

This article is archived because it is not considered relevant for third-party developers creating commercial solutions today. The article is believed to be still valid for the original topic scope.



About

This is a UI Test widget which illustrates most of the features of the WidSets Scripting Language (WSL) UI components and how they can be used.

Note: These files and resource images are available in *SDK/examples/uitest*.

WidSets Scripting Language code: uitest.he

```
class
{
  /* The following example demonstrates the usage of different UI components
  * introduced in WidSets API.
  *
  * It also works as a coding style guide on how to produce good
  * looking and functioning WSL code.
  *
  * By using layouts and styles, you can easily re-skin your widgets
  * without code-changes.
  *
  * Some designlines:
  * + Each example is isolated within one createXXX function
  *   that follow Viewable prototype
  *
  * + Some examples have inner helper functions, for example
  *   createTable. These functions can access local variables
  *   declared in enclosing function, which make them very
  *   usable in disabling redundancy.
  *
  * + Inner functions are also used to implement example-specific
  *   action, softkey, menu, key, timer, or paint handlers - they
  *   are named myXXXHandler. The rest use default handlers defined
  *   in widget namespace.
  *
  */
}
```


UITest

```
    }
}

return null;*/

List encTypes = new List()
    .add("encoding=jpeg&width=32&height=24")
    .add("encoding=jpeg")
    .add("encoding=jpg&width=32&height=24")
    .add("encoding=jpg")
    .add("encoding=png");

Flow flow = new Flow(getStyle("default"));
Camera cam = new Camera(getStyle("default")); // Camera cam is global component
cam.setFlags(VISIBLE|LINEFEED);
cam.setPreferredSize(-100,-100);
flow.add(cam);
Shell shell = new Shell(flow);
shell.setSoftKeyHandler(handleSoft);
shell.setActionHandler(handleAction);

MenuItem SHOOT = new MenuItem(CMD_SHOOT, "Shoot"); //menuItem for taking the picture

void handleAction(Shell shell, Component source, int action)
{
    if (action == CMD_SHOOT) {
        setBubble(getImage(cam.capture(encTypes)), "this is a camera");

// picture is taken from camera and sent to createPreview function for previewing

        popShell(shell);
        cam.close();
        cam = null;

    } else if (action == CMD_BACK) {
        popShell(shell);
        cam.close();
        cam = null;
    }
}

MenuItem handleSoft(final Shell shell,
                    final Component focused,
                    final int key)
{
    if (key == SOFTKEY_MIDDLE) {
        return SHOOT;

    } else if (key == SOFTKEY_BACK) {
        return BACK;

    } else {
        return null;
    }
}

return shell;
}

//-----
```

UITest

```
//TODO, simpler example
Shell createCanvas()
{
    Canvas clock = new Canvas(getStyle("clock"), painter);
    int x, int y = getScreenSize();
    clock.setPreferredSize(x-4, y-2);
    Calendar time = new Calendar();
    Timer timer = null;

    void resetTimer(int interval)
    {
        if (timer != null) {
            timer.cancel();
            timer = null;
        }
        if (interval > 0) {
            time.setMillis(currentTimeMillis());
            timer = schedule(1000, interval, timerCall);
        }
    }

    void painter(Component c, Graphics g, Style style, int width, int height)
    {
        int h = time[ HOUR ];
        int m = time[ MINUTE ];
        int s = time[ SECOND ];

        int cx = width/2;
        int cy = height/2;

        {
            int angle = 6*s - 90;
            int dx = cx+(60*cos(angle))/MATH_SCALE;
            int dy = cy+(60*sin(angle))/MATH_SCALE;
            g.setColor(0x606060);
            drawHand(g, cx, cy, angle, 6, 60);
        }
        g.drawImage(style.image(0), cx, cy, HCENTER|VCENTER);
    }

    void drawHand(Graphics g, int cx, int cy, int angle, int base, int height)
    {
        int dx = cx+(height*cos(angle))/MATH_SCALE;
        int dy = cy+(height*sin(angle))/MATH_SCALE;

        int ax = cx+(base*cos(angle-90))/MATH_SCALE;
        int ay = cy+(base*sin(angle-90))/MATH_SCALE;

        int bx = cx+(base*cos(angle+90))/MATH_SCALE;
        int by = cy+(base*sin(angle+90))/MATH_SCALE;

        g.fillTriangle(ax, ay, bx, by, dx, dy);
    }

    void timerCall(Timer timer)
    {
        time.setMillis(currentTimeMillis());
        clock.repaint(false);
        flushScreen(false);
    }
}
```

