CrossUI Cookbook

Getting Started Guide

All rights reserved.

[Preface 7](#_Toc453071658)

[Chapter 1. Preparation 8](#_Toc453071659)

[1.1. Download the package 8](#_Toc453071660)

[1.2. The package folder 8](#_Toc453071661)

[1.3. Glance at examples and API 9](#_Toc453071662)

[Chapter 2. Hello World 10](#_Toc453071663)

[2.1. The first application 10](#_Toc453071664)

[2.2. Render onto a html node 11](#_Toc453071665)

[2.3. Do it in CrossUI RAD Tools 13](#_Toc453071666)

[2.4. Application loading process 19](#_Toc453071667)

[2.5. Code Editor 21](#_Toc453071668)

[2.5.1. Highlight code from Outline window 22](#_Toc453071669)

[2.5.2. Code Folding 22](#_Toc453071670)

[2.5.3. Code Intellisense 23](#_Toc453071671)

[2.5.3.1. When context doesn’t recognize the input string 23](#_Toc453071672)

[2.5.3.2. Type dot after a recognizable variable 25](#_Toc453071673)

[2.5.3.3. When use dbclick 25](#_Toc453071674)

[2.5.4. Find the object definition code 26](#_Toc453071675)

[2.5.5. Generate event code automatically 26](#_Toc453071676)

[Chapter 3. Controls Facebook 27](#_Toc453071677)

[3.1. Script testing environment 27](#_Toc453071678)

[3.2. “Hello world” in env.html 29](#_Toc453071679)

[3.3. Control creation and runtime update 30](#_Toc453071680)

[3.4. Button related 31](#_Toc453071681)

[3.4.1. onClick event 31](#_Toc453071682)

[3.4.2. Boolean Controls 32](#_Toc453071683)

[3.4.3. Link Control 32](#_Toc453071684)

[3.5. Label related 33](#_Toc453071685)

[3.6. Input related 34](#_Toc453071686)

[3.6.1. setValue/setValue/getUIValue/setUIValue 34](#_Toc453071687)

[3.6.2. Dirty Mark 34](#_Toc453071688)

[3.6.3. Password Input 35](#_Toc453071689)

[3.6.4. Multi-lines 35](#_Toc453071690)

[3.6.5. Input validation 36](#_Toc453071691)

[3.6.5.1. valueFormat property 36](#_Toc453071692)

[3.6.5.2. beforeFormatCheck event 36](#_Toc453071693)

[3.6.6. Dynamic input validation 37](#_Toc453071694)

[3.6.7. Error Mark 37](#_Toc453071695)

[3.6.7.1. Default Error Mark 37](#_Toc453071696)

[3.6.7.2. Validation Tips 37](#_Toc453071697)

[3.6.7.3. Binding Validation 38](#_Toc453071698)

[3.6.7.4. Custom Error Mark 38](#_Toc453071699)

[3.6.8. Mask Input 39](#_Toc453071700)

[3.6.9. xui.UI.ComboInput 40](#_Toc453071701)

[3.6.9.1. Pop list for selection 40](#_Toc453071702)

[3.6.9.2. combobox, listbox and helpinput 40](#_Toc453071703)

[3.6.9.3. Date Piker 41](#_Toc453071704)

[3.6.9.4. Time Picker 42](#_Toc453071705)

[3.6.9.5. Color Picker 42](#_Toc453071706)

[3.6.9.6. File Picker 43](#_Toc453071707)

[3.6.9.7. Getter 43](#_Toc453071708)

[3.6.9.8. Custom Pop Window 44](#_Toc453071709)

[3.6.9.9. Command Buttons 45](#_Toc453071710)

[3.6.10. RichEditor 45](#_Toc453071711)

[3.7. List related 46](#_Toc453071712)

[3.7.1. A Simple one 46](#_Toc453071713)

[3.7.2. A little bit complicated 47](#_Toc453071714)

[3.7.3. RadioBox 48](#_Toc453071715)

[3.7.4. IconList and Gallery 48](#_Toc453071716)

[3.7.5. Item selection 49](#_Toc453071717)

[3.7.6. Container related 49](#_Toc453071718)

[3.7.7. Pane and Panel 50](#_Toc453071719)

[3.7.8. Block 51](#_Toc453071720)

[3.8. Dialog related 51](#_Toc453071721)

[3.8.1. Normal state 51](#_Toc453071722)

[3.8.2. Min and Max status 53](#_Toc453071723)

[3.8.3. Modal Mode 53](#_Toc453071724)

[3.9. Layout Control 54](#_Toc453071725)

[3.10. Multi-pages Controls 55](#_Toc453071726)

[3.10.1. noPanel property 56](#_Toc453071727)

[3.10.2. ButtonViews types 57](#_Toc453071728)

[3.10.3. Page selection 57](#_Toc453071729)

[3.10.4. Pages 58](#_Toc453071730)

[3.10.4.1. Close and options Button 58](#_Toc453071731)

[3.10.4.2. Add/Remove Pages 60](#_Toc453071732)

[3.10.5. Dynamic content loading 61](#_Toc453071733)

