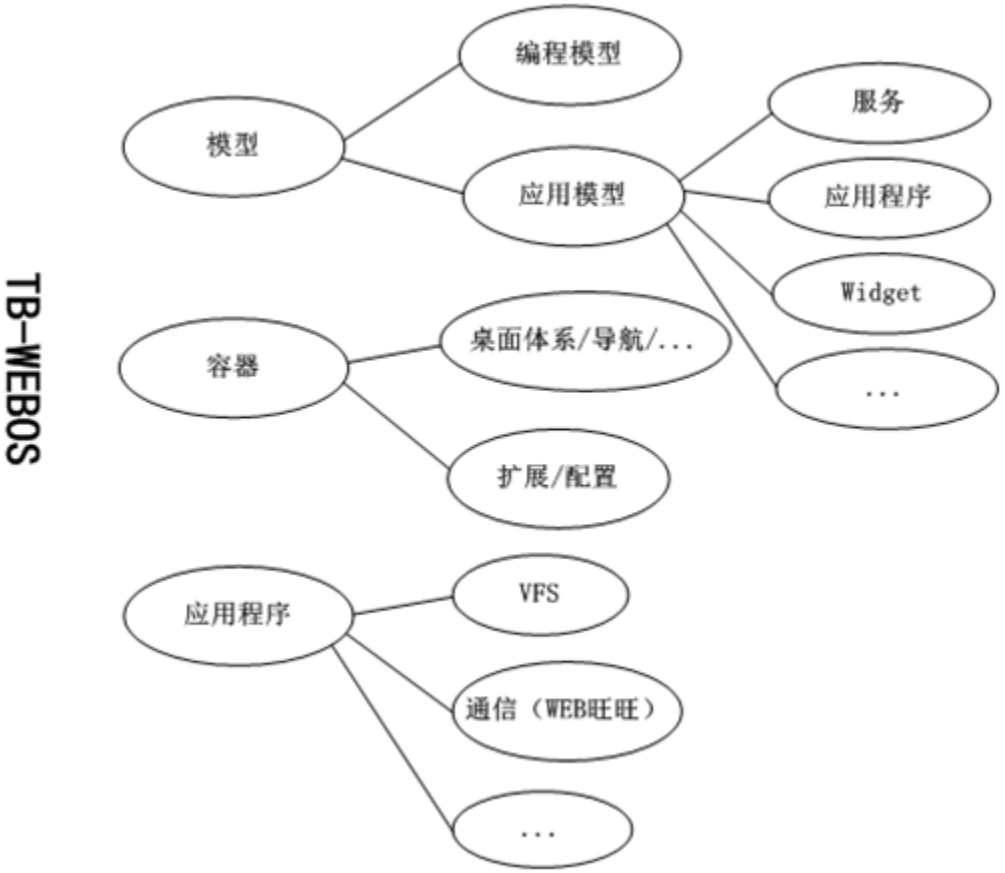
5.Aquarell模板引擎

6.ToyBricks RIA

7.ToyBricks 组件库

8. ToyBricks WebOS

1.淘宝WEBOS概述


TB-WEBOS
模型
编程模型
应用模型
服务
应用程序
Widget
...
容器
桌面体系/导航/...
扩展/配置
应用程序
VFS
通信（WEB旺旺）
...

1.淘宝WEBOS概述


业务应用
应用程序
应用程序
应用程序
应用程序
应用程序
应用程序
应用程序
应用程序
容器
桌面体系/导航/...
扩展/配置
应用模型
服务
应用程序
Widget
...
核心模型
编程模型?

2. 设计原则

1.奥卡姆剃刀

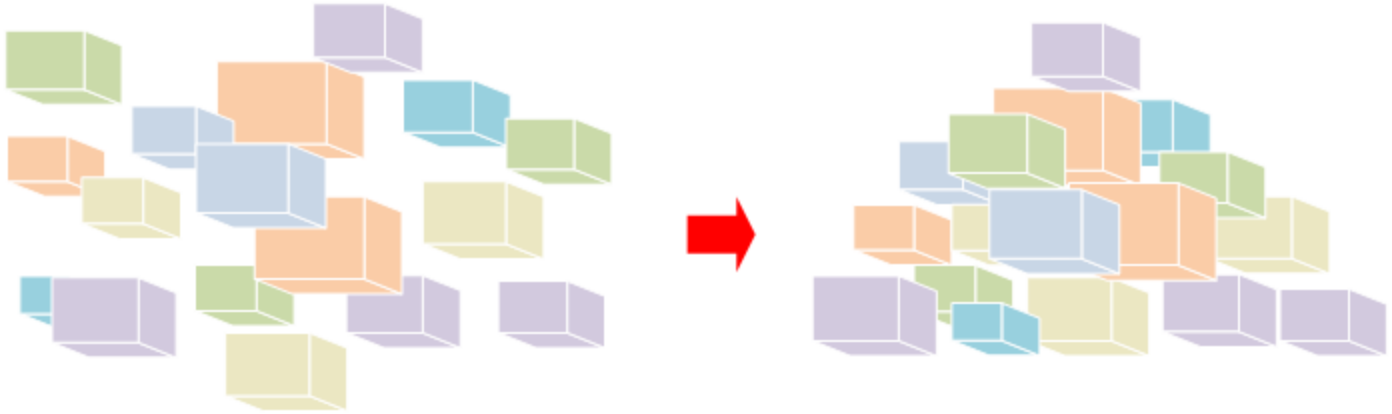
2.架构水平化/应用垂直化

3.聚合优先

4.最大程度复用

5....

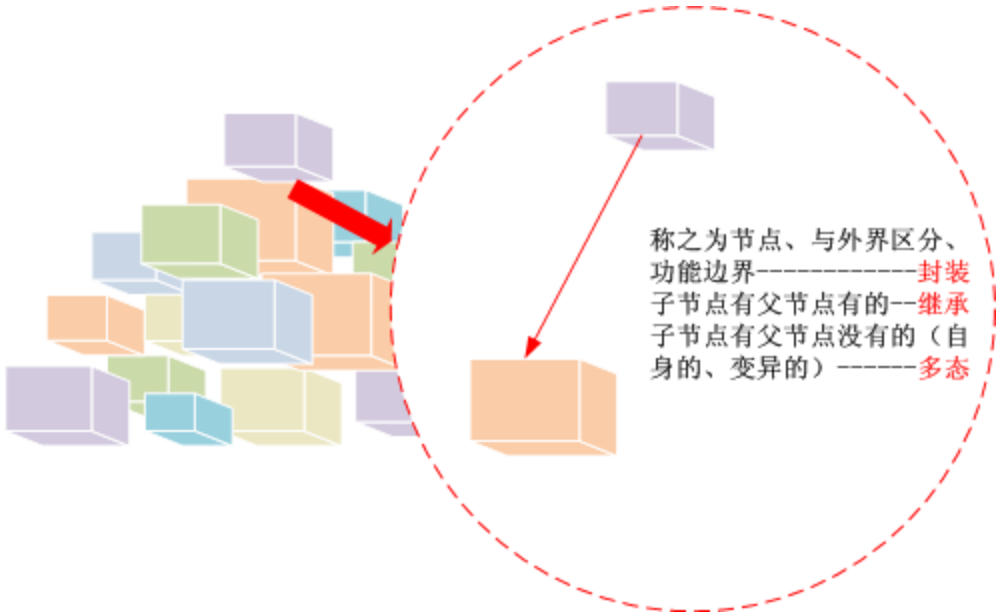
3. 组件模型 / 设计的本质



设计的本质是基于某种目的将一个体系按照某种方法转化为另一个体系。

很多情形下，设计是向着“有序”的方向演进的。

3. 组件模型 / 结构金字塔中的OO



系统全部功能数= ∑ f(n)=∑ f(s(n)+r(n))
上式中，s(n)为节点n特定的功能数，r(n)为该节点在其他节点(m)中s(m)的子集

结论，我们要构造的组件模型需体现：
→节点：具备“封装”、“继承”、“多态”特征的个体；
→不同尺度中的节点都具备上述特征（同构）
→分形结构（架构）

3. 组件模型 / 业界当前主流的组件模型

向OSGI及SCA致敬

Spring的创新（实现了将组件实现（Java等）到通用语言（XML）描述的映射）

…

3. 组件模型

ToyBricks（积木）的理想是打造程序员喜欢的组件模型，实现系统真正的模块化构建，这个体系可以有许多特性，如：

1.简单、开放
以开发者所想的方式构建

2. 一切皆组件
不同尺度空间的同构性

3.模型与实现无关
组件用通用的语言（比如XML）描述，至于实现则可用Java/C++/.NET/Ruby/RMI/WebService…

4.覆盖任意技术梯度的开发者
初出茅庐会觉得很容易，大神也有发挥的广阔空间，允许“为所欲为”

