ESUG 2000

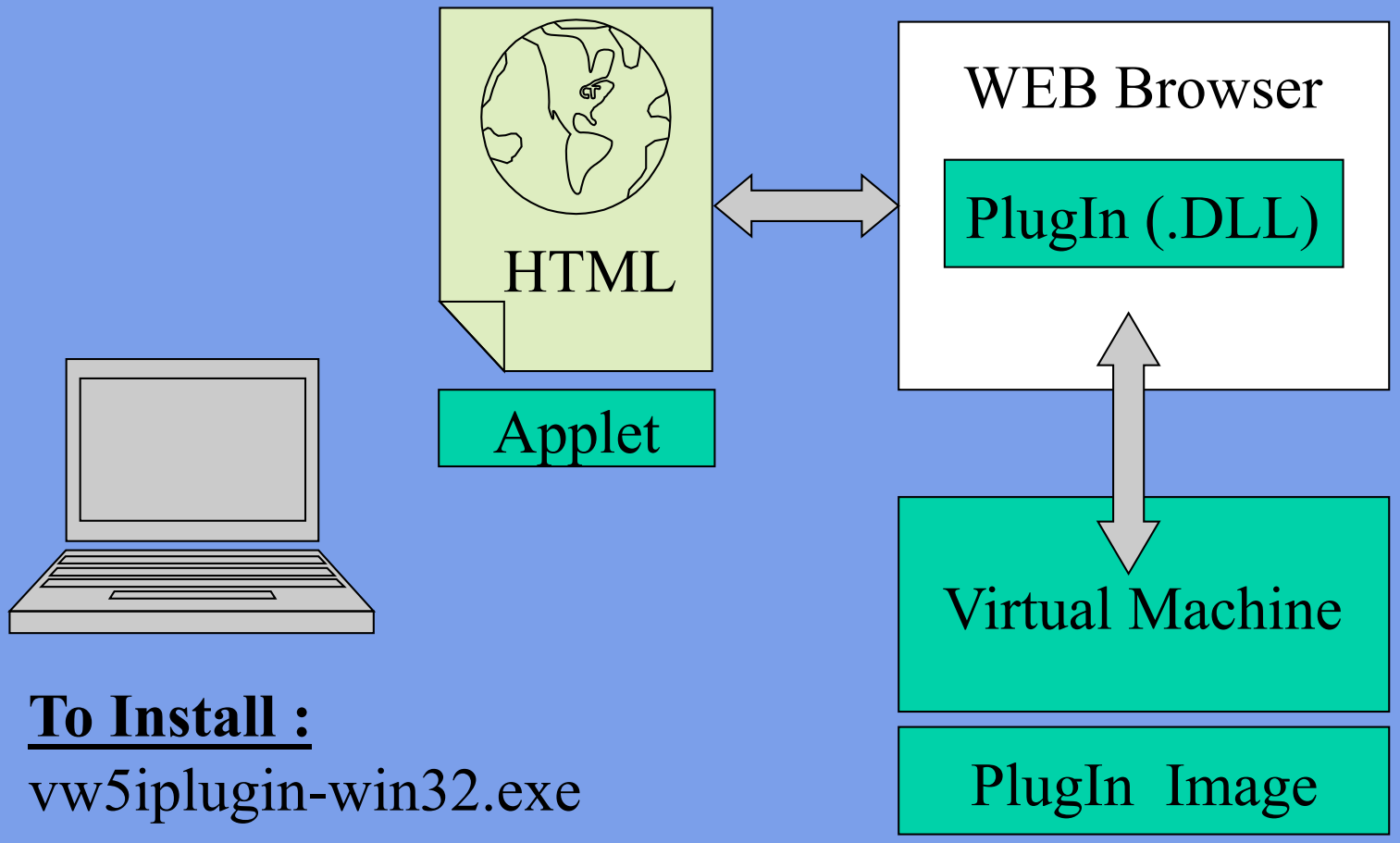
When Smalltalk Meets the WEB

Juan Carlos Cruz
Valtech AG, Zurich

Solutions

- VisualWave™ - Cincom
- VisualWorks plugIn™ - Cincom
- Classic Blend 2.0 ™ - Applied Reasoning
- Classic Blend 3.0 ™ - Applied Reasoning
- VisualAge ULC - IBM
- CORBA - OMG
- RMI-IIOP - Sun

The VisualWorks plugIn



To Install :

vw5iplugin-win32.exe

The plugIn Development Environment (PDE)

- Generate Smalltalk applets
- Generate a customize plugIn image
- Debug Smalltalk web-based applications

PluginDev.pcl

Developing a PlugIn Applet

- Install the PDE in your development image

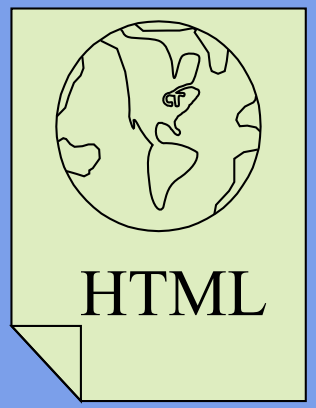
  Tools->Load Parcel Named… PluginDev.pcl
- Define your applet as Subclass of **AppletModel**
- Create a parcel containing your applet
- Save the parcel as an applet