UITest

```
        resetTimer(1000);

    return getViewShell(clock);
}

//-----

Shell createForm()
{
    Style style = getStyle("form.choice.item");
    Flow content = new Flow(getStyle("default"));
    content.setPreferredSize(-100, -100);

    content.add(createHeaderText("Form example"));

    //create some text components and choices
    addText("I tend to");
    addChoice(["-Select-", "Hack", "Thunk", "Fix", "Solve problems", "Just hang around"]);
    addText("at work, when my buddies are playing");
    addChoice(["-Select-", "Golf", "WOW", "Football", "Paintball", "Fool"]);
    addText("which makes me");
    Choice selector = addChoice(["-Select-", "A dull", "Happy", "Sad", "Rich"]);

    //create text-input that can contain any characters
    //and is not visible by default
    Input input = new Input(getStyle("form.input"), "Edit title", "boy", ANY);
    input.setPreferredWidth(-100);
    input.setFlags(LINEFEED);
    content.add(input);

    final Shell shell = new Shell(content);
    shell.setStyle(getStyle("maxi"));
    shell.setActionHandler(myActionHandler);

    void myActionHandler(final Shell shell, final Component source, final int action)
    {
        if (action == CMD_BACK) {
            popShell(shell);

        } else if (action == ITEM_CHANGED) {
            if (Choice(source) == selector) {
                if (selector.getSelected() == 0) {
                    input.setFlags(LINEFEED);
                } else {
                    input.setFlags(VISIBLE|LINEFEED|FOCUSABLE);
                }
                //component was hidden/revealed, redraw and
                //calculate dimension again
                input.repaint(true);
                flushScreen(false);
            }
        }
    }

    Text addText(String st)
    {
        Text text = new Text(getStyle("form.label"), st);
        text.setPreferredWidth(-100);
    }
}
```

UITest

```
text.setFlags(VISIBLE|LINEFEED);
content.add(text);
return text;
}

Choice addChoice(Value items)
{
    List elements = new List();
    foreach (Value item : items) {
        Label label = new Label(style, String(item));
        label.setFlags(VISIBLE|FOCUSABLE|LINEFEED);
        elements.add(label);
    }

    Choice choice = new Choice(getStyle("form.choice.display"), getStyle("form.choice.list"));
    choice.setChoices(elements, 0);
    choice.setFlags(VISIBLE|FOCUSABLE|LINEFEED);
    choice.setPreferredWidth(-100);
    content.add(choice);
    return choice;
}

return shell;
}

//-----

Shell createInput()
{
    final String STORE = "input.text";
    final int CMD_CHANGE = MAX_CMD_ID+1;

    Flow content = new Flow(getStyle("default"));
    content.setFlags(VISIBLE|LINEFEED);
    content.setPreferredSize(-100, -100);

    content.add(createHeaderText("Contents of following Input-field are stored into the
phone's flash memory and will be here when you re-open this view"));

    Store store = getStore();
    String text = store.has(STORE)? String(store.getValue(STORE)) : "Example content";

    Input input = new Input(getStyle("input"), "Your text", text, ANY);
    input.setFlags(VISIBLE|FOCUSABLE|LINEFEED);
    input.setPreferredWidth(-100);
    content.add(input);

    final Shell shell = new Shell(content);
    shell.setStyle(getStyle("maxi"));
    shell.setSoftKeyHandler(mySoftKeyHandler);
    shell.setActionHandler(myActionHandler);

    //Softkey handler is called whenever shell is being painted again, framework
    //asks for softkey button labels (MenuItem instances) for phone's OK and BACK keys
    MenuItem mySoftKeyHandler(final Shell shell, final Component focused, final int key)
    {
        if (key == SOFTKEY_BACK) {
            return BACK;
        }
    }
}
```

UITest

```
} else if (key == SOFTKEY_MIDDLE) {
    return new MenuItem(CMD_CHANGE, "Change");
}
return null;
}

//Action handler is called for widget/shell whenever event is fired, event such as
//menu-item-select takes place, softkeys are clicked, item is focused, item contents are changed
//etc
void myActionHandler(final Shell shell, final Component source, final int action)
{
    if (action == CMD_BACK) {
        //remove current shell (view) from display stack (= return to last view)
        popShell(shell);

    } else if (action == CMD_CHANGE) {
        //soft key was clicked, start editing mode
        //Note that Input-component captures phone's joystick-fire event and starts
        //editing mode itself
        input.edit();

    } else if (action == ITEM_CHANGED && source == Component(input)) {
        //editing done, save new text to store
        getStore().put(STORE, Value(input.getText()));
    }
}

return shell;
}

//-----

Shell createList()
{
    final int CMD_SHOW = MAX_CMD_ID+1;

    //from imdb.com
    //stars-rating was rounded as CLCD 1.0 (and WidSets Scripting Language)
    //does not support floating point numbers

    Value data = [
        ["name" => "Flight of Fury",
         "year" => 2007,
         "stars" => 3,
         "image" => "list_flight_of_fury.png",
         "tagline" => "A flight plan to freedom...",
         "plot" => "Shot at Castel Studios in Bucharest: John (Seagal), is sent in to
                  recover a stolen Stealth Bomber. His trusty sidekick Rojar (Alki David) and John
                  ever faithful Jessica (Ciera Payton), fight the rebel forces of Banansistan, led
                  the vivacious Ellianna (Katie Jones)."],
        ["name" => "Attack Force",
         "year" => 2006,
         "stars" => 3,
         "image" => "list_attack_force.png",
         "tagline" => "It's humanity's greatest hope... and our last chance.",
         "plot" => "Marshall Lawson loses his strike-team in a cold-blooded and
                  seemingly random attack. After this he takes it upon himself to investigate the
                  suspicious circumstances of the brutal killings. Soon he uncovers CTX Majestic,
                  covert military operation so secret, that now the military wants Marshall elimi
```