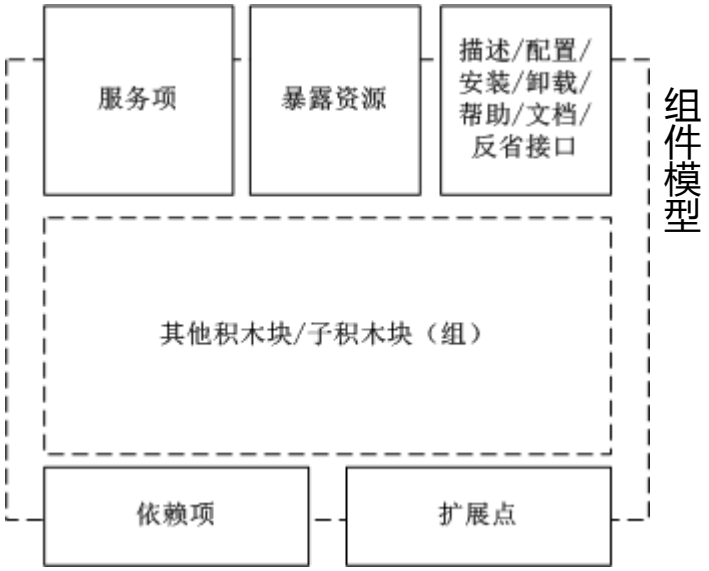
5.OO
组件良好封装（清晰边界），可继承，可多态

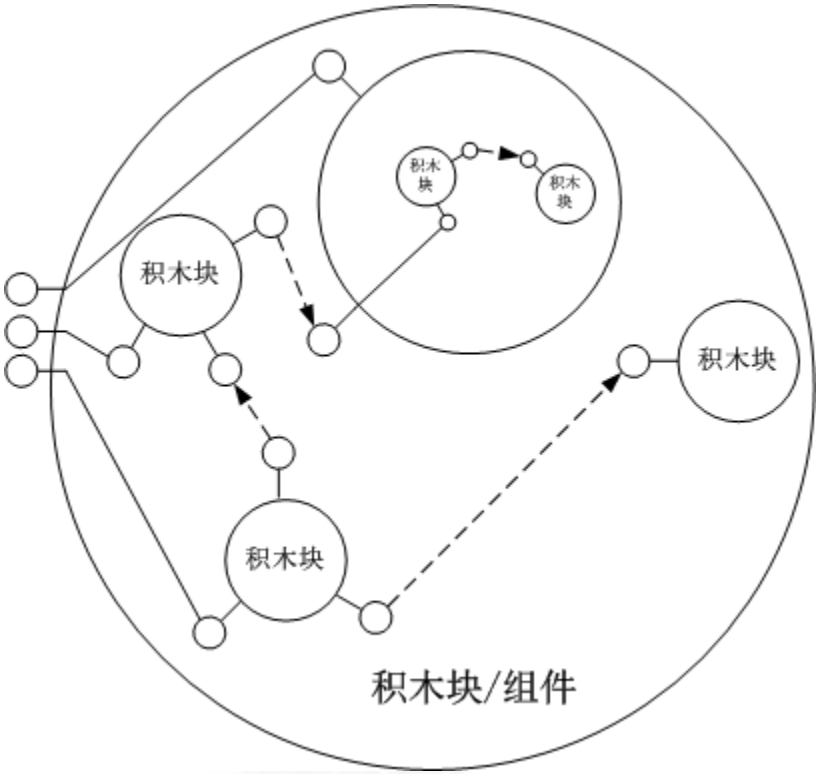
6.为微内核的架构提供可行性

….

也可以只有一个，那就是：分形

3. 组件模型


服务项
暴露资源
描述/配置/
安装/卸载/
帮助/文档/
反省接口
组件模型
其他积木块/子积木块（组）
依赖项
扩展点


积木块
积木块
积木块
积木块
积木块
积木块/组件

3. 组件模型 / 分形架构（套娃架构）

1. 问题空间决定了解空间中积木单元的层级（计算、业务逻辑、UI、大粒度业务、应用程序、模块、子系统...）
2. 职责决定了积木的边界与外部的契约

解决不同层级组件模型的巨大差异、知识资产积累的困难等问题的关键是“将积木定义为可以抽象概括所有层级的概念”。

ToyBricks视角下的组件（积木块）概念具备如下四个基本特征：

1. 具备某种意义上的完整性与独立边界；
2. 同一或不同尺度上的可组合（聚合）特性；
3. 可扩展能力，且扩展不以破坏组件内部结构为前提、以被扩展为另一个组件的方式而进行（**微核优于分层**）；

4. 特征的自解释(反省)或提供解释的手段（接口）；

显然，以上定义体现了积木之间的分形特征：多个组件可以组装成为一个更大粒度的组件，一个通用型的组件可以被扩展成为一个个性化的组件，一个按钮是一个组件，一个复杂的窗体本身也是一个组件，而且两者没有结构上的本质区别（体现不同尺度上的自相似性），从而实现最大程度的重用。