  Parcel -> save…

Running an Applet in a Browser

Calculator.html

```
<HTML>
<HEAD> </HEAD>
 <BODY> <P> MyTest </P>
<EMBED SRC="Calculator.pcl"
    VWOPEN= "CalculatorExample"
    WIDTH="233" HEIGHT = "245"
    TYPE ="application/x-visualworks-parcel">
</BODY>
</HTML>
```



Do it

Adapting an Existing Application

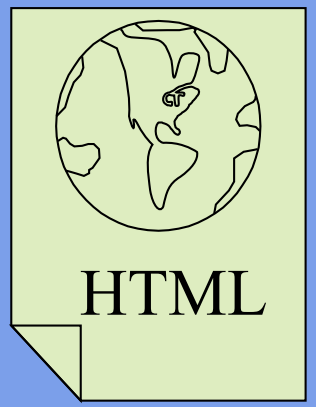
- Duplicate a subset of the AppletModel protocol in your application class
- File in… applet-api.st
- The essential protocol is:
  - AppletModel >> **isAppletModel**
  - AppletModel >> **pluginConnection:**
  - AppletModel >> **pluginConnection**

Using the plugIn debugger

- Start a plugIn debug session

  Tools -> Plug-in Debug tool
- Click **Enable Debug**
- Open the HTML page containing the applet in a web browser
- Click **Connect**
- Debug as you normally do …
- Click **Disable Debug** when you have finished

Adding an Application to a Web Page



<**EMBED**
SRC="Calculator.pcl"
VWOPEN= "CalculatorExample"
WIDTH="233" **HEIGHT** = "245"
TYPE ="application/x-visualworks-parcel">

Attributes:

SRC	The parcel containing the applet code
WIDTH	The width of the applet window
HEIGHT	The height of the applet window
TYPE	MIME type for the plugIn applet
VWOPEN	Class containing window specification
VWPRELOAD	A list of parcel file names to be loaded

Building a Custom plugIn Image

- Standart plug-in image is `plugin-base.im`
- For larger applications include much of the application in either:
  - a custom-image or
  - as additional parcels (invoked by the **VWPRELOAD** attribute in the **EMBED** tag)
- Use Runtime Packager™ to build a custom-image. Parameters file: `vwplugin.rtp`

PlugIn Initialization File
VWPLUGIN.INI

Attributes:

DIRECTORY	Current directory for launching VisualWorks
OBJECTENGINE	Name of the OE
BASEIMAGE	Name of the image to load
APPLICATION	Value matching the **VWAPPL** tag
EXTRA	Extra information of interest to the application

Security Mesures: "Trusted Site Strategy"

ALLOW	< all | local | hostID >
DENY	< all | local | hostID >

-By default if a site is not listed it is **ALLOW**ed. To switch begin sequence with **DENY ALL**.
-Parcels are downloaded only if their sites are **ALLOW**ed.

Communicating With a PlugIn Application

- HTTP GET and POST operations.

GET

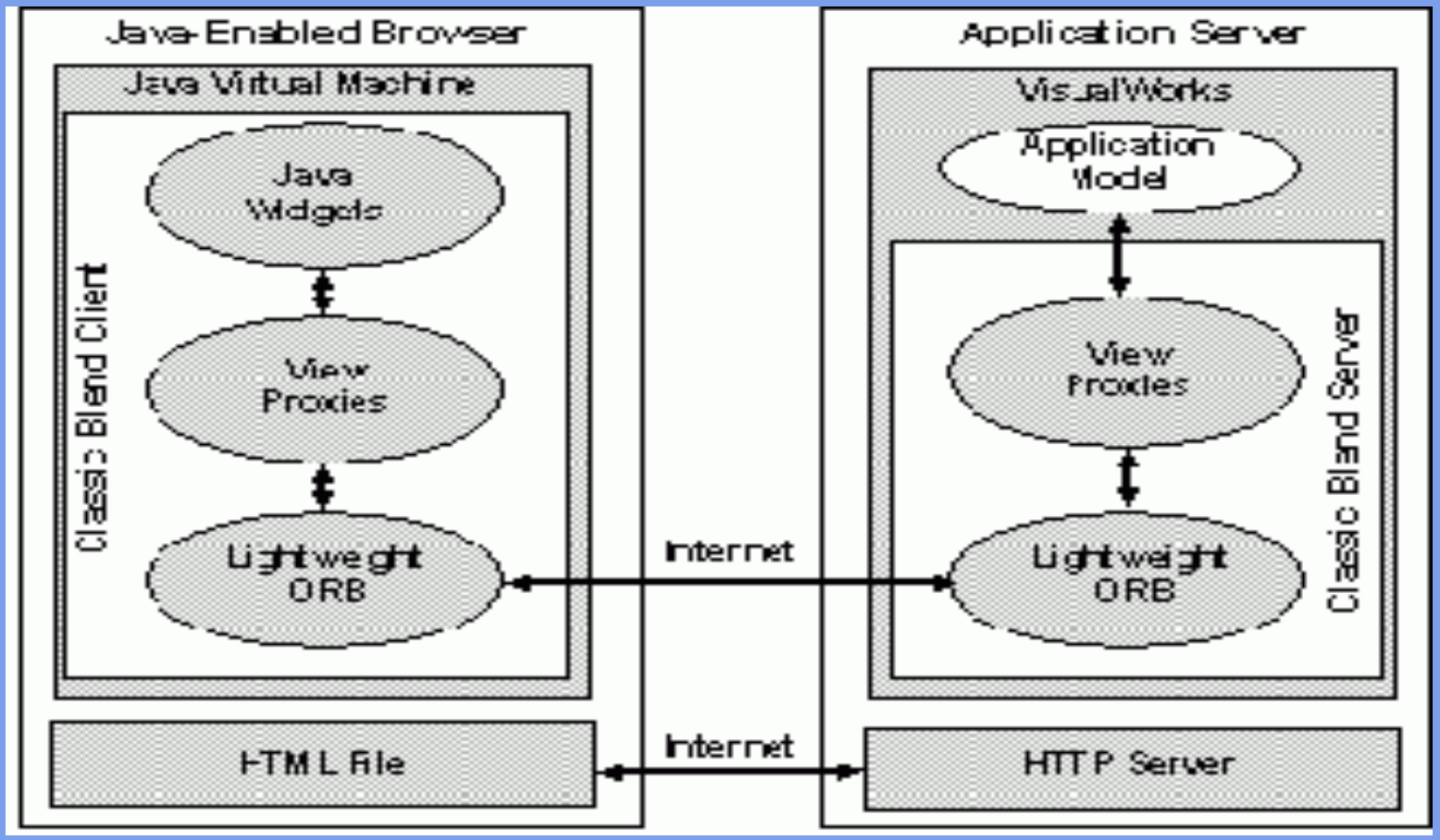
```
self getURL: url target: 'target'
self getURLAsString: url    …
```