UITest

```
Resolute in his pursuit, Marshall engages in a merciless battle with a drug
dealer operation that appears to be secretly funded by a rogue arm of the milit
],
["name" => "Shadow Man",
 "year" => 2006,
 "stars" => 4,
 "image" => "list_shadow_man.png",
 "tagline" => "Either you're with him... or you're dead.",
 "plot" => "An intelligence operative discovers that no one is what they seem in the shadow
],
["name" => "Mercenary for Justice",
 "year" => 2006,
 "stars" => 4,
 "image" => "list_mercenary_for_justice.png",
 "tagline" => "This time, he's not for hire.",
 "plot" => "Seagal plays a CIA covert operative who is forced into helping a team of mercen
a wealthy client when his best friend's wife and son are kidnapped by the same
the CIA are in pursuit in an attempt to silence Seagal."
],
["name" => "Black Dawn",
 "year" => 2005,
 "stars" => 4,
 "image" => "list_black_dawn.png",
 "tagline" => "It's always darkest before dawn.",
 "plot" => "Jonathan Cold returns, this time he goes Undercover to stop a group of Terroris
]
];

Flow content = new Flow(getStyle("default"));
content.setPreferredSize(-100, -100);

content.add(createHeaderText("Movie list"));

foreach (Value movie : data)
{
    Flow itemFlow = new Flow(getStyle("list.bg"));
    itemFlow.setFlags(VISIBLE|FOCUSABLE|LINEFEED);
    itemFlow.setPreferredWidth(-100);
    itemFlow.setData(movie);
    itemFlow.setAction(CMD_SHOW);

    Picture pic = new Picture(getStyle("list.img"), getImage(String(movie.image)));
    pic.setFlags(VISIBLE);
    pic.setPreferredWidth(-25);
    itemFlow.add(pic);

    {
        Flow textFlow = new Flow(getStyle("default"));
        textFlow.setPreferredWidth(-100);
        textFlow.setFlags(VISIBLE);

        Label name = new Label(getStyle("list.name"), movie.name);
        name.setPreferredWidth(-75);
        name.setFlags(VISIBLE|LINEFEED);
        textFlow.add(name);

        Text tag = new Text(getStyle("list.tag"), movie.tagline);
        tag.setPreferredWidth(-75);
        tag.setFlags(VISIBLE|LINEFEED);
        textFlow.add(tag);

        itemFlow.add(textFlow);
    }
}
}
```

UITest

```
    }

    content.add(itemFlow);
}

final Shell shell = getViewShell(content);
shell.setActionHandler(myActionHandler);

Shell movieView(Value movie)
{
    Flow view = new Flow(getStyle("default"));
    view.setPreferredSize(-100, -100);

    view.add(createHeaderText(movie.name));

    Flow starsFlow = new Flow(getStyle("list.star"));
    starsFlow.setFlags(VISIBLE|LINEFEED);
    starsFlow.setPreferredWidth(-100);
    Image starImage = getImage("list_star.png");
    for (int i=0; i<int(movie.stars); i++) {
        Picture star = new Picture(getStyle("list.star"), starImage);
        starsFlow.add(star);
    }
    view.add(starsFlow);

    Text plot = new Text(getStyle("list.plot"), movie.plot);
    plot.setFlags(VISIBLE);
    plot.setPreferredWidth(-100);
    view.add(plot);

    return getViewShell(view);
}

void myActionHandler(final Shell shell, final Component source, final int action)
{
    if (action == CMD_BACK) {
        //same logic works for both views, re-use ;)
        popShell(shell);

    } else if (action == CMD_SHOW) {
        pushShell(movieView(Value(source.getData())));
    }
}

return shell;
}

//-----

Shell createMenu()
{
    final int CMD_FILE      = MAX_CMD_ID+1;
    final int CMD_FILE_NEW  = MAX_CMD_ID+11;
    final int CMD_FILE_OPEN = MAX_CMD_ID+12;
    final int CMD_FILE_EXIT = MAX_CMD_ID+13;

    final int CMD_EDIT      = MAX_CMD_ID+2;
    final int CMD_VIEW      = MAX_CMD_ID+3;
    final int CMD_WINDOW    = MAX_CMD_ID+4;
    final int CMD_HELP      = MAX_CMD_ID+5;
```