世界上最稳定、最优化的结构组成应该是分形的。

面向积木编程（OBP）

OOP的两个基本方面
分工+协作

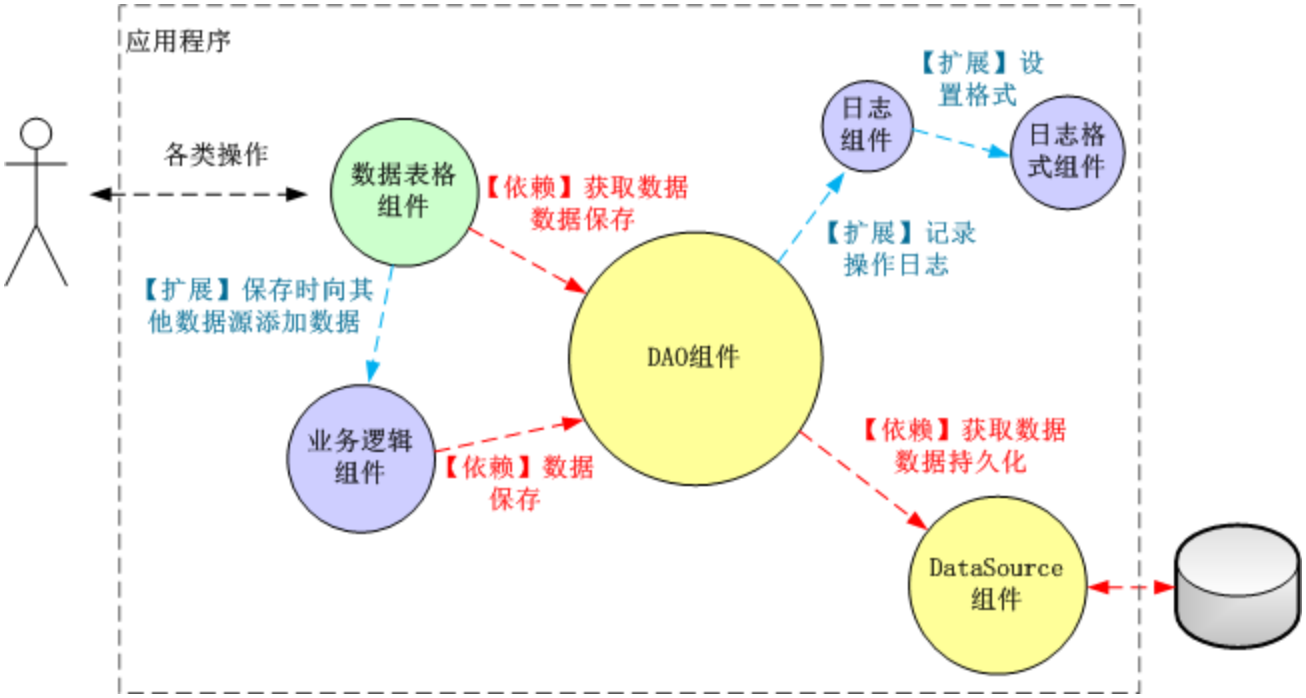
分工：识别、界定参与者的边界、职责
协作：定义参与者之间的契约

OBP是OOP的扩展，在OBP的视角下，系统是众多积木块基于某种规则的组合，其思维过程如OOP一样分为两个步骤：

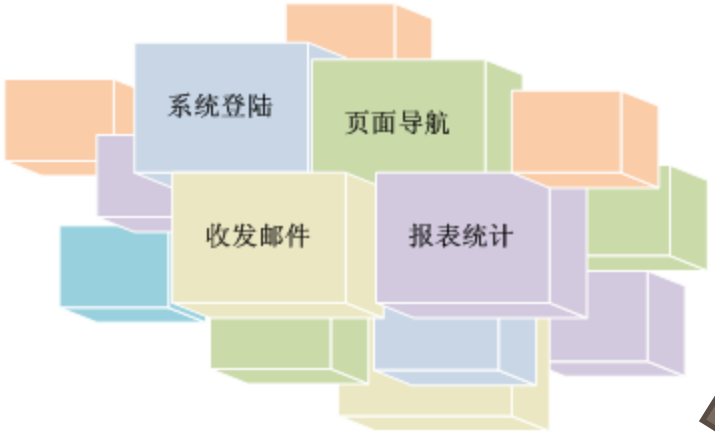
1.识别积木块；
2.连接积木块；

比如：现在要做带有CIUD功能的应用程序，那么思维方式应该是：首先想搭建这个积木块（该应用程序也应该是一个更大粒度的积木块），那么可以采用如下现有的积木块：DataGrid（负责展现数据）、DAO（数据操纵接口）、DataSource（数据源），然后关系是：DataSource依赖与DAO，DAO依赖于DataSource，根据实际需要，DataGrid本身能力不够，需要编写业务逻辑组件进行扩展。