POST

```
self postToURL: url file: aFile
self postToURL: url string: aString    …
```

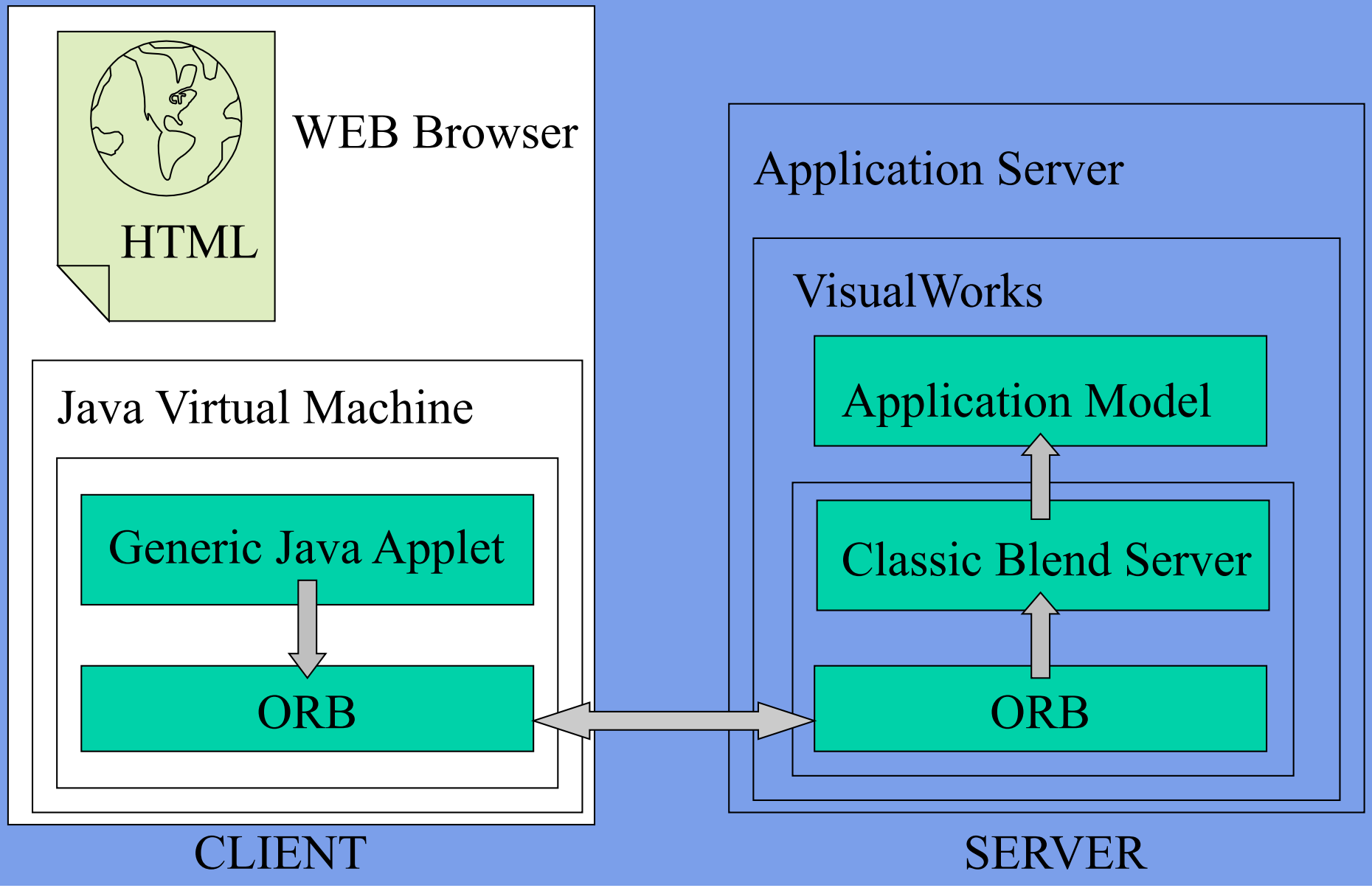
- Sockets
- Database Connect, DST, VisualWave