UITest

```
//Note that OPEN_SETTINGS is declared in WidSets Scripting Language API
//Events with OPEN_SETTINGS id are captured and consumed
//by the framework, firing it will open Widget's
//setting view
Menu menu = new Menu().begin(CMD_FILE, "File")
    .add(CMD_FILE_NEW, "New")
    .add(CMD_FILE_OPEN, "Open")
    .add(CMD_FILE_EXIT, "Exit")
    .end()
    .add(CMD_EDIT, "Edit")
    .add(CMD_VIEW, "View")
    .add(OPEN_SETTINGS, "Settings")
    .add(CMD_WINDOW, "Window")
    .add(CMD_HELP, "Help");

Flow content = new Flow(getStyle("default"));
content.setPreferredSize(-100, -100);

//item #0
content.add(createHeaderText("Menus can contain submenus, items/submenus can be disabled/enabled"));
addLabel("I have menu"); //item #1
addLabel("I do not have have a menu"); //item #2
addLabel("I have only Settings-menu"); //item #3

final Shell shell = new Shell(content);
shell.setStyle(getStyle("maxi"));
shell.setSoftKeyHandler(mySoftKeyHandler);
shell.setActionHandler(myActionHandler);
shell.setMenuHandler(myMenuHandler);

void addLabel(String name)
{
    Label label = new Label(getStyle("mainLabel"), name);
    label.setPreferredWidth(-100);
    label.setFlags(VISIBLE|FOCUSABLE|LINEFEED);
    content.add(label);
}

MenuItem mySoftKeyHandler(final Shell shell, final Component focused, final int key)
{
    if (key == SOFTKEY_BACK) {
        return BACK;
    }
    else if (key == SOFTKEY_OK) {
        int index = focused.getParent().indexOf(focused);

        if (index == 2) {
            //this element did not have menu, so do not display open menu-softkey
            return null;
        }
        else {
            return new MenuItem(OPEN_MENU, "Menu");
        }
    }
    return null;
}

void myActionHandler(final Shell shell, final Component source, final int action)
{

```

UITest

```
if (action == CMD_BACK) {
    popShell(shell);

} else if (action == FOCUS_CHANGED) {
    //make system refresh soft keys each time focus is changed
    shell.updateMenu();

} else if (action > MAX_CMD_ID) {
    setBubble(null, "This feature is not implemented.");
}
}

Menu myMenuHandler(Shell shell, Component source)
{
    menu.reset();

    int index = source.getParent().indexOf(source);
    if (index == 2) {
        return null;

    } else if (index == 3) {
        //note that disabling a MenuItem disables also it's
        //child-menus
        menu.enable(CMD_FILE, false);
        menu.enable(CMD_EDIT, false);
        menu.enable(CMD_VIEW, false);
        menu.enable(CMD_WINDOW, false);
        menu.enable(CMD_HELP, false);
    }
    return menu;
}

return shell;
}

//-----

Shell createPicture()
{
    Menu m_menu = new Menu().add(MAX_CMD_ID+1, "Picture 1")
        .add(MAX_CMD_ID+2, "Picture 2")
        .add(MAX_CMD_ID+3, "Picture 3");

    Flow content = new Flow(getStyle("picture.bg"));
    content.setPreferredSize(-100, -100);

    Picture pic = new Picture(getStyle("picture.frame"), getImage("picture_1.png"));
    pic.repaint(true);
    content.add(pic);

    final Shell shell = new Shell(content);
    shell.setStyle(getStyle("maxi"));
    shell.setSoftKeyHandler(mySoftKeyHandler);
    shell.setActionHandler(myActionHandler);
    shell.setMenuHandler(myMenuHandler);

    MenuItem mySoftKeyHandler(final Shell shell, final Component focused, final int key)
    {
        if (key == SOFTKEY_BACK) {
```

UITest

```
    return BACK;

    } else if (key == SOFTKEY_OK) {
        return new MenuItem(OPEN_MENU, "Change");
    }
    return null;
}

void myActionHandler(final Shell shell, final Component source, final int action)
{
    if (action == CMD_BACK) {
        popShell(shell);

    } else if (action >= MAX_CMD_ID+1 && action <= MAX_CMD_ID+3) {
        Image img = getImage("picture_" + (action-MAX_CMD_ID) + ".png");
        pic.setImage(img);
        pic.repaint(true);
        flushScreen(false);
        int x, int y = pic.getSize();
    }
}

Menu myMenuHandler(Shell shell, Component source)
{
    return m_menu.reset();
}

return shell;
}

//-----

Shell createProgress()
{
    int current = 0;
    int maximum = 10;

    //create progress Prompt with 10 steps
    Prompt prompt = new Prompt(null, "This is progress", null, CANCEL);
    prompt.setProgress(current, maximum);
    prompt.setActionHandler(myActionHandler);
    prompt.push();

    //schedule (start) Timer that notifies time-handler once a second
    //(1000ms) to update the current progress
    Timer timer = schedule(0, 1000, myTimerHandler);

    void myTimerHandler(Timer timer)
    {
        prompt.setProgress(current++ % maximum, maximum);
        //repaint screen, no need to recalculate component dimensions
        flushScreen(false);
    }

    void myActionHandler(Shell shell, Component source, int action)
    {
        if (action == CMD_CANCEL) {
            //always remember to cancel your timers when your widget/shell
            //exists or they'll run there forever!
            if (timer != null) {
```