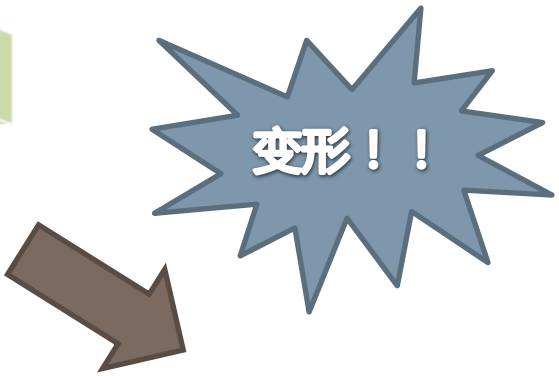
4.面向积木方法论 / 面向积木编程



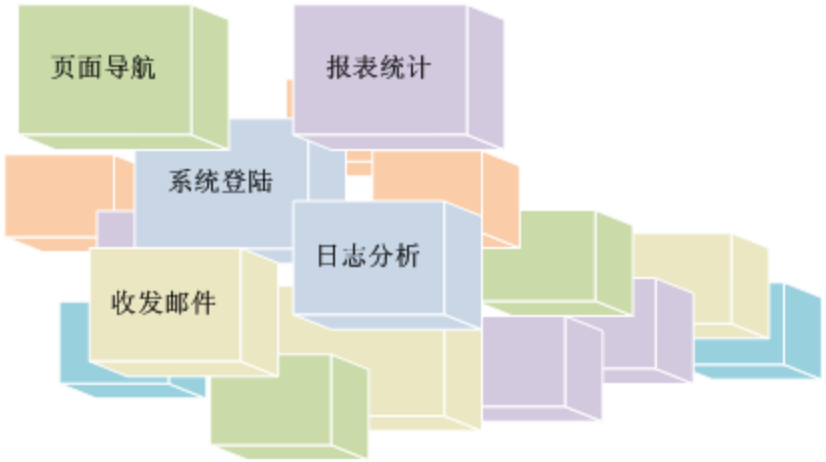
4.面向积木方法论 / 积木化视角下的系统构成



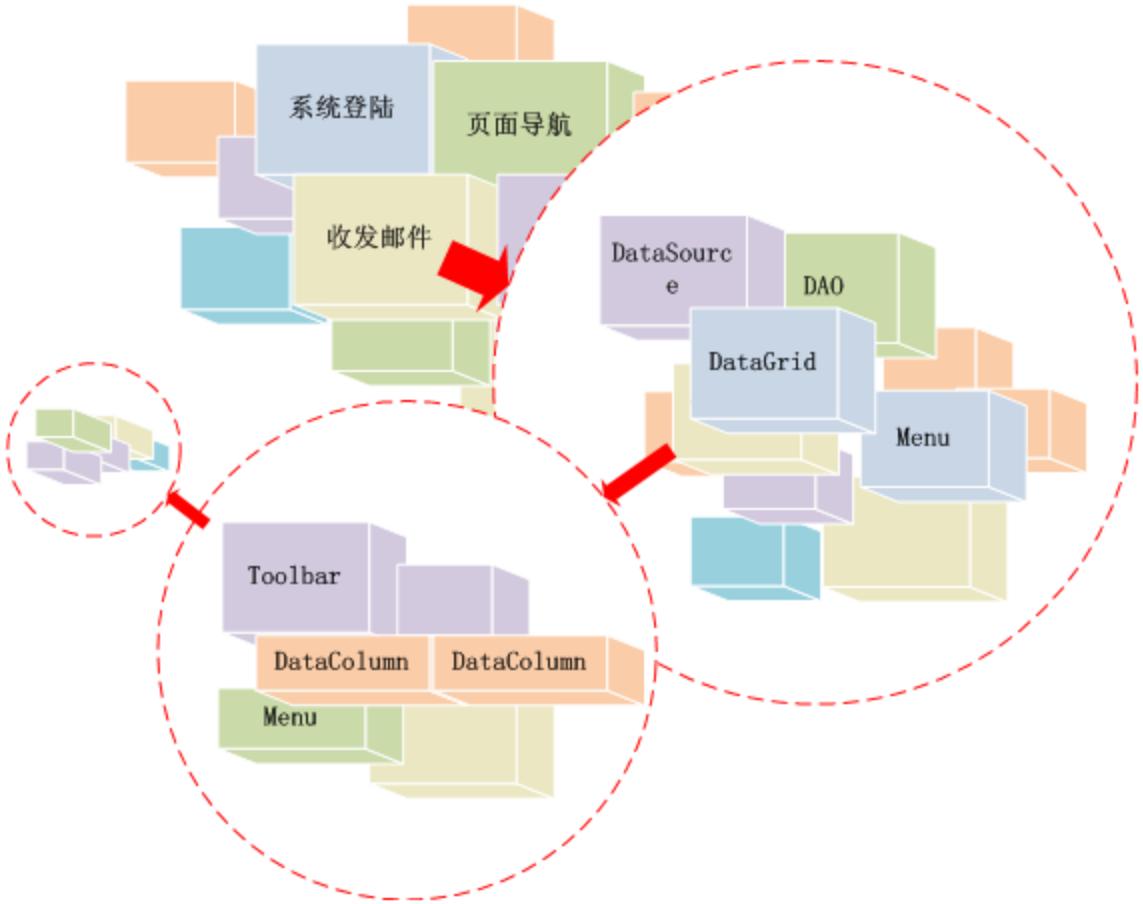
A系统



B系统

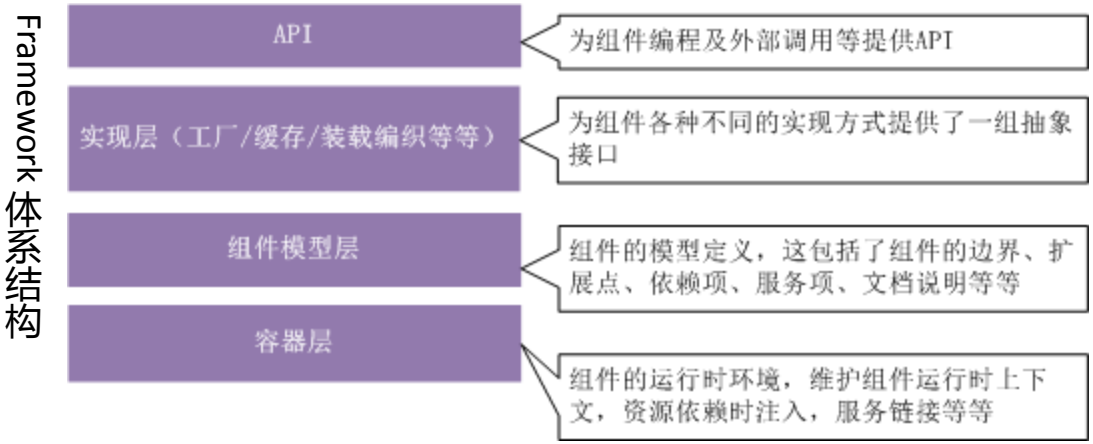


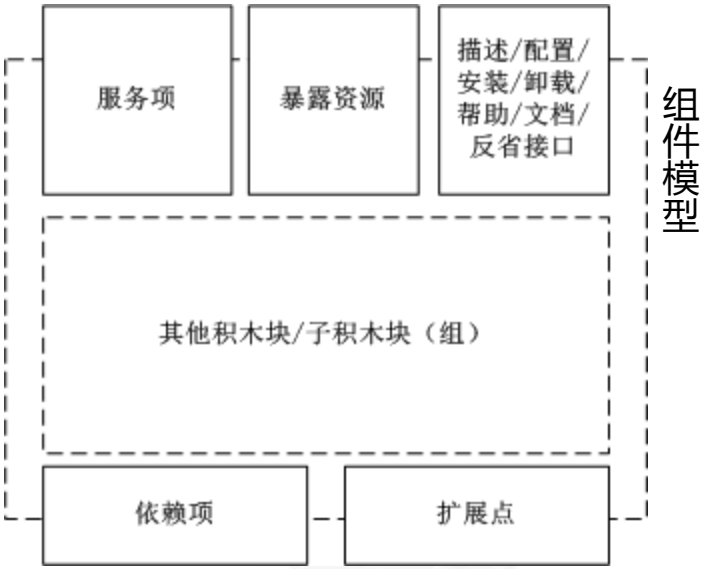
4.面向积木方法论 / 积木化视角下的系统构成



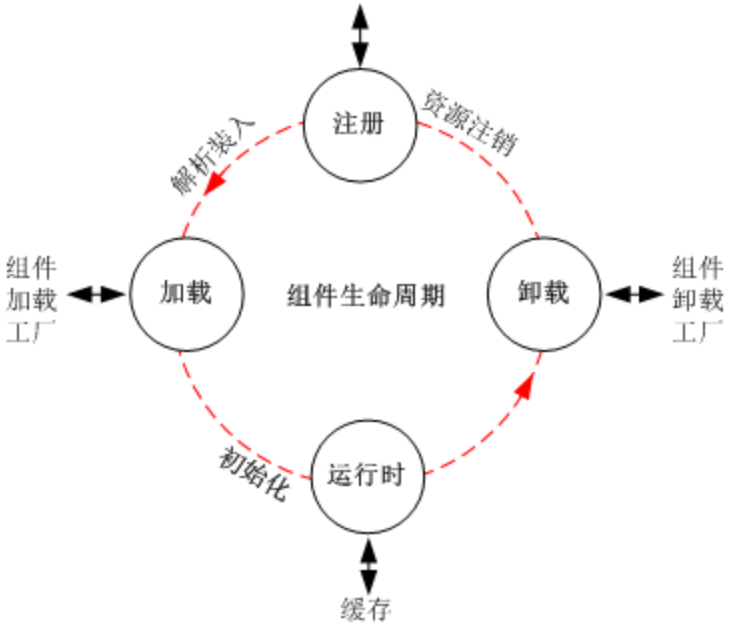
不同尺度空间下的系统构成

5.ToyBricks Framework / 总体结构





5.ToyBricks Framework / 总体结构



组件生命周期

5.ToyBricks Framework / 组件描述符举例(RIA Prolet实现方式)

```
<?xml version="1.0" encoding="UTF-8"?>
<toybricks xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:noNamespaceSchemaLocation="http://www.cloudsinger.org/toybricks/component/ria">
    <component.prolet id="rsManager" title="资源管理器"
        viewFile.cur="rsManager.aq.html" className.cur="RSManagerBiz"
        icon16.cur="icon16.gif" icon32.cur="icon32.gif" parent="cloudsinger.webos"
        extends="cloudsinger.webos/$rsManagerBase" lafPath.curView="$laf/$"
        visibility="all">

        <!-- 文件夹 -->
        <sub-component.prolet id="folder" title="文件夹">
            <sub-component.prolet id="new" title="新建文件夹"
                viewFile.cur="folder/new.aq.html" className.cur="folder/NewBiz"
                icon16.cur="folder/icon16.gif" />
            <sub-component.prolet id="property" title="文件夹属性"
                viewFile.cur="folder/property.aq.html" className.cur="folder/PropertyBiz"
                icon16.cur="folder/icon16.gif" />
        </sub-component.prolet>

        <!-- 文件-->
        <sub-component.prolet id="uploadFile" title="文件">
            <sub-component.prolet id="upload" title="上传文件"
                viewFile.cur="uploadFile/upload.aq.html" className.cur="uploadFile/UploadBiz"
                icon16.cur="uploadFile/icon16.gif" lafPath.curView="$laf/$" />
            <sub-component.prolet id="download" title="下载|打开文件"
                className.cur="uploadFile/DownloadBiz" icon16.cur="uploadFile/icon16.gif" />
            <sub-component.prolet id="property" title="文件属性"
                viewFile.cur="uploadFile/property.aq.html" className.cur="uploadFile/PropertyBiz"
                icon16.cur="uploadFile/icon16.gif" />
        </sub-component.prolet>

```

5.ToyBricks Framework / V4.0特性

ToyBricks –Base V4.0主要特性：

1.容易理解、简单易用；

2.描述与实现分离，可配置的运行时环境（ClassLoader）；

3.支持组件间的继承与多态；

4.**支持最小粒度到方法（服务）的DI/AOP，支持多对一的服务实现**；

5.可实现真正意义的多重继承；

6.支持组件关系的运行时改变；

……

（以上红色字体部分为区别于Spring Framework的方面）

5.ToyBricks Framework / 依赖项 / 简单的依赖连接（DI）

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <toybricks xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
3      xsi:noNamespaceSchemaLocation="http://www.cloudsinger.org/toybricks/components/java">
4      <component id="a" title="测试组件A"
5          className="test.org.cloudsinger.toybricks.api.BeanA">
6          <junction to="this:getBBean" from="b" />
7          <junction to="this:foolD" from="b:foo" />
8      </component>
9      <component id="_b" title="测试组件_B" abstract="true
10         className="test.org.cloudsinger.toybricks.api
11     <component id="b" title="单例且预加载组件" extends="_b" singleton="true"
12         startup="0">
13     </component>
14     <junction from="b:method0" to="a:method0" />
15 </toybricks>
```

该Schema定义了组件的实现方式为java

Junction（挂接），路径为b的组件的服务项连接到当前组件的依赖项

直接描述组件的关系

5.ToyBricks Framework / 依赖项 / 简单的依赖连接（DI）

```
3  import org.cloudsinger.toybricks.api._Dependency;
4
5  public abstract class BeanA {
6
7      public BeanA() {
8
9      }
10
11     public void test0() {
12         System.out.println(getBBean().foo("somebody one"));
13     }
14
15     public void test1() {
16         System.out.println(foolD("somebody two"));
17     }
18
19     @_Dependency
20     protected abstract IBean getBBean();
21
22     @_Dependency
23     protected String foolD(String _input) {
24         return _input + "fool--------test";
25     }
26 }
27
```

组件(id="a")的代码

依赖项，直接以抽象方法体现，返回结果为另一个组件的实例

依赖项，非抽象方法（该方法可以直接用于单元测试）

百家讲坛

5.ToyBricks Framework / 依赖项 / 简单的依赖连接（DI）

```
<component id="a" title="测试组件A"
    className="test.org.cloudsinger.toybricks.ap
    <dependency name="getBBean" bind="getBBean">
        <input>
            <param type="type.string" />
        </input>
        <output>
            <param type="type.string" />
        </output>
    </dependency>
```

关于依赖项，除在Class（对于java的实现方式）中通过注释描述外，亦可以在描述符中这样描述

5.ToyBricks Framework / 依赖项 / 简单的依赖连接（DI）

