

# Embedding native ActiveX Controls

This topic is "hot". - Please do not misunderstand: embedding ActiveX controls into a page is not something what we recommend. This is a documentation HOW to do it - this is no a documentation THAT you should do it!

## When to think about it

Casabac produces HTML pages that use Javascript as active scripting language in the browser. There are a lot of controls coming with Casabac based on HTML and Javascript - so where's the need for embedding native objects?

The situations we know are:

- Integration of existing "special controls": maybe there are some special controls, e.g. in the area of image processing or in the area of high speed spread processing. In this case HTML with Javascript cannot compete with native implementations - both because of functional restrictions (we would not recommend to write CAD programs on base of HTML and Javascript...) and because of performance restrictions.
- Integration of sub-devices. If you want to integrate input devices like barcode scanners, digital photo cameras, etc. into your browser client then Javascript does not offer any functions to break out of the functional sandbox that is defined by Javascript: for example you cannot access hardware interfaces of the client computer. If you want to listen to a serial interface, firing barcode reader signals to the client then there is no way 'round to embedding non-Javascript-software.

Of course there are big disadvantages:

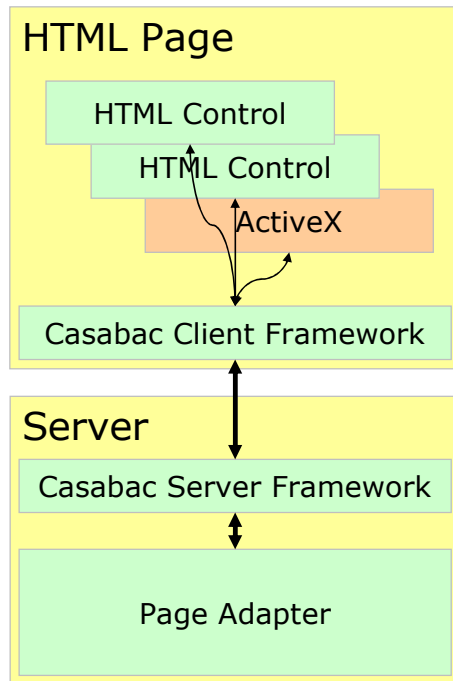
- You loose the client's platform independency. You are bound to Windows platforms.
- You loose your "zero installation" model. You have to install native components onto the client.
- You loose your "guaranteed sandbox". ActiveX components can do "anything" once they are started...

Finally it's you to decide if to use native components. Maybe you find solutions in which "some" of your users have "heavy clients with native components" (e.g. at workplaces with sub-devices) and the "big majority" of users has browser-only clients.

## Technical Overview

Technically, it's rather simple:

- ActiveX components can be embedded into HTML pages by using an <OBJECT>-tag.
- From JavaScript you can access ActiveX properties, methods and events.
- By using the Casabac control concept (read "Control Developer's Guide" for detailed information) you can create a control that generates the required HTML and Javascript. The Javascript accesses the Casabac client framework (Javascript library functions) that e.g. allow to set adapter properties and to call methods in adapters.
- The ActiveX control uses the Casabac data channel transferring net data from the browser client to the server (and back) in order to send and receive its data.
- The control can be embedded into the Layout Painter definitions to make it available as new control to your application developers.



You see that the ActiveX control are brought into Casabac pages by using Casabac controls. This means your application developers do not see anything of the ActiveX control (well, they have to install - but that's it...). For them it's just another control operating as normal Casabac control.