Classic Blend 2.0

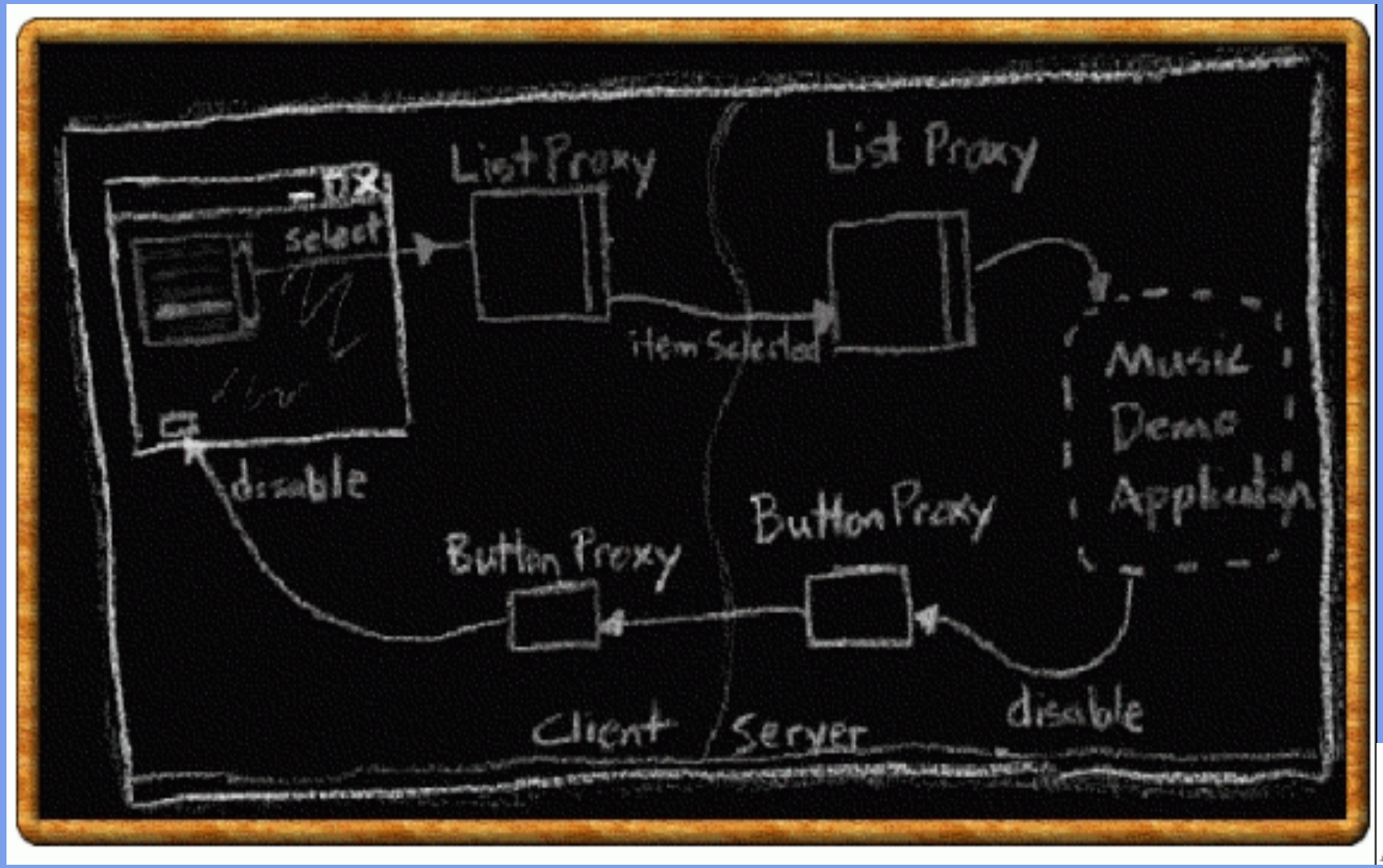


Architecture: (1) a generic Java applet on the client, (2) a portable application server, and (3) a lightweigth ORB

How CB works ?

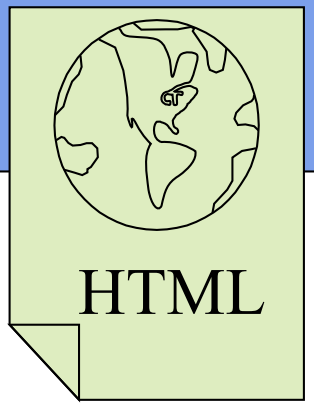

WEB Browser
HTML
Java Virtual Machine
Generic Java Applet
ORB
CLIENT
Application Server
VisualWorks
Application Model
Classic Blend Server
ORB
SERVER

How CB Works ?