```
1  package test.org.cloudsinger.toybricks.api;
2
3  import java.net.URL;
10
11 public class ToyBricksTest {
12
13     ToyBricks tb;
14
15     @Before
16     public void init() {
17         List<URL> list = new ArrayList<URL>();
18         list.add(ToyBricksTest.class.getResource("beans.xml"));
19         tb = new ToyBricks(list);
20     }
21
22     @Test
23     public void dependency() {
24         ((BeanA) tb.getComponent("a")).test1();
25     }
26
27     //@Test
28     public void cache() throws InterruptedException {
29         int ct = 0;
30         while (++ct < 1000) {
31             BeanA ba = tb.createComponent("a", BeanA.class);
32             Thread.currentThread().sleep(1000);
33         }
34
35     }
36 }
```

启动，构建上下文对象

获取组件，调用组件的服务项

5.ToyBricks Framework / 依赖项 / 多对一的依赖连接

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <toybricks xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
3      xsi:noNamespaceSchemaLocation="http://www.cloudsinger.org/toybricks/components/java">
4      <component id="a" title="测试组件A"
5          className="test.org.cloudsinger.toybricks.api.BeanA">
6          <junction to="this:getBBean" from="b" />
7          <junction to="this:getBBean" from="b2" />
8          <junction to="this:foolD" from="b:foo" />
9      </component>
10     <component id="_b" title="测试组件_B" abstract="true"
11         className="test.org.cloudsinger.toybricks.api.BeanB" />
12     <component id="b" title="单例且预加载组件" extends="_b" singleton="true"
13         startup="0" />
14     <component id="b2" title="组件2" extends="_b" />
15     <junction from="b:method0" to="a:method0" />
16 </toybricks>
```

两个组件都连接到“a”组件的依赖项

b2组件

```
@_Dependency
protected abstract Vector<IBean> getBBean();
```

a组件中的依赖项代码

5.ToyBricks Framework / 扩展点

场景：构造一个Dom（div）组件，其属性可被扩展。

举例：1.构造Dom组件，预留属性扩展点；2.构造Event组件，其可扩展Dom组件，且预留不同的事件扩展点；3.构造OnClick组件，扩展Event组件。

描述文件中三个组件的关系（依次扩展）

```
<component id="divDom" title="DivDom对象"
    className="test.org.cloudsinger.toybricks.api.extention.DivDom">
    <junction to="decorate" from="eventDecorator:decorate" />
</component>
<component id="eventDecorator" title="事件装饰"
    className="test.org.cloudsinger.toybricks.api.extention.EventDecorator">
    <junction to="onClick" from="onclickDecorator:click" />
</component>
<component id="onclickDecorator" title="单击事件装饰"
    className="test.org.cloudsinger.toybricks.api.extention.OnClickDecorator" />
```

5.ToyBricks Framework / 扩展点

```
5  public class DivDom {
6
7      public StringBuffer getHtml(String _innerText) {
8          StringBuffer rtn = new StringBuffer();
9
10         StringBuffer decorated = decorate();
11
12         rtn.append("<div" + (decorated != null ? decorated.toString() : "")
13                 + ">" + _innerText + "</div>");
14         return rtn;
15     }
16
17     /**
18      * 扩展点
19      *
20      * @return
21      */
22     @_Extention
23     protected StringBuffer decorate() {
24         return null;
25     }
26 }
```

divDom组件的部分源代码

5.ToyBricks Framework / 扩展点

eventDecorator组件的部分源代码

```
5  public class EventDecorator {
6
7      public StringBuffer decorate() {
8          StringBuffer rtn = new StringBuffer();
9          String onclick = onClick();
10         if (onclick != null) {
11             rtn.append(" onclick=\"" + onclick + "\"");
12         }
13         String onmouseover = onMouseover();
14         if (onmouseover != null) {
15             rtn.append(" onmouseover=\"" + onmouseover + "\"");
16         }
17         return rtn;
18     }
19
20     @_Extention
21     protected String onClick() {
22         return null;
23     }
24
25     @_Extention
26     protected String onMouseover() {
27         return null;
28     }
29
30 }
```

5.ToyBricks Framework / 扩展点

onclickDecorator组件的部分源代码

```
3  public class OnClickDecorator {
4
5      public String click() {
6          return "alert('hello')";
7      }
8  }
```

单元测试部分源代码

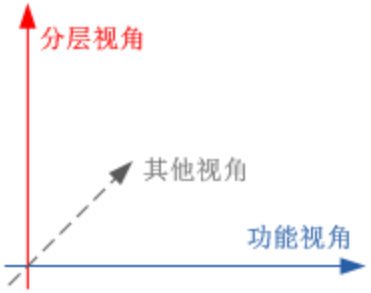
```
29     @Test
30     public void extention() {
31         StringBuffer html = ((DivDom) tb.getComponent("divDom"))
32                 .getHtml("hello");
33         System.out.println("-----ToyBricksTest.java-->" + html);
34     }
```

输出结果

```
-----ToyBricksTest.java--><div onclick="alert('hello')">hello</div>
```

百家讲坛

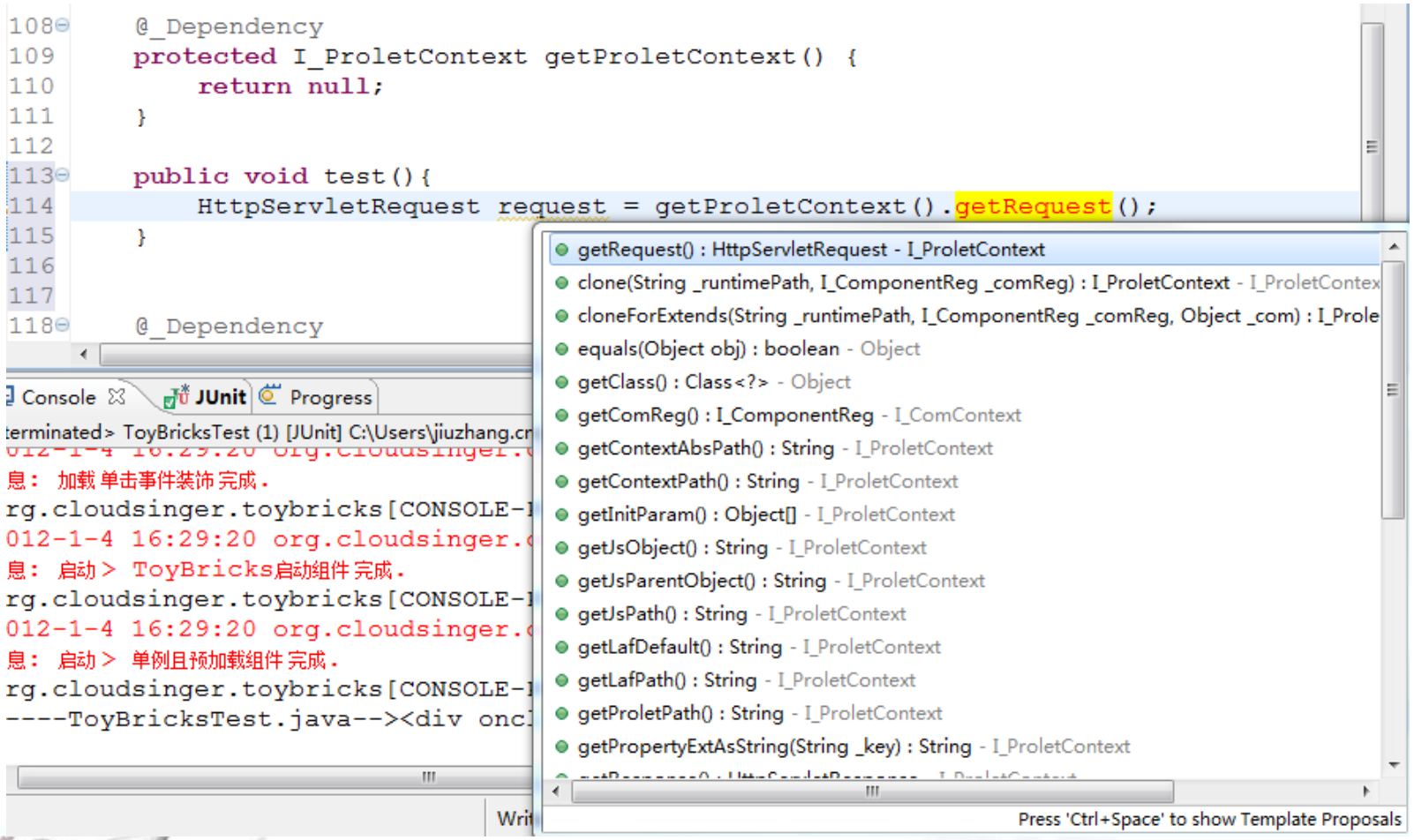
分层 VS 微内核



功能视角的解耦，一个最佳实践是构造“微内核”架构的系统，在这个方法论指导下，系统是“逐渐扩展”出来的。

5.ToyBricks Framework / 预定义的上下文对象

不同的实现中提供了许多可直接被声明为依赖项的“约定服务调用”，而且，自定义该类约定依赖项也是非常容易的。



5.ToyBricks Framework / 还有很多非常好玩的东西

代码合并的“真正多重继承”