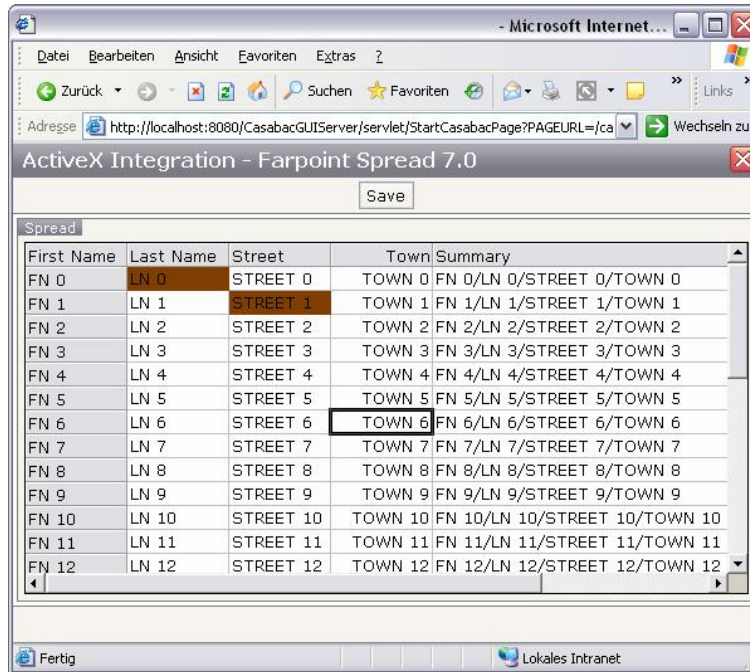
Please also note that the ActiveX component is running on client only and does not have any implications on the server side processing.

## Example

In the following example we embed an ActiveX grid processing "huge grid data". The ActiveX grid is produced by the company FarPoint, in the example version 7.0 of the "FarPoint Spread" is used. We assume that you already installed the control, e.g. a 30 day trial version that you can obtain via <http://www.fpoint.com>.

## Goal

The following screen shot shows what the goal of this chapter is: having a page in which the control is embedded, and in which the control is filled with data - coming from a normal Casabac adapter object.



You do not need to follow each coding line you will see in the following text. But you should somehow determine the principal way how to do things.

## Writing the Casabac Control

First you need to define a new Casabac control, which internally embeds the ActiveX Spread. The definition is done in the following steps:

- (1) Write the control handler, i.e. the Java class implementing the ITagHandler interface. This interface is base of all Casabac's Web Controls.
- (2) Write the server side representation of the control.
- (3) Register the control in Casabac's design time environment.