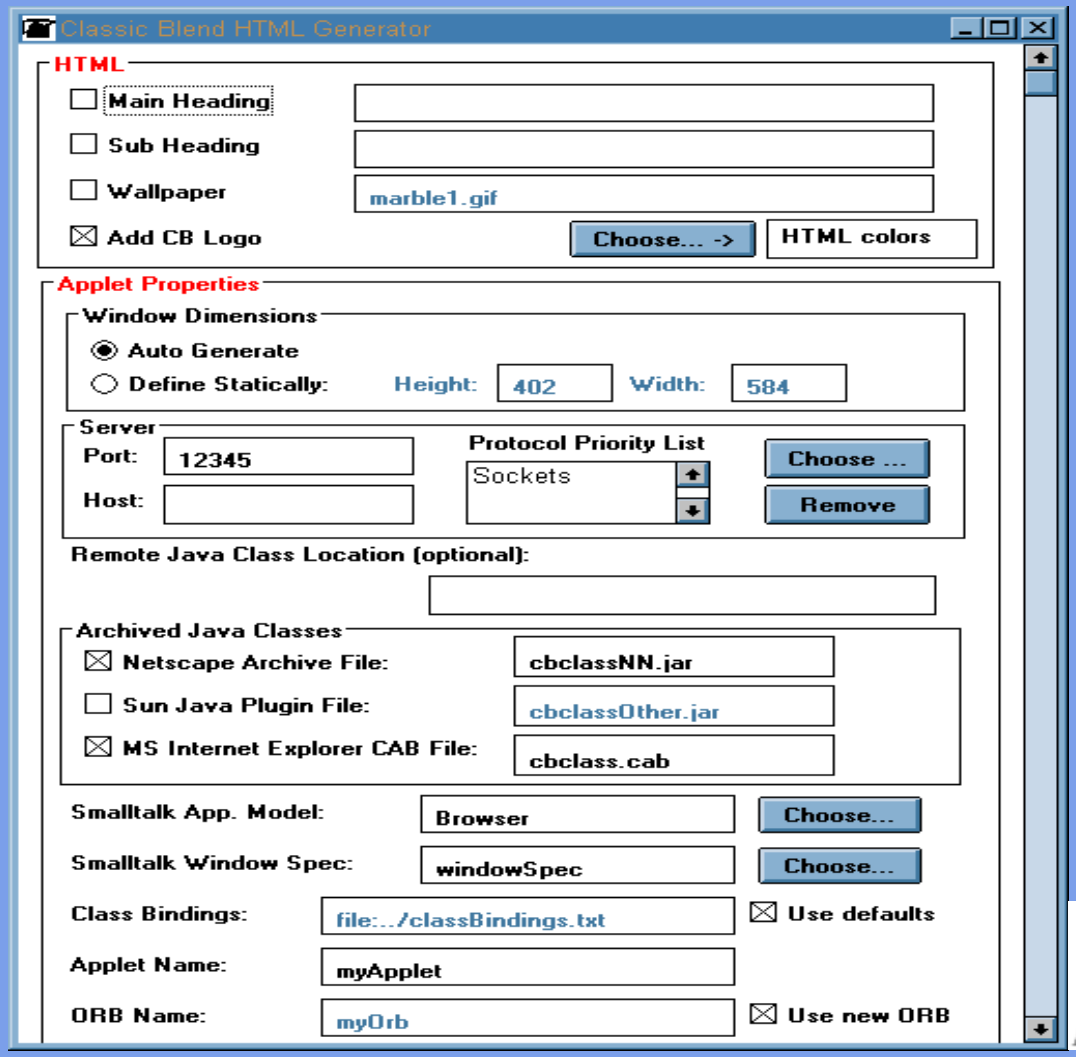
AppliedReasoning

HTML APPLET tag

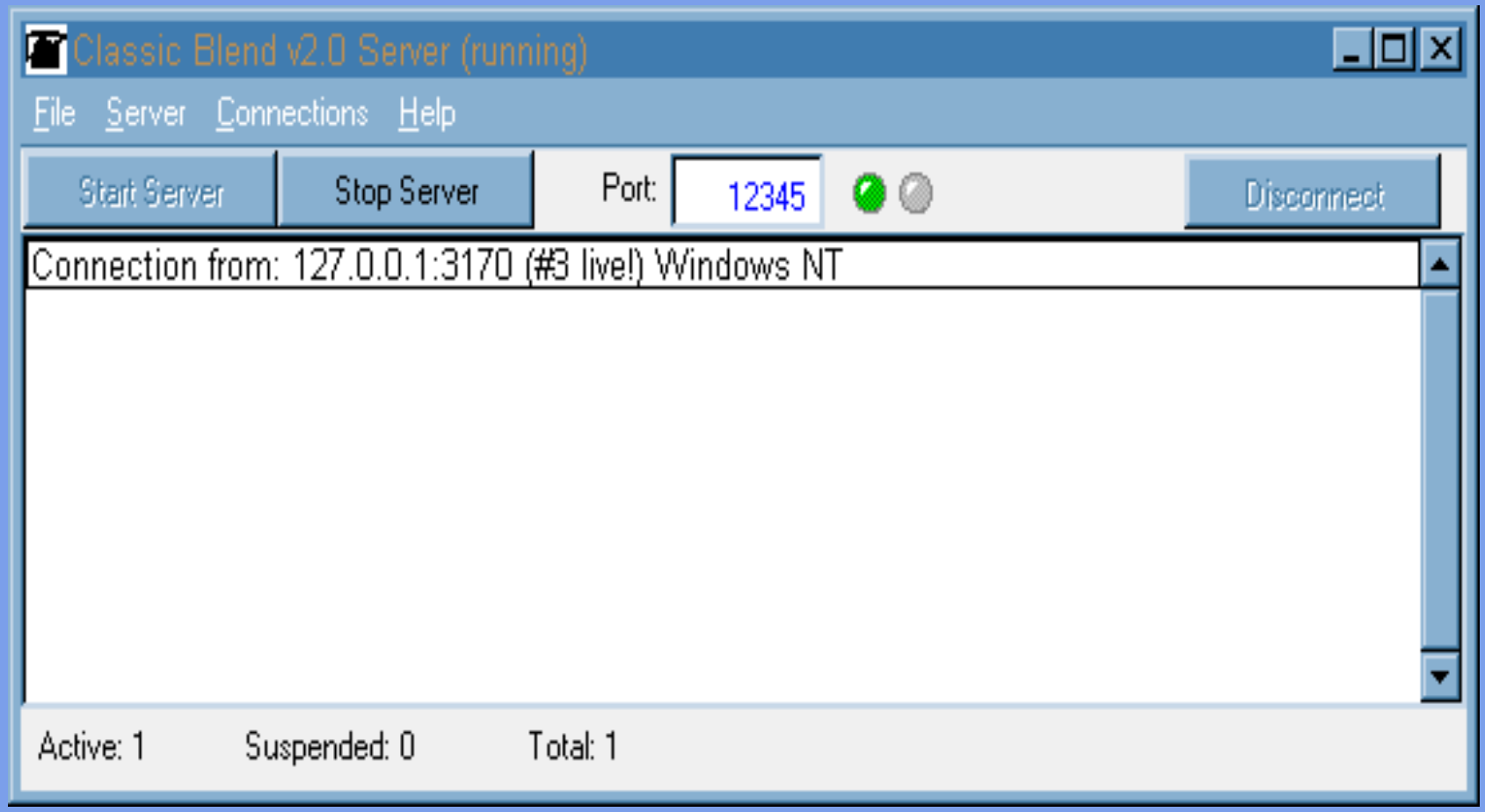


<APPLET
  code = com.arscorp.cb.icf.support.CbApplet
  codebase = <absolute URL for CbApplet>
  width = <pixel width> height= <pixel height>
  archive = <path to JAR files> ex: “cbClassNN.jar”>”
 <**PARAM** name “server” value = “socket://<hostA: portA>;
  http://<hostB: portB>”>
 <**PARAM** name “appletType” value = “<Smalltalk-class>”>
 <**PARAM** name ‘specName” value = “<window-spec-method>”
 ..
</**APPLET**>