基于RMI的组件实现

… …

6.Aquarell模板引擎

Aquarell=Freemarker+Jelly

1.目前（V1.3）已实现Freemarker绝大部分功能：

各类控制标签：if/else/elseIf for break/continue 三元表达式
表达式输出：常规/EL风格/识别方法的get/is/直接方法名 …
指令：Import/assign/include…

2.Aqarell取代Freemarker的Macro以标签代之，对标签实现了全面支持，特性如下：

基于命名空间
简单的生命周期模型
支持普通标签、集合标签、属性标签等；子标签无限级嵌套/属性标签化表示

6.Aquarell模板引擎

```
<#include "org.cloudsinger.webos.rsManager.js.html"/>
<#-- ----------------------------------------------------------------------------body菜单 -->
<#assign menuXML = newProletMenuXML/>
<cs:Menu id="containerMenu">
    <cs:menuItem id="menuItemNew" title="新建">
        <cs:xml>${menuXML}</cs:xml>
    </cs:menuItem>
    <cs:menuItem/>
    <cs:menuItem title="属性" onclick="$$RSManager.openWithDialog('cloudsinger.webos/rsMa
</cs:Menu>
<#assign plusList = contextBeanList/>
<#for plusbean:plusList>
    <cs:Menu id="${plusbean.type}"><cs:xml>${plusbean.contextMenuXml}</cs:xml></cs:Menu>
</#for>
<#-- ----------------------------------------------------------------------------主导航 -->
<#assign fpList = folderPathList/>
<cs:Toolbar id="toobar">
    <div class="navbar">
    <li>当前位置: </li>
    <#if fpList!=null>
        <#for folder:fpList>
            <a href="javascript:$$RSManager.openFolder('${folder.path}')">${folder.name}
            <span onclick="$$RSManager.createCateMenu('${folder.path}',this)" title="单击
        </#for>
    <#else>
        <a style="font-weight:normal;font-style:italic;color:gray;">对不起，没有配置"运行时文件库"，i
    </#if>
    </div>
    <cs:Button id="btnNew" title="新建" disabled="${fpList==null?'true':'false'}">
        <cs:menu>
            <cs:Menu>
                <cs:xml>${menuXML}</cs:xml>
            </cs:Menu>
        </cs:menu>
    </cs:Button>
    <cs:Button title="删除" onclick="deleteResource" disabled="true" subscribe="focusItem
</cs:Toolbar>
```

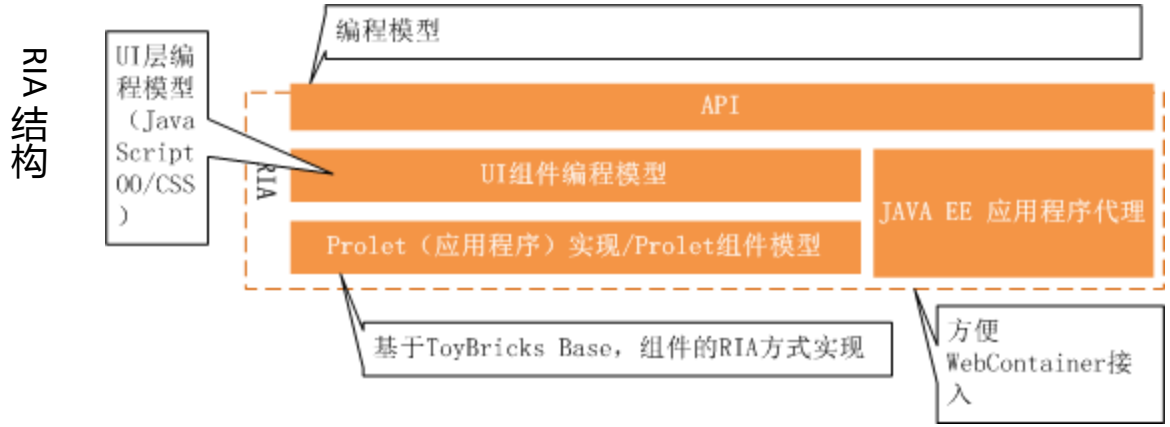
包含其他模板

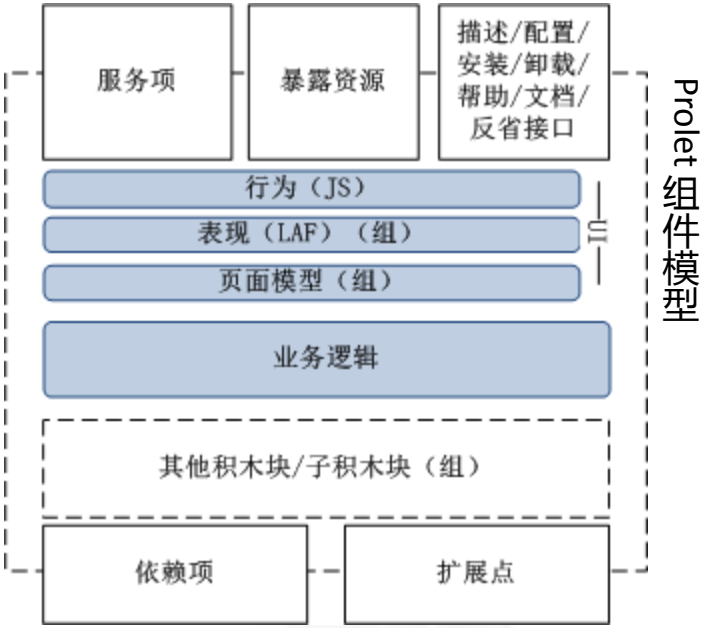
引用标签库中的标签

标签对文本（HTML）的良好嵌套支持

百家讲坛

7.ToyBricks RIA





7.ToyBricks RIA / Prolet描述文件举例

```
<?xml version="1.0" encoding="UTF-8"?>
<toybricks xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:noNamespaceSchemaLocation="http://www.cloudsinger.org/toybricks/components/java">
    <!-- 配置toybricks.config组件 -->
    <config for="toybricks.config">
        <property name="implementation" additional="true">
            <value.list>
                <value.map>
                    <value.entry key="type">
                        <!-- 解析prolet -->
                        <value.string>prolet</value.string>
                    </value.entry>
                    <value.entry key="parser">
                        <value.java-class>org.cloudsinger.ria.impl.ProletImplParser</value.java-class>
                    </value.entry>
                </value.map>
            </value.list>
        </property>
    </config>

    <component id="cloudsinger.ria" title="Toybricks-RIA(TM)核心服务"
        className="org.cloudsinger.ria.impl.RIACom" visibility="none" startup="1"
        singleton="true">
        <property name="logMode" type="type.string">
            <!-- 日志模式console|normal -->
            <value.string>normal</value.string>
        </property>
        <property name="runMode" type="type.string">
            <!-- 运行模式debug|deploy -->
            <value.string>debug</value.string>
        </property>
        <dependency name="getExtender" desc="获取对组件RIACom扩展的重要扩展点">
            <input />
            <output>
                <param>
                    <type.java-class>org.cloudsinger.ria.api.I_RIAExtention</type.java-class>
                </param>
            </output>
        </dependency>
```