The control handler's code looks as follows:

```
package com.casabac.demolibrary;

import org.xml.sax.AttributeList;

import com.casabac.gui.generate.ITagHandler;
import com.casabac.gui.protocol.ProtocolItem;

/**
 * This controls is an encapsulation of the FarPoint Spread Control
 * which is available as ActiveX Control.
 */
public class FPOINTSPREADHandler
    implements ITagHandler
{
    String m_spreadprop;
    String m_height;
    String m_width;

    public void generateHTMLForStartTag(int id, String tagName,
        AttributeList attrlist, ITagHandler[] handlersAbove,
        StringBuffer sb, ProtocolItem protocolItem)
    {
        // read attributes
        m_spreadprop = attrlist.getValue("spreadprop");
        m_height = attrlist.getValue("height");
    }
}
```

```

m_width = attrlist.getValue("width");
if (m_height == null) m_height = "100";
if (m_width == null) m_width = "100";
// fill protocol
protocolItem.addProperty(m_spreadprop, "SPREADInfo");
protocolItem.addProperty(m_spreadprop + ".titles");
protocolItem.addProperty(m_spreadprop + ".widths");
protocolItem.addProperty(m_spreadprop + ".items[*].values");
protocolItem.addProperty(m_spreadprop + ".commands[*].bc");
protocolItem.addProperty(m_spreadprop + ".commands[*].x");
protocolItem.addProperty(m_spreadprop + ".commands[*].y");
protocolItem.addJsLib("../casabacdemos/pages/FPOINTSREADControl.s.js");
// append control
sb.append("<td>\n");
sb.append("<OBJECT height='"+m_height+"' width='"+m_width+"' id='FP'+id+'
classid='clsid:7114683A-020D-4D16-80FD-6ACE384B66DF'>"+
"<PARAM NAME='_Version' VALUE='458752'>"+
"<PARAM NAME='_ExtentX' VALUE='13070'>"+
"<PARAM NAME='_ExtentY' VALUE='5874'>"+
"<PARAM NAME='_StockProps' VALUE='64'>"+
"<PARAM NAME='Enabled' VALUE='1'>"+
"<PARAM NAME='AllowCellOverflow' VALUE='0'>"+
"<PARAM NAME='AllowDragDrop' VALUE='0'>"+
"<PARAM NAME='AllowMultiBlocks' VALUE='0'>"+
"<PARAM NAME='AllowUserFormulas' VALUE='0'>"+
"<PARAM NAME='ArrowsExtEditMode' VALUE='0'>"+
"<PARAM NAME='AutoCalc' VALUE='1'>"+
"<PARAM NAME='AutoClipboard' VALUE='1'>"+
"<PARAM NAME='AutoSize' VALUE='0'>"+
"<PARAM NAME='BackColorStyle' VALUE='0'>"+
"<PARAM NAME='BorderStyle' VALUE='1'>"+
"<PARAM NAME='ButtonDrawMode' VALUE='0'>"+
"<PARAM NAME='ColHeaderDisplay' VALUE='2'>"+
"<PARAM NAME='ColsFrozen' VALUE='0'>"+
"<PARAM NAME='DAutoCellTypes' VALUE='1'>"+
"<PARAM NAME='DAutoFill' VALUE='1'>"+
"<PARAM NAME='DAutoHeadings' VALUE='1'>"+
"<PARAM NAME='DAutoSave' VALUE='1'>"+
"<PARAM NAME='DAutoSizeCols' VALUE='2'>"+
"<PARAM NAME='DInformActiveRowChange' VALUE='1'>"+
"<PARAM NAME='DisplayColHeaders' VALUE='1'>"+
"<PARAM NAME='DisplayRowHeaders' VALUE='1'>"+
"<PARAM NAME='EditEnterAction' VALUE='0'>"+
"<PARAM NAME='EditModePermanent' VALUE='0'>"+
"<PARAM NAME='EditModeReplace' VALUE='0'>"+
"<PARAM NAME='FormulaSync' VALUE='1'>"+
"<PARAM NAME='GrayAreaBackColor' VALUE='12632256'>"+
"<PARAM NAME='GridColor' VALUE='12632256'>"+
"<PARAM NAME='GridShowHoriz' VALUE='1'>"+
"<PARAM NAME='GridShowVert' VALUE='1'>"+
"<PARAM NAME='GridSolid' VALUE='1'>"+
"<PARAM NAME='MaxCols' VALUE='500'>"+
"<PARAM NAME='MaxRows' VALUE='500'>"+
"<PARAM NAME='MoveActiveOnFocus' VALUE='1'>"+
"<PARAM NAME='NoBeep' VALUE='0'>"+
"<PARAM NAME='NoBorder' VALUE='0'>"+
"<PARAM NAME='OperationMode' VALUE='0'>"+
"<PARAM NAME='Position' VALUE='0'>"+
"<PARAM NAME='ProcessTab' VALUE='0'>"+
"<PARAM NAME='Protect' VALUE='1'>"+
"<PARAM NAME='Redraw' VALUE='1'>"+
"<PARAM NAME='RestrictCols' VALUE='0'>"+
"<PARAM NAME='RestrictRows' VALUE='0'>"+
"<PARAM NAME='RetainSelBlock' VALUE='1'>"+
"<PARAM NAME='RowHeaderDisplay' VALUE='1'>"+
"<PARAM NAME='RowsFrozen' VALUE='0'>"+
"<PARAM NAME='ScrollBarExtMode' VALUE='0'>"+
"<PARAM NAME='ScrollBarMaxAlign' VALUE='1'>"+
"<PARAM NAME='ScrollBars' VALUE='3'>"+
"<PARAM NAME='ScrollBarShowMax' VALUE='1'>"+
"<PARAM NAME='SelectBlockOptions' VALUE='15'>"+
"<PARAM NAME='ShadowColor' VALUE='-2147483633'>"+
"<PARAM NAME='ShadowDark' VALUE='-2147483632'>"+
"<PARAM NAME='ShadowText' VALUE='-2147483630'>"+
"<PARAM NAME='StartingColNumber' VALUE='1'>"+
"<PARAM NAME='StartingRowNumber' VALUE='1'>"+
"<PARAM NAME='UnitType' VALUE='1'>"+
"<PARAM NAME='UserResize' VALUE='3'>"+
"<PARAM NAME='VirtualMaxRows' VALUE='-1'>"+
"<PARAM NAME='VirtualMode' VALUE='0'>"+
"<PARAM NAME='VirtualOverlap' VALUE='0'>"+