HTML Generator Tool



Classic Blend Server



13. Appendix A: Unsupported VisualWorks features

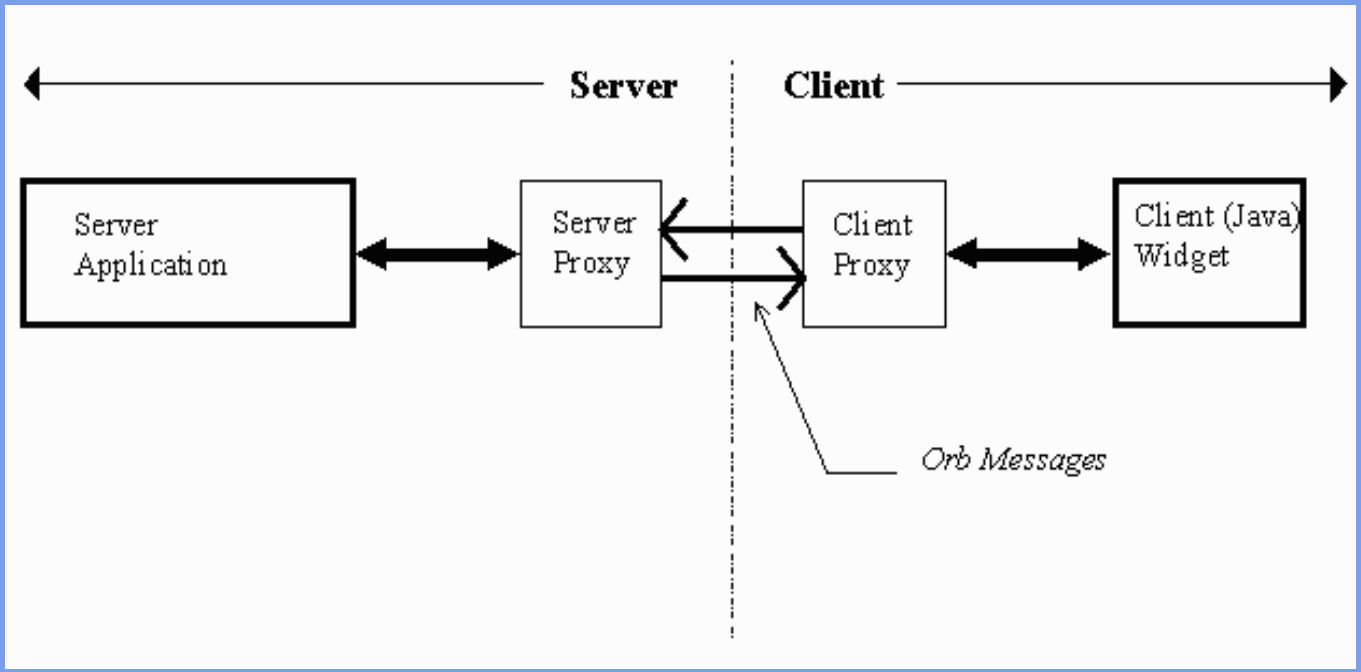
All Widgets

CBDevelopersGuide 2.0

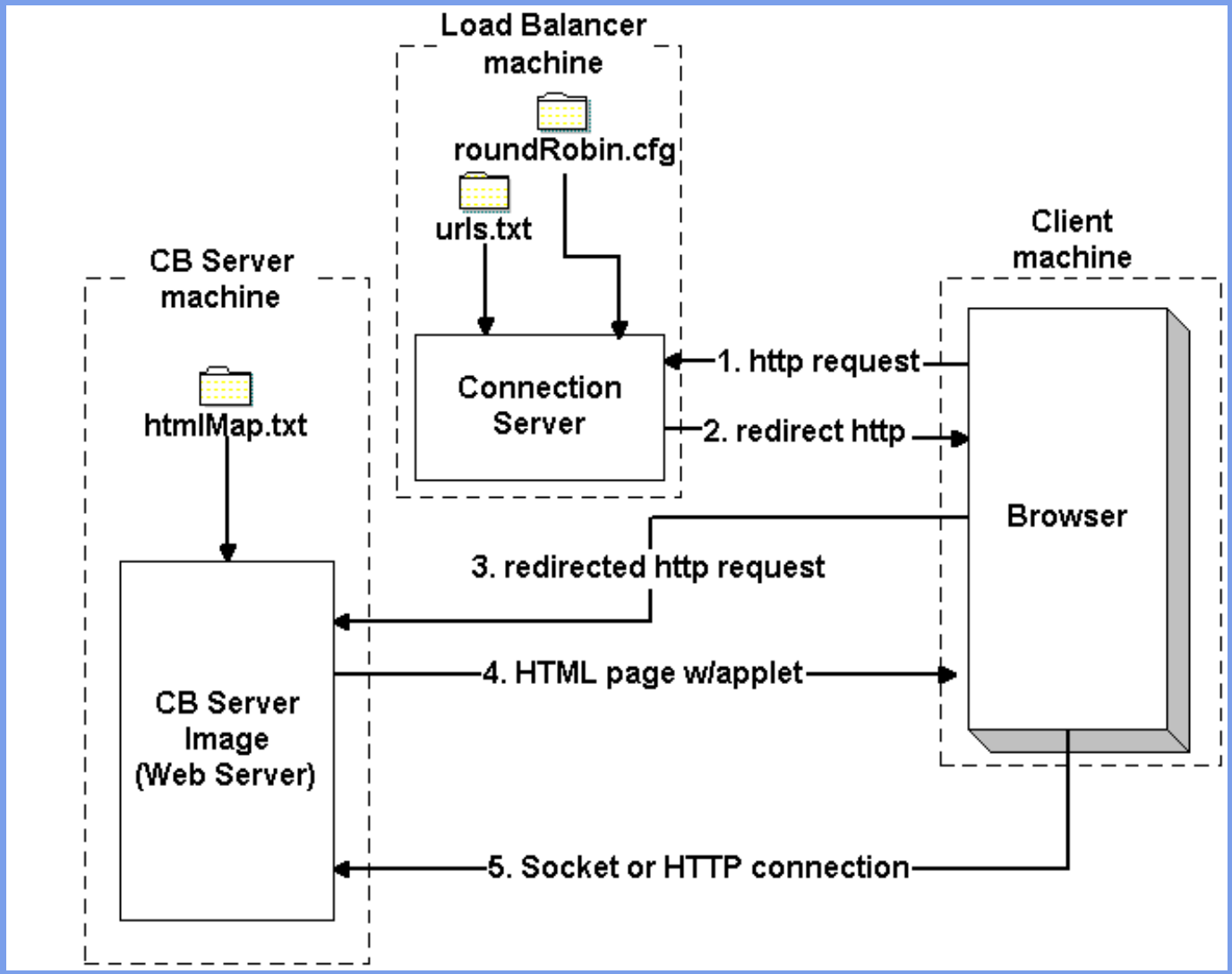
- drag and drop
- entry and exit notification
- entry and exit validation
- selection foreground and background color
- full text attributes
- dynamic focus control (#hasFocus: and its derivates)
- border decoration policies