7.ToyBricks RIA / Prolet描述文件举例

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <toybricks xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance
3      xsi:noNamespaceSchemaLocation="http://www.cloudsinger
4      <component.prolet id="rsManager" title="资源管理器"
5          viewFile.cur="rsManager.aq.html" className.cur="RSManagerBiz"
6          icon16.cur="icon16.gif" icon32.cur="icon32.gif" parent="cloudsinger.webos"
7          extends="cloudsinger.webos/$rsManagerBase" lafPath.curView="$laf/$"
8          visibility="all">
9
10         <!-- 文件夹 -->
11         <sub-component.prolet id="folder" title="文件夹">
12             <sub-component.prolet id="new" title="新建文件夹"
13                 viewFile.cur="folder/new.aq.html" className.cur
14                 icon16.cur="folder/icon16.gif" />
15             <sub-component.prolet id="property" title="文件夹属性
16                 viewFile.cur="folder/property.aq.html" className.cur="folder/PropertyBiz"
17                 icon16.cur="folder/icon16.gif" />
18         </sub-component.prolet>
19
20         <!-- 文件-->
21         <sub-component.prolet id="uploadFile" title="文件">
22             <sub-component.prolet id="upload" title="上传文件"
23                 viewFile.cur="uploadFile/upload.aq.html" className.cur="uploadFile/UploadBiz"
24                 icon16.cur="uploadFile/icon16.gif" lafPath.curView="$laf/$" />
25             <sub-component.prolet id="download" title="下载|打开文件"
26                 className.cur="uploadFile/DownloadBiz" icon16.cur="uploadFile/icon16.gif" />
27             <sub-component.prolet id="property" title="文件属性"
28                 viewFile.cur="uploadFile/property.aq.html" className.cur="uploadFile/PropertyBiz"
29                 icon16.cur="uploadFile/icon16.gif" />
30         </sub-component.prolet>
```

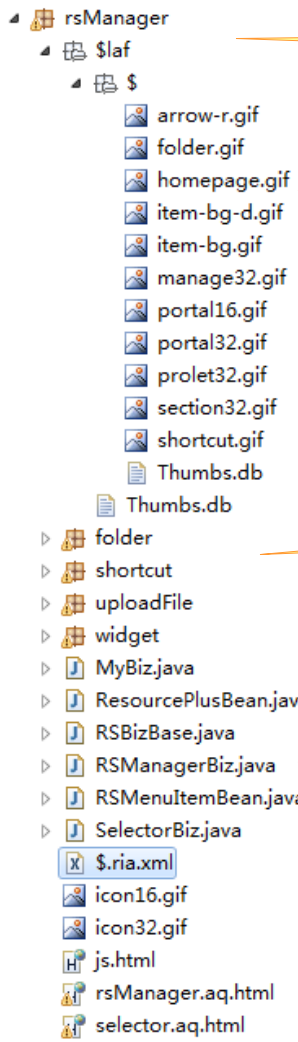
视图文件位置

业务逻辑位置

Laf（Look And Feel）位置

百家讲坛

7.ToyBricks RIA / Prolet源文件举例



所有的文件都在同一个包中，所以该Prolet（组件）可以直接以jar的方式发布

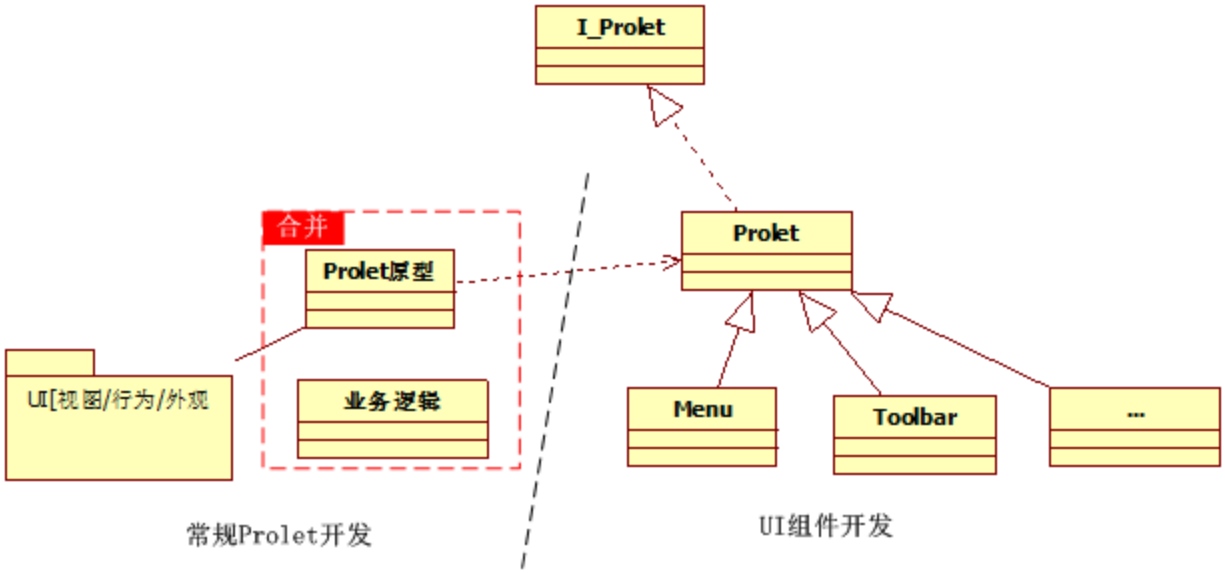
外观文件（图片/CSS等），RIA支持多套外观

子程序所在的包

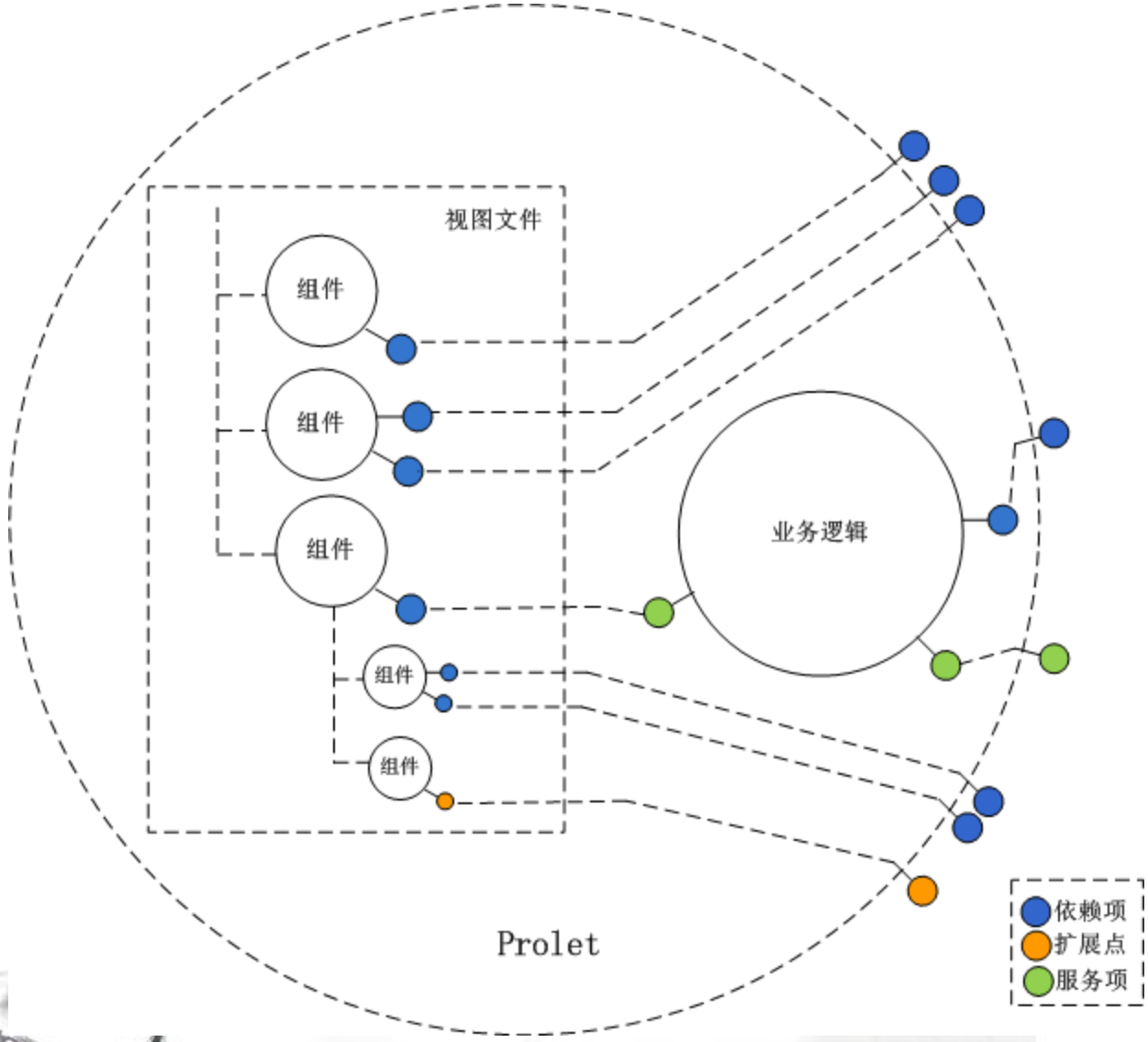
业务逻辑

视图文件

7.ToyBricks RIA / Prolet组件实现原理



7.ToyBricks RIA / 组件组合后的依赖及扩展叠加



7.ToyBricks RIA / 前端解决方案

标准的HTML/CSS/JavaScript代码
前端JS API

当前Component后台服务 API

（面向OO的）Component API

Jquery（PlugIn扩展） API