```

```

" <PARAM NAME=' Virtual Rows' VALUE=' 0' >" +
" <PARAM NAME=' Virtual Scroll Buffer' VALUE=' 0' >" +
" <PARAM NAME=' VisibleCols' VALUE=' 0' >" +
" <PARAM NAME=' VisibleRows' VALUE=' 0' >" +
" <PARAM NAME=' VScrollSpecial' VALUE=' 0' >" +
" <PARAM NAME=' VScrollSpecialType' VALUE=' 0' >" +
" <PARAM NAME=' Appearance' VALUE=' 0' >" +
" <PARAM NAME=' TextTip' VALUE=' 0' >" +
" <PARAM NAME=' TextTipDelay' VALUE=' 500' >" +
" <PARAM NAME=' ScrollBarTrack' VALUE=' 0' >" +
" <PARAM NAME=' ClipboardOptions' VALUE=' 15' >" +
" <PARAM NAME=' CellNoteIndicator' VALUE=' 0' >" +
" <PARAM NAME=' ShowScrollTips' VALUE=' 0' >" +
" <PARAM NAME=' DataMember' VALUE=' ' >" +
" <PARAM NAME=' OLEDropMode' VALUE=' 0' >" +
" </OBJECT>\n";
    sb.append("</td>\n");
    sb.append("<script>function romu"+id+"() { C.romuFPOINTSPREAD(C_"+id+"); }</script>\n");
    sb.append("<script for=' FP"+id+"'>");
event=' ScriptLeaveCell (Col , Row, NewCol , NewRow, Cancel) ' >C. leaveCell FPOINTSPREAD(C_"+id+", Col , Row, NewC
ol , NewRow, Cancel) ; </script>\n");
}

public void generateHTMLForEndTag(String tagName, StringBuffer sb)
{
}

public void generateJavaScriptForInit(int id, String tagName,
StringBuffer sb)
{
    sb.append("C_"+id+" = document.getElementById(' FP"+id+"');\n");
    sb.append("C_"+id+".CASA_spreadprop = '"+m_spreadprop+"';\n");
    sb.append("C.regi sterLi stener(romu"+id+");\n");
}
}

```

In the code you see...:

- The OBJECT-tag is generated, pointing to the unique classid that comes with the ActiveX control.
- Quite a lot of properties are set for the control. These properties are part of the ActiveX control definition, you can find explanation in the documentation that comes with the FarPoint control.
- The controls gets a certain id ("FP"+id), which is referenced by JavaScript processing later on.
- A script function "romu"+id+"()" is added and registered to be called every time a response from the server is processed.
- A script function is added that listens to the "ScriptLeaveEvent" of the spread control.

The Casabac control itself is defined by three attributes that are read from the attribute list passed in the method "generateHTMLForStartTag()":

- width
- height
- name of a central property that at runtime provides for the data to be displayed

The HTML code generated by the code above does not contain the Javascript to process data to be filled into the grid - this code is "outsourced" into a Javascript library "../casabacdemos/pages/FPOINTSPREADControls.js". The code of this library is:

```

var FONTNAME_FPOINTSSPREAD = "Verdana";
var FONTSIZE_FPOINTSSPREAD = 10;

function romuFPOINTSPREAD(cc)
{
    cc.ScriptEnhanced = true;
    var vtitlesCSV = getPropertyVal ue(cc.CASA_spreadprop + ". titles");
    var vpropsCSV = getPropertyVal ue(cc.CASA_spreadprop + ". val uepropsprop");
    var vwidsCSV = getPropertyVal ue(cc.CASA_spreadprop + ". wi dths");
    var vhal ignsCSV = getPropertyVal ue(cc.CASA_spreadprop + ". hal igns");

    // pass hal igns
    var vhal igns = decodeCSVSt ring(vhal ignsCSV);

    // pass titles into spread