…. etc

Designing and Implementing Custom Widgets



Classic Blend Load Balancer

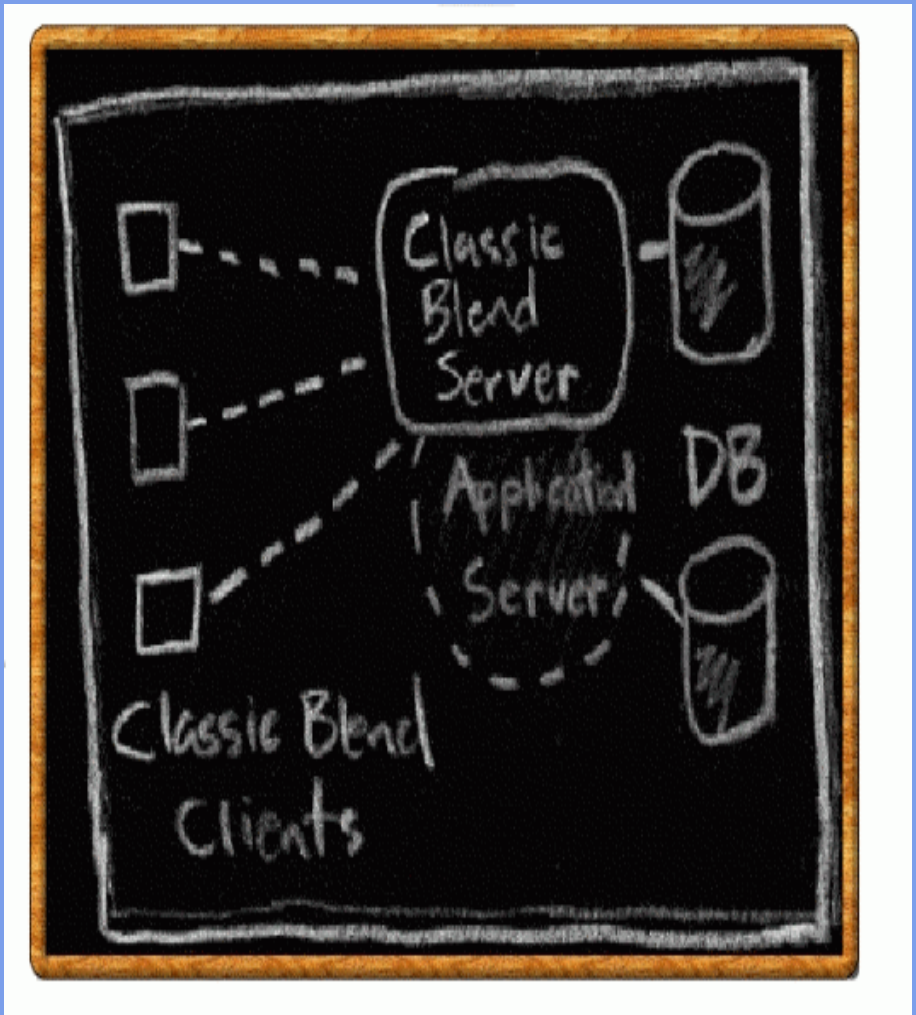


AppliedReasoning

What is the difference between CB 2.0 and CB 3.0

- Include Java and VisualAge Smalltalk as server platforms
- Classic Blend is bean-based. Client GUIs can be constructed in any bean-compliant GUI builder
- Flexibility to add any custom code or validation they want to their clients

Classic Blend 3.0



- Client CB Server

  cbClient.jar

- Install Server

  Java

  cbServer.jar
  serverSrc.zip

  Smalltalk

  cb30st.dat /
  otiadds.pcl+cb30vw.pcl

APPLIEDREASONING

Building the Java GUI

```
package com.arscorp.demos.colorTest;
import java.awt.*;
import java.awt.event.*;
import com.arscorp.cbng.client.*;
import com.arscorp.cbng.client.builder.*;

public class ColorTest extends CbBeanContainer implements
    InitializeListener {
    protected CbFrame _cbFrame;
    protected Panel _panel = null;
    protected Presenter _presenter;
    protected Button _buttonRed = null;
    protected Button _buttonBlue = null;
}
```

Building the Java GUI

```
public void buildWidgets() {
    _cbFrame.setName("CbFrame");
    _panel.setName("ButtonPanel");
    _cbFrame.setServerObject("ColorTestPresenter");
    _cbFrame.setSessionName("ColorServer");
    _cbFrame.setLayout(new GridLayout());
    _cbFrame.setBounds(116, 255, 538, 65);
    _buttonRed = new Button("Red");
    _buttonRed.setName("RedButton");
    _buttonRed.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent ae) {
                    _presenter.sendMessage(ae, "redButtonClicked");
            }
    });
    …

    _cbFrame.add(_panel, BorderLayout.NORTH);
    _panel.add(_buttonRed);
    _panel.add(_buttonBlue);
}
```

Creating the SM Server

- Create a subclass of **ArcbPresenter** named ColorTestPresenter
- Create a service method

```
changePanelColorTo: newColor
    | buttonPanel |
    buttonPanel := self getPanelNamed:'ButtonPanel'.
    buttonPanel setBackground: newColor.
    buttonPanel repaint.
```

- Add a method name

```
redButtonClicked
    self changePanelColorTo: ColorValue red.
    ^nil
```

HTML File

```
<html>
<body>
<applet code="com.arscorp.cbng.client.CbApplet" align="baseline"
width="350" height="60">
<param name="BeanClass" value="com.arscorp.demos.colorTest.ColorTest">
<param name="SkipMessage" value="false">
<param name="LaunchButtonText" value="Press to start the Color Test">
<param name="PropertiesURL" value="file://127.0.0.1/c:\\cbClient\
\clientProperties.txt">
</applet>
</body>
</html>
```

ATTRIBUTES

code=com.arscorp.cbng.client.CbApplet
<**PARAM** name="BeanClass" value="[class-of-application-window]">
<**PARAM** name="LaunchMessage" value="[text-of-launch-message]">

References

Technology

- VW plugIn: www.cincom.com
- ClassicBlend: www.appliedreasoning.com
- VAST-ULC: www.ibm.com

Consulting

- Valtech: www.valtech.com