```
90 <script>
91     var $$RSManager = {
92         container : null,
93         //-------------------------------------------------------------------------------
94         menuCache : {},
95         createCateMenu : function(_cateId,_img){
96             if(this.menuCache[_cateId]!=undefined){
97                 this.menuCache[_cateId].show();
98             }else{
99                 var desc = ${jsObject}.ajax("getSubFolderList",[_cateId]);
100                var html;
101                if(desc){
102                    var ary = desc.split(";"),tAry;
103                    var html = "<div class='menu'>";
104                    for(var i=0;i<ary.length;i++){
105                        tAry = ary[i].split(":");
106                        html += "<table class='item' onclick=\"$$RSManager.openFol
107                        +"')\"><tr><td style='width:25px;text-align:center'><img s
108                        html += "<td>"+unescape(tAry[1])+"</td></tr></table>";
109                    }
```

7.ToyBricks RIA / 前端解决方案

前端调用后台服务：

1

```
36      public String getRSMessage(String _menuId,int _counter){
37          //...
38          return null;
39      }
```

2

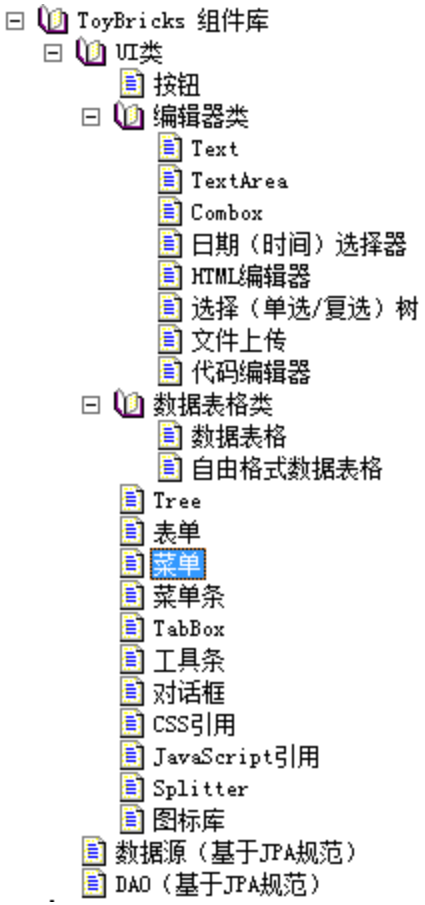
```
12          <service name="getRSMessage" bind="getRSMessage">
13              <input>
14                  <param type="type.string"/>
15                  <param type="type.int"/>
16              </input>
17              <output>
18                  <param type="type.string"/>
19              </output>
20          </service>
```

3

```
92  <script>
93      var msg = ${window}.ajax.getRSMessage("${params[0]}",12);
94      alert(msg);
95
96      var msg = ${window}.ajax("getRSMessage",["${params[0]}",12]);
97      alert(msg);
98  </script>
```

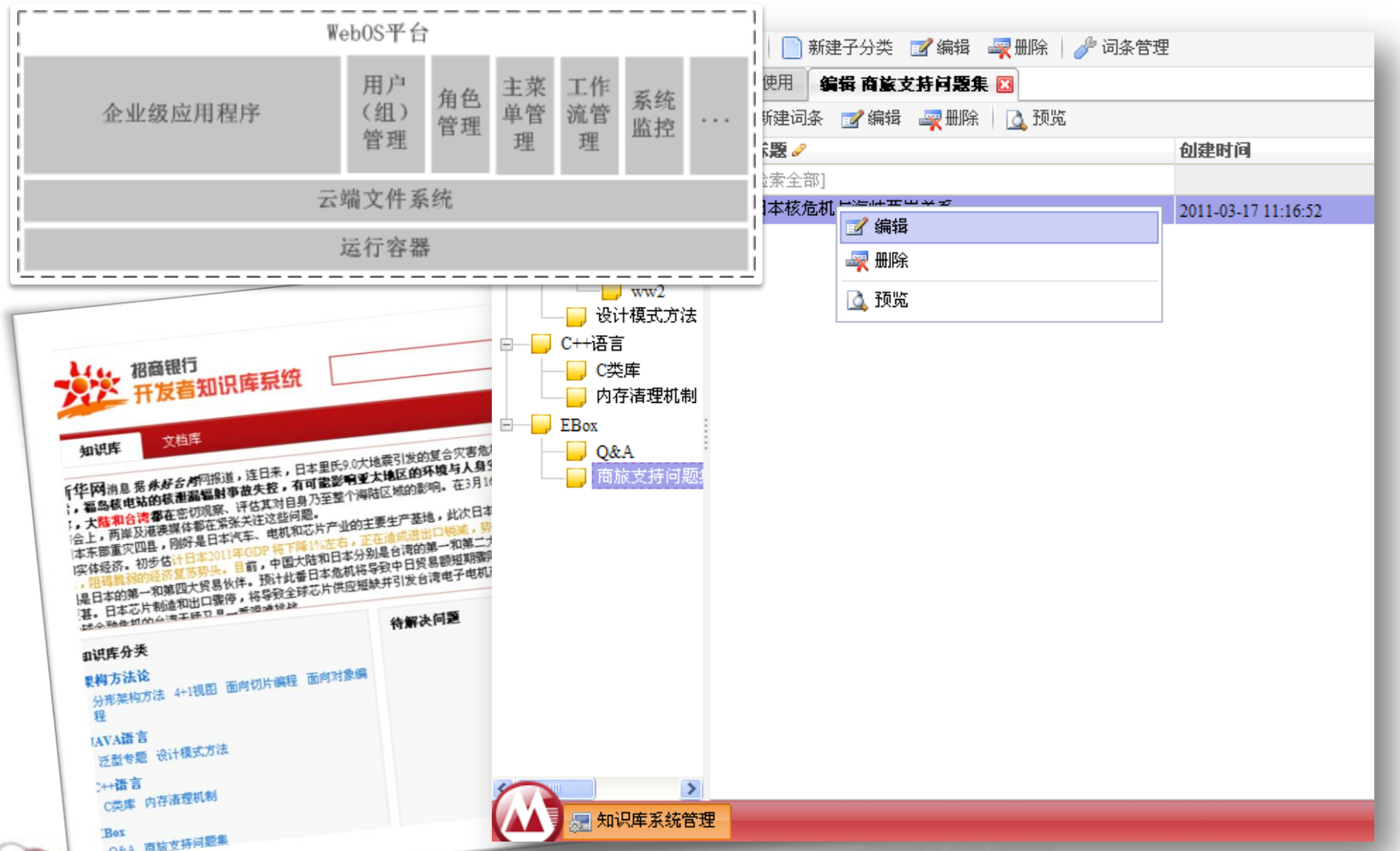
8.ToyBricks 组件库

当前版本（ToyBricksV4.0）提供的组件库



9.ToyBricks WebOS

基于ToyBricks RIA的WebOS实现



http://cloudsinger.iteye.com

与您一同成长

致 谢