UITest

```
        timer.cancel();
        timer = null;
    }
    popShell(shell);
}

return null;
}
```

```
//-----
```

```
Shell createPrompt()
```

```
{
    final int CMD_LEFT = MAX_CMD_ID+1;
    final int CMD_RIGHT = MAX_CMD_ID+2;

    //softkey labels
    MenuItem left = new MenuItem(CMD_LEFT, "Maybe");
    MenuItem right = new MenuItem(CMD_RIGHT, "Why not");

    Prompt prompt = new Prompt(null, "To be, or not to be?", left, right);
    prompt.setProgress(0, 0);
    prompt.setActionHandler(myActionHandler);
    prompt.push();

    void myActionHandler(final Shell shell, final Component source, final int action)
    {
        if (action == CMD_LEFT || action == CMD_RIGHT) {
            popShell(shell);
        }
    }

    return null;
}
```

```
//-----
```

```
Shell createSliding()
```

```
{
    Flow content = new Flow(getStyle("default"));
    content.setPreferredSize(-100, -100);

    content.add(createHeaderText("Click number keys to see different transitions between views..."));

    final Shell shell = new Shell(content);
    shell.setStyle(getStyle("maxi"));
    shell.setSoftKeyHandler(defaultSoftKeyHandler);
    shell.setActionHandler(defaultActionHandler);
    shell.setKeyHandler(myKeyHandler);

    boolean myKeyHandler(Component source, int op, int code)
    {
        int w, int h = getScreenSize();
        Flow flow = new Flow(getStyle("sliding.bg"));
        Picture pic = new Picture(getStyle("sliding.text"), getImage("bubble_genie.png"));
        pic.setPreferredSize(-100, -100);
        flow.add(pic);
    }
}
```

UITest

```
Shell view = getViewShell(flow);

if (op != KEY_RELEASED) {
    if (code >= '1' && code <= '9') {
        //push new view on the screen
        pushShell(view);

        //and then slide it out using selected transformation
        switch (code) {
            case '1':
                slideOut(w, h, w, h, view);
                break;

            case '2':
                slideOut(0, h, w, h, view);
                break;

            case '3':
                slideOut(w*-1, h, w, h, view);
                break;

            case '4':
                slideOut(w, 0, w, h, view);
                break;

            case '5':
                //just push it, no sliding
                break;

            case '6':
                slideOut(w*-1, 0, w, h, view);
                break;

            case '7':
                slideOut(w, h*-1, w, h, view);
                break;

            case '8':
                slideOut(0, h*-1, w, h, view);
                break;

            case '9':
                slideOut(w*-1, h*-1, w, h, view);
                break;
        }
        //we consume this key-event
        return true;
    }
    return false;
}

return shell;
}

//-----

Shell createStyles()
{
    Style style = getStyle("form.choice.item");
}
```

UITest

```
Flow content = new Flow(getStyle("default"));
content.setPreferredSize(-100, -100);

content.add(createHeaderText("List of Label components with different styles"));

addLabel("styles.1", "Style label 1");
addLabel("styles.2", "Style label 2");
addLabel("styles.3", "Style label 3");
addLabel("styles.4", "Style label 4");
addLabel("styles.5", "Style label 5");
addLabel("styles.6", "Style label 6");
addLabel("styles.7", "Style label 7");
addLabel("styles.8", "Style label 8");

Flow addLabel(String style, String text)
{
    Label label = new Label(getStyle(style), text);
    label.setPreferredWidth(-100);
    //label.setFlags(VISIBLE|FOCUSABLE);

    Flow flow = new Flow(getStyle("styles.container"));
    flow.setPreferredWidth(-100);
    flow.setFlags(VISIBLE|LINEFEED|FOCUSABLE);
    flow.add(label);
    content.add(flow);
    return flow;
}

return getViewShell(content);
}

//-----

Shell createTable()
{
    //get length of letter "m" to be used as measure for cell-widths
    final int FONT_LENGTH = getStyle("table.cell").font(0).stringWidth("m");
    final int CELL_WIDTH = FONT_LENGTH*6;
    final int MARGIN_WIDTH = FONT_LENGTH*3;
    final int MAX_COLS = int('Z')-int('A');
    final int COLS = 10;
    final int ROWS = 10;

    List cells = new List();

    Flow content = new Flow(getStyle("default"));

    content.add(createHeaderText("Spreadsheet Table"));

    //create headers
    addHeader("", MARGIN_WIDTH, false);
    for (int x=0; x<COLS; x++) {
        String name = char(int('A') + (x % MAX_COLS)); //A, B, C, ...
        addHeader(name, CELL_WIDTH, x+1 == COLS);
    }

    //Cells can be easily accessed from a Map, but using
    //cell name as key doesn not work because 2-char long
    //alphanumeric hashes do collide
}
```