```

```

var vtitles = decodeCSVString(vtitlesCSV);
for (var i=0; i<vtitles.length; i++)
{
    if (cc.CASA_firstCall != false) cc.FontName = FONTNAME_FPOINTSPREAD;
    if (cc.CASA_firstCall != false) cc.FontSize = FONTSIZE_FPOINTSPREAD;
    cc.Col = i;
    cc.Row = 0;
    cc.TypeHAlign = vhaligns[i];
    cc.Text = vtitles[i];
}

// pass widths
var vwids = decodeCSVString(vwidsCSV);
for (var i=0; i<vwids.length; i++)
    cc.ColWidth(i) = vwids[i];

// value props
var vprops = decodeCSVString(vpropsCSV);
cc.CASA_vprops = vprops;

// pass data into spread
var icount = 0;
while (true)
{
    var vallNull = true;
    for (var i=0; i<vprops.length; i++)
    {
        var value =
getPropertyValue(cc.CASA_spreadprop+.items["+i count+"]."+vprops[i]);
        if (value == null)
            continue;
        vallNull = false;
        cc.Col = i;
        cc.Row = icount+1;
        if (cc.CASA_firstCall != false) cc.TypeHAlign = vhaligns[i];
        cc.Text = value;
    }
    if (vallNull == true) break;
    icount++;
}
cc.MaxCols = vwids.length-1;
cc.MaxRows = icount;

// process commands
var icount = 0;
while (true)
{
    var vbc = getPropertyValue(cc.CASA_spreadprop+.commands["+i count+"].bc");
    var vx = getPropertyValue(cc.CASA_spreadprop+.commands["+i count+"].x");
    var vy = getPropertyValue(cc.CASA_spreadprop+.commands["+i count+"].y");
    if (vx == null)
        break;
    icount++;
    cc.Col = vx;
    cc.Row = vy;
    cc.BackgroundColor = vbc;
}

cc.CASA_firstCall = false;
}

function leaveCellFPOINTSPREAD(cc, col, row, newCol, newRow, cancel)
{
    if (col == -1 || row == -1 || newCol == -1 || newRow == -1) return;
    cc.Col = col;
    cc.Row = row;
    var vtext = cc.Text;
    var vprop = cc.CASA_vprops[col];
    var vrowindex = row-1;
    var vnowvalue = getPropertyValue(cc.CASA_spreadprop+.items["+vrowindex+"]."+vprop);
    if (vnowvalue == vtext) return; // no change
    setPropertyValue(cc.CASA_spreadprop+.items["+vrowindex+"]."+vprop, vtext);
    submitModel("submit");
}

```

The Javascript function "romuFPOINTSPREAD()" the processing is contained that is executed every time a response is processed inside the Casabac client framework. The function accesses the spread's data by using the "getPropertyValue()" method. You see:

- The configuration of the grid (columns, widths, ...) is kept in properties themselves keeping values that are stored as comma separated string. The program reads the property (e.g. "titles") and transfers the comma separated string into an array of strings.
- The data lines of the grid are accessed using array- and point-notation. The name of the access path to the properties is dynamically built, dependent on the grid's configuration: `getPropertyValue(cc.CASA_spreadprop+".items["+icount+"]."+vprops[i]);`