UITest

```
//create cells
for (int y=0; y<ROWS; y++) {
    addHeader(String(y+1), MARGIN_WIDTH, false);

    for (int x=0; x<COLS; x++) {
        Input c = addCell("", CELL_WIDTH, x+1 == COLS);
        cells.add(c);
    }
}

//generate some content
getCell("A3").setText("Foo");
getCell("B3").setText("Bar");
getCell("J10").setText("=");

void addHeader(String text, int width, boolean linefeed)
{
    Label l = new Label(getStyle("table.header"), text);
    l.setPreferredWidth(width);
    //when adding components to flow, if there is no LINEFEED or
    //WRAP flag set, they will be all on same line
    l.setFlags(linefeed? VISIBLE|LINEFEED : VISIBLE);
    content.add(l);
}

Input addCell(String text, int width, boolean linefeed)
{
    Input l = new Input(getStyle("table.cell"), text, ANY);
    l.setPreferredWidth(width);
    l.setFlags(linefeed? VISIBLE|LINEFEED|FOCUSABLE : VISIBLE|FOCUSABLE);
    content.add(l);
    return l;
}

Input getCell(String name)
{
    int col = int(name.charAt(0))-int('A');
    int row = int(name.substring(1, name.length()))-1;
    int offset = row*COLS+col;

    return (offset <= cells.size())? cells[offset] : null;
}

return getViewShell(content);
}

//-----

Shell createTicker()
{
    Flow content = new Flow(getStyle("default"));
    content.setPreferredSize(-100, -100);

    content.add(createHeaderText("Scrollin' Tickers"));

    add(new Ticker(getStyle("ticker.loop.blurred"), "Looping Ticker text (while not focused)"));
    add(new Ticker(getStyle("ticker.loop.focused"), "Looping Ticker text (while focused)"));
    add(new Label(getStyle("ticker.normal"), "Non-scrolling Label text that hopefully is too long");
    add(new Ticker(getStyle("ticker.bounce.blurred"), "Bouncing Ticker text (while not focused)
        [There is no one who loves pain itself, who seeks after it and
```

UITest

```
        wants to have it, simply because it is pain...]"));
add(new Ticker(getStyle("ticker.bounce.focused"), "Bouncing Ticker text (while focused)
[There is no one who loves pain itself, who seeks after it and want
to have it, simply because it is pain...]"));

void add(Component c)
{
    c.setPreferredWidth(-100);
    c.setFlags(VISIBLE|FOCUSABLE|LINEFEED);
    content.add(c);
}

return getViewShell(content);
}

//-----
//-----
//-----

//Shared functions

Shell getViewShell(Component content)
{
    final Shell shell = new Shell(content);
    shell.setStyle(getStyle("maxi"));
    shell.setSoftKeyHandler(defaultSoftKeyHandler);
    shell.setActionHandler(defaultActionHandler);

    return shell;
}

MenuItem defaultSoftKeyHandler(final Shell shell, final Component focused, final int key)
{
    if (key == SOFTKEY_BACK) {
        return BACK;
    }
    return null;
}

void defaultActionHandler(final Shell shell, final Component source, final int action)
{
    if (action == CMD_BACK) {
        popShell(shell);
    }
}

Text createHeaderText(String name)
{
    Text text = new Text(getStyle("header"), name);
    text.setPreferredWidth(-100);
    text.setFlags(VISIBLE|LINEFEED);
    return text;
}

Label createMainLabel(String name, Viewable viewable)
{
    Label label = new Label(getStyle("mainLabel"), name);
```

UITest

```
label.setPreferredWidth(-100);
label.setFlags(VISIBLE|FOCUSABLE|LINEFEED);
label.setData(viewable);
label.setAction(CMD_SELECT);
return label;
}

// Widget stuff

Component createElement(String viewName,
                        String elementName,
                        Style style,
                        Object context)
{
    if (elementName.equals("maxi")) {
        return new Scrollable(style, createMain());
    }
    return null;
}

Flow createMain()
{
    Flow content = new Flow(getStyle("default"));
    content.setPreferredSize(-100, -100);

    String header = "UI Examples";

    content.add(createHeaderText(header));

    //function references to example implementing function
    //that implements Viewable -type, passed as parameter
    content.add(createMainLabel("Bubble", createBubble));
    content.add(createMainLabel("Camera", createCamera));
    content.add(createMainLabel("Canvas", createCanvas));
    content.add(createMainLabel("Form", createForm));
    content.add(createMainLabel("Input", createInput));
    content.add(createMainLabel("List", createList));
    content.add(createMainLabel("Menu", createMenu));
    content.add(createMainLabel("Picture", createPicture));
    content.add(createMainLabel("Progress", createProgress));
    content.add(createMainLabel("Prompt", createPrompt));
    content.add(createMainLabel("Sliding", createSliding));
    content.add(createMainLabel("Styles", createStyles));
    content.add(createMainLabel("Table", createTable));
    content.add(createMainLabel("Ticker", createTicker));
    return content;
}

void startWidget ()
{
    setMinimizedView(createMinimizedView("viewMini", getStyle("white")));
}

Shell openWidget ()
{
    Flow view = createMaximizedView("viewMaxi", getStyle("maxi"));
    Shell shell = new Shell(view);
}
```

UITest

```
    return shell;
}

//Soft key callback, which would normally handle all soft key-calls,
//in this case it handles only Shells which do not have
//soft key callback set, and that is the main list created above

MenuItem getSoftKey(Shell shell, Component focused, int key)
{
    if (key == SOFTKEY_OK) {
        return null;

    } else if (key == SOFTKEY_BACK) {
        return BACK;
    }
    return null;
}

//Widget's generic event-callback, normally all
//events occurred on Shells on this widget would
//come here, but they are captured on each example
//using inner callback-functions

void actionPerformed(Shell shell, Component source, int action)
{
    switch(action)
    {
        case CMD_SELECT:
        {
            //Label from main menu was clicked, take
            //function-reference and invoke it.
            Object data = source.getData();
            if (data != null) {
                final Shell s = Viewable(data)->();

                //If it returned a shell, push that shell
                //to view stack.
                if (s != null) {
                    pushShell(s);
                }
            }
        }
        break;

        case CMD_BACK:
        {
            popShell(shell);
        }
        break;
    }
}
}
```