On server side the control is represented by an instance of class "FPOINTSPREADHandler". The class provides exactly the structure that is required by the Javascript processing for the control on client side:

```
package com.casabac.demolibrary;

import java.util.ArrayList;
import java.util.HashMap;

public class FPOINTSPREADInfo
{
    String m_haligns;
    public String getHaligns() { return m_haligns; }

    String m_titles;
    public String getTitles() { return m_titles; }

    String m_widths;
    public String getWidths() { return m_widths; }

    String m_valuepropsprop;
    public String getValuepropsprop() { return m_valuepropsprop; }

    ArrayList m_items = new ArrayList();
    public Object[] getItems() { return m_items.toArray(); }

    HashMap m_cellCommands = new HashMap();

    /**
     * Adds a column.
     *
     * @param title
     * Text that is displayed as columns header.
     *
     * @param valueprop
     * Property from which the value for the columns is taken. Each
     * row in the grid is represented by one object. This object must
     * provide a certain property per column. This is the "valueprop"
     * that you pass with this parameter.
     *
     * @param width
     * Width in FPOINT-Spread measurement. Seems to be something like
     * character-width but I am not sure...
     *
     * @param halign
     * Horizontal alignment of cells of column.
     */
    public void addColumn(String title,
                        String valueprop,
                        int width,
                        int halign)
    {
        if (m_valuepropsprop == null)
        {
            m_valuepropsprop = valueprop;
            m_titles = title;
            m_widths = "" + width;
            m_haligns = "" + halign;
        }
        else
        {
            m_valuepropsprop += ";" + valueprop;
            m_titles += ";" + title;
            m_widths += ";" + width;
            m_haligns += ";" + halign;
        }
    }

    /**
     * Adds an individual command that is operated on the grid.
     */
}
```

```

public void addCellCommand(FPOINTSREADCellCommand command)
{
    String key = command.getX()+"_"+command.getY();
    m_cellCommands.put(key, command);
}

/**
 * Casabac internal use.
 */
public Object[] getCommands()
{
    Object[] result = m_cellCommands.values().toArray();
    return result;
}

/**
 * Adds a row item object to the grid.
 *
 * @param row
 * Item object representing a row.
 */
public void addRow(Object row)
{
    m_items.add(row);
}

/**
 * Clear grid and grid definition.
 */
public void clear()
{
    clearItems();
    m_titles = "";
    m_valuepropsprop = "";
    m_haligns = "";
}

/**
 * Clears items of a grid.
 */
public void clearItems()
{
    m_items.clear();
    m_cellCommands.clear();
}
}

```

For using the control you have to register the control in:

- (1) `controllibraries.xml` ⇨ the package of the control library must be associated with a certain prefix
- (2) `editor_<extension>.xml` ⇨ the control is available in the Casabac Layout Painter.

This is the `controllibraries.xml` definition:

```

<controllibraries>
  <library package="com.casabac.demolibrary"
    prefix="demo">
  </library>
</controllibraries>

```

This is the editor extension file:

```

<controllibrary>
  <editor>
    <tag name="demo:fpointspread">
      <attribute name="spreadprop" datatype="property"/>
      <attribute name="height" datatype="height"/>
      <attribute name="width" datatype="width"/>
    </tag>
    <tagsubnodeextension control="itr" newsubnode="demo:fpointspread"/>
  </editor>
</controllibrary>

```

## Example

The control and its server side representation are used in the following example screen:

```
<page model="com.casabac.test35.FpointSpreadAdapter">
  <titlebar name="ActiveX Integration - Farpoint Spread 7.0">
  </titlebar>
  <header withdstance="false">
    <button name="Save">
    </button>
  </header>
  <pagebody takefullheight="true">
    <rowarea name="Spread" height="100%">
      <itr width="100%" height="100%">
        <demo:fpointspread spreadprop="spread" height="100%" width="100%">
        </demo:fpointspread>
      </itr>
    </rowarea>
  </pagebody>
  <statusbar withdstance="false">
  </statusbar>
</page>
```