Widget.xml

```
<?xml version="1.0" encoding="utf-8"?>

<widget spec_version="2.0">
  <info>
    <name>uitest</name>
    <version>0.1</version>
    <author>render</author>
    <clientversion>0.98</clientversion>
    <shortdescription>Example widget demonstrating use of WSL UI components</shortdescription>
    <longdescription>Example widget demonstrating use of WSL UI components</longdescription>
    <tags>test example WSL script ui component view</tags>
  </info>

  <parameters>
    <parameter type="string" name="widgetname" description="Name of widget" editable="no">UI Test</parameter>
  </parameters>

  <resources>
    
    

    

    

    
    
    
    

    
    
    
    

    
    
    
    
    
    

    
    
    

    <code src="uitest.he"/>

  <stylesheet>
    white {
      background: solid white;
    }

    minbg {
      background: grid9 "bkg.png" 6 6 6 6;
    }

    titleLabel {
      padding: 0 0 10 0;
    }
  </stylesheet>
</widget>
```

UITest

```
font-1: small bold;
color-1: #000000;
align: hcenter vcenter;
}

maxi {
  color-1: black;
  background: solid white;
  border: 1 1 1 1;
  border-type: rectangle black;
}

mainLabel {
  background: grid9 "selector.png" 5 5 8 5;
  padding: 1 5 1 5;
  border: 1 0 0 0;
  font-1: small;
  color-1: #000000;

  focused
  {
    padding: 1 4 1 4;
    border-type: rectangle #b54000;
    border: 1 0 0 0;
    background: vgradient #f3742e #f35600;
    color-1: #ffffff;
  }
}

header {
  background: vgradient #112f55 #30517a;
  font-1: small bold;
  align: hcenter vcenter;
  color-1: #ffffff;
  padding: 3sp 6sp 3sp 6sp;
}

<!-- Canvas example -->
clock {
  background: solid white;
  color-1: black;
  image-1: "canvas_nuppi.png";
}

<!-- Form example -->
form.base {
  margin: 0 10 5 10;
  background: solid #e3e3e3;
  border: 1 1 1 1;
  border-type: rectangle #a4a4a4;
  padding: 2 4 2 2;
  font: proportional small plain;
  color: #333333;

  focused {
    border-type: rectangle red;
  }
}

form.label : form.base {
  font: proportional medium plain;
  background: none;
}
```

UITest

```
border-type: none;
}

form.input : form.base {
  font: proportional medium plain;
}

form.choice.display : form.base {
  label-icon-orientation: right;
  image: "form_choice_arrow.png";

  focused {
    border-type: rectangle red;
  }
}

form.choice.list {
  width: 100%;
  align: left bottom;
  background: solid #385179;
  border: 3 3 3 3;
  border-type: image "form_choice_bg.png";
  padding: 3 3 3 3;
}

form.choice.item {
  padding: 2 10 2 10;
  font: proportional medium bold;
  color: white;
  image: "menu2.png";
  label-icon-orientation: right;

  focused {
    background: grid9 "form_choice_bg_selected.png" 3 3 3 3;
  }
}

<!-- Input example -->
input {
  border: 1 1 1 1;
  border-type: rectangle black;
  margin: 2 2 2 2;
  padding: 2 2 2 2;
  color: black;
  font: small;
}

<!-- List example -->
list.bg {
  background: grid9 "selector.png" 5 5 8 5;
  padding: 1 5 1 5;
  border: 1 0 0 0;
  font-1: small;
  color-1: #000000;
  /*vspacing: 5;
  hspacing: 5;*/

  focused
  {
    padding: 1 4 1 4;
    border-type: rectangle #b54000;
```

UITest

```
        border: 1 0 0 0;
        background: vgradient #f3742e #f35600;
        color-1: #ffffff;
    }
}

list.img {
    padding: 3sp 6sp 3sp 0;
}

list.star {
    padding-top: 4sp;
    align: vcenter hcenter;
}

list.name {
    color: black;
    font: medium bold;
}

list.tag {
    color: black;
    font: small;
}

list.plot {
    color: black;
    font: small;
    padding: 6sp 6sp 6sp 6sp;
}

<!-- Picture example -->
picture.bg {
    align: hcenter vcenter;
}

picture.frame {
    border: 20 21 21 21;
    border-type: image "picture_frame.png";
    align: hcenter vcenter;
    margin: 10 10 10 10;
}

<!-- Sliding example -->
sliding.bg {
    border: 4 4 4 4;
    border-type: rectangle black;
}

sliding.text {
    color-1: black;
    font-1: large bold;
    align: hcenter vcenter;
}

<!-- Styles -->
styles.base {
    margin: 3 3 3 3;
    padding: 2 5 2 5;
}

styles.container {
```

UITest

```
border: 1 1 1 1;
border-type: none;

focused {
    border: 1 1 1 1;
    border-type: rectangle red;
}
}

styles.1 : styles.base {
    background: vgradient white #cccccc;

    focused {
        background: vgradient #cccccc white;
    }
}

styles.2 : styles.base {
    border-type: rectangle red #cccccc blue brown;
    border: 4 4 4 4;

    focused {
        background: solid yellow;
    }
}

styles.3 : styles.base {
    background: grid9 "styles_selector.png" 2 2 2 2;

    focused {
        font: medium bold;
    }
}

styles.4 : styles.base {
    font: small underlined;
    align: vcenter right;

    focused {
        align: vcenter left;
    }
}

styles.5 : styles.base {
    align: vcenter hcenter;
    border-bottom: 10;
    padding-top: 10;
    border-type: rectangle black;

    focused {
        padding-top: 0;
        border: 10 10 10 10;
    }
}

styles.6 : styles.base {
    background: image "styles_wood.png" transparent top left repeat-x repeat-y;
    color: white;

    focused {
        color: red;
    }
}
```

UITest

```
}

styles.7 : styles.base {
  background: hgradient blue black;
  font: large bold;
  color: #cccccc;

  focused {
    background: hgradient black blue;
  }
}

styles.8 : styles.base {
  padding-left: 12;
  background: image "styles_bullet.png" transparent vcenter left;

  focused {
    padding-left: 0;
    padding-right: 12;
    align: vcenter right;
    background: image "styles_bullet.png" transparent vcenter right;
  }
}

<!-- Table example -->
table.header {
  background: solid #D4D0C8;
  border: 0 1 1 0;
  border-type: rectangle #808080;
  padding: 2 2 2 2;
  font: small;
  color: black;
  align: vcenter hcenter;
}

table.cell {
  background: solid white;
  border: 0 1 1 0;
  border-type: rectangle #C0C0C0;
  padding: 2 2 2 2;
  font: small;
  color: black;
  align: left vcenter;

  focused {
    border-type: rectangle black;
    border: 2 2 2 2;
    padding: 0 0 0 0;
    cursor-color: white;
  }
}

<!-- Ticker example -->
ticker.normal {
  background: grid9 "selector.png" 5 5 8 5;
  padding: 1 5 1 5;
  border: 1 0 0 0;
  font-1: small;
  color-1: #000000;
```

UITest

```
focused
{
  padding: 1 4 1 4;
  border-type: rectangle #b54000;
  border: 1 0 0 0;
  background: vgradient #f3742e #f35600;
  color-1: #ffffff;
}
}

ticker.loop.blurred {
  background: grid9 "selector.png" 5 5 8 5;
  padding: 1 5 1 5;
  border: 1 0 0 0;
  font-1: small;
  color-1: #000000;
  ticker-speed: 200;
  ticker-mode: loop;

  focused
  {
    padding: 1 4 1 4;
    border-type: rectangle #b54000;
    border: 1 0 0 0;
    background: vgradient #f3742e #f35600;
    color-1: #ffffff;
  }
}

ticker.loop.focused {
  background: grid9 "selector.png" 5 5 8 5;
  padding: 1 5 1 5;
  border: 1 0 0 0;
  font-1: small;
  color-1: #000000;

  focused
  {
    padding: 1 4 1 4;
    border-type: rectangle #b54000;
    border: 1 0 0 0;
    background: vgradient #f3742e #f35600;
    color-1: #ffffff;
    ticker-speed: 200;
    ticker-mode: loop;
  }
}

ticker.bounce.blurred {
  background: grid9 "selector.png" 5 5 8 5;
  padding: 1 5 1 5;
  border: 1 0 0 0;
  font-1: large bold;
  color-1: #000000;
  ticker-speed: 100;
  ticker-mode: bounce;

  focused
  {
    padding: 1 4 1 4;
    border-type: rectangle #b54000;
    border: 1 0 0 0;
```

UITest

```
        background: vgradient #f3742e #f35600;
        color-1: #ffffff;
    }
}

ticker.bounce.focused {
    background: grid9 "selector.png" 5 5 8 5;
    padding: 1 5 1 5;
    border: 1 0 0 0;
    font-1: large bold;
    color-1: #000000;

    focused
    {
        padding: 1 4 1 4;
        border-type: rectangle #b54000;
        border: 1 0 0 0;
        background: vgradient #f3742e #f35600;
        color-1: #ffffff;
        ticker-speed: 100;
        ticker-mode: bounce;
    }
}
</stylesheet>
</resources>

<layout minimizedheight="65sp">
    <view id="viewMini" class="minbg">
        <label class="titleLabel" top="0%" right="100%-8px" bottom="100%" left="8px">${widgetname}</label>
    </view>

    <view id="viewMaxi" class="maxi">
        <script id="maxi" class="maxi"/>
    </view>

    <webview>
        <weblabel class="top:0px;left:10px;">${widgetname}</weblabel>
    </webview>
</layout>

</widget>
```