The corresponding adapter code is:

```
package com.casabac.test35;

// This class is a generated one.

import java.util.*;

import com.casabac.demolibrary.FPOINTSPREADCellCommand;
import com.casabac.demolibrary.FPOINTSPREADInfo;
import com.casabac.server.*;
import com.casabac.server.util.*;
import com.casabac.util.*;

public class FpointSpreadAdapter
  extends Model
{
  public class RowItem
  {
    String m_firstName;
    String m_lastName;
    String m_street;
    String m_town;
    String m_country;
    public String getCountry()
    {
      return m_country;
    }
    public void setCountry(String country)
    {
      m_country = country;
    }
    public String getFirstName()
    {
      return m_firstName;
    }
    public void setFirstName(String firstName)
    {
      m_firstName = firstName;
    }
    public String getLastName()
    {
      return m_lastName;
    }
    public void setLastName(String lastName)
    {
      m_lastName = lastName;
    }
    public String getStreet()
    {
      return m_street;
    }
  }
}
```

```

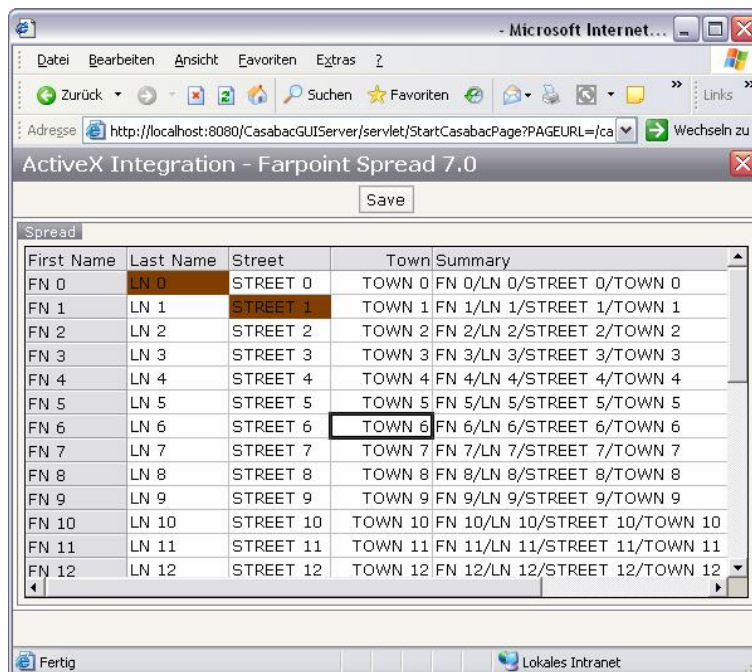
public void setStreet(String street)
{
    m_street = street;
}
public String getTown()
{
    return m_town;
}
public void setTown(String town)
{
    m_town = town;
}
public String toString()
{
    return m_firstName + "/" + m_lastName + "/" + m_street + "/" + m_town;
}
public String getSummary()
{
    return this.toString();
}
}

FPOINTSPREADInfo m_spread = new FPOINTSPREADInfo();
public FPOINTSPREADInfo getSpread() { return m_spread; }

public void init()
{
    m_spread.addColumn("First Name", "firstName", 10, 0);
    m_spread.addColumn("Last Name", "lastName", 10, 0);
    m_spread.addColumn("Street", "street", 10, 0);
    m_spread.addColumn("Town", "town", 10, 1);
    m_spread.addColumn("Summary", "summary", 30, 0);
    for (int i=0; i<30; i++)
    {
        RowItem row = new RowItem();
        row.setFirstName("FN " + i);
        row.setLastName("LN " + i);
        row.setStreet("STREET " + i);
        row.setTown("TOWN " + i);
        m_spread.addRow(row);
    }
    m_spread.addCellCommand(new FPOINTSPREADCellCommand(2, 2, 16000));
    m_spread.addCellCommand(new FPOINTSPREADCellCommand(1, 1, 16000));
}
}

```

The result looks as already shown at the beginning of this chapter:



## Conclusion

It's quite easy to integrate ActiveX controls into your pages. The server adapter processing is not touched at all, the ActiveX intergration is a client-only issue. In other words: if you have to do it: here you are!

The example we chose in this documentation is not a "mini-example" but already contains some complexity. A grid of data is transferred forth and back. The configuration of the grid is passed in parallel. Maybe your scenarios are much simpler.

...and: please be aware of what you do. It does not make sense to integrate optical ActiveX controls (e.g. "special buttons") that are easily reproducible using HTML and scripting. It only makes sense to use this kind of controls if you have to break out off limitations that are given by the browser design.

Also consider: the same way we embedded an ActiveX control in this documentation you can add applet controls to the page. The wording is different (you do not embed "OBJECT" tags but "APPLET" tags) but the structure is the same.