[3.10.5.1. onIniPanelView 61](#_Toc453071734)

[3.10.5.2. beforeUIValueSet/afterUIValueSet 61](#_Toc453071735)

[3.11. Menus and toolbars 62](#_Toc453071736)

[3.11.1. Pop Menu 62](#_Toc453071737)

[3.11.2. MenuBar 63](#_Toc453071738)

[3.11.3. Toolbars 64](#_Toc453071739)

[3.12. TreeBar and TreeView 65](#_Toc453071740)

[3.12.1. Three selection mode 65](#_Toc453071741)

[3.12.1.1. No-selection 65](#_Toc453071742)

[3.12.1.2. Single-selection 66](#_Toc453071743)

[3.12.1.3. Multi-selection 66](#_Toc453071744)

[3.12.2. Group Item 67](#_Toc453071745)

[3.12.3. Expand all nodes by default 67](#_Toc453071746)

[3.12.4. Mutex Expand 68](#_Toc453071747)

[3.12.5. Dynamic Destruction 69](#_Toc453071748)

[3.12.6. Dynamically loading 69](#_Toc453071749)

[3.13. TreeGrid 70](#_Toc453071750)

[3.13.1. Header and Rows 70](#_Toc453071751)

[3.13.1.1. Sets standard format 71](#_Toc453071752)

[3.13.1.2. Sets simplified format 72](#_Toc453071753)

[3.13.2. getHeader 72](#_Toc453071754)

[3.13.3. getRows 73](#_Toc453071755)

[3.13.4. Active Modes 74](#_Toc453071756)

[3.13.4.1. non-active appearance 74](#_Toc453071757)

[3.13.4.2. row-active appearance 75](#_Toc453071758)

[3.13.4.3. cell-active appearance 75](#_Toc453071759)

[3.13.5. Selection Mode 76](#_Toc453071760)

[3.13.5.1. Non-selection 76](#_Toc453071761)

[3.13.5.2. Single row selection 77](#_Toc453071762)

[3.13.5.3. Multi-row selection 77](#_Toc453071763)

[3.13.5.4. Single cell selection 78](#_Toc453071764)

[3.13.5.5. Multi-cells selection 78](#_Toc453071765)

[3.13.6. The Tree Grid 79](#_Toc453071766)

[3.13.7. Column config 80](#_Toc453071767)

[3.13.7.1. The first column 80](#_Toc453071768)

[3.13.7.2. Column width 81](#_Toc453071769)

[3.13.7.3. Drag&Drop to modify column width 81](#_Toc453071770)

[3.13.7.4. Drag&Drop to modify column position 82](#_Toc453071771)

[3.13.7.5. Default Sorting 82](#_Toc453071772)

[3.13.7.6. Custom Sorting 83](#_Toc453071773)

[3.13.7.7. Hide columns 83](#_Toc453071774)

[3.13.7.8. Setting Cell Types in column header 84](#_Toc453071775)

[3.13.7.9. column header style 84](#_Toc453071776)

[3.13.7.10. column header icon 85](#_Toc453071777)

[3.13.7.11. Update column header dynamically 86](#_Toc453071778)

[3.13.8. Row config 86](#_Toc453071779)

[3.13.8.1. Row height 86](#_Toc453071780)

[3.13.8.2. Drag&Drop to modify row height 87](#_Toc453071781)

[3.13.8.3. Setting cell type in row 87](#_Toc453071782)

[3.13.8.4. Row style 88](#_Toc453071783)

[3.13.8.5. Row numbers 88](#_Toc453071784)

[3.13.8.6. Custom row numbers 89](#_Toc453071785)

[3.13.8.7. Alternate Row Colors 90](#_Toc453071786)

[3.13.8.8. Group 90](#_Toc453071787)

[3.13.8.9. Preview and Summary region 91](#_Toc453071788)

[3.13.8.10. Update row dynamically 92](#_Toc453071789)

[3.13.9. Cell config 93](#_Toc453071790)

[3.13.9.1. Cell types 93](#_Toc453071791)

[3.13.9.2. Cell style 94](#_Toc453071792)

[3.13.9.3. Update cell dynamically 95](#_Toc453071793)

[3.13.10. Editable 95](#_Toc453071794)

[3.13.10.1. Editable TreeGrid 96](#_Toc453071795)

[3.13.10.2. Editable column 96](#_Toc453071796)

[3.13.10.3. Editable row 97](#_Toc453071797)

[3.13.10.4. Editable cell 97](#_Toc453071798)

[3.13.10.5. The Editor 98](#_Toc453071799)

[3.13.10.6. Custom the editor 99](#_Toc453071800)

[3.13.11. Add/Remove rows 100](#_Toc453071801)

[3.14. Other standard controls 101](#_Toc453071802)

[3.14.1. ProgressBar 101](#_Toc453071803)

[3.14.2. Slider 102](#_Toc453071804)

[3.14.3. Image 103](#_Toc453071805)

[3.14.4. PageBar 103](#_Toc453071806)

[Chapter 4. Data exchanging(Ajax) 104](#_Toc453071807)

[4.1. Fiddler 105](#_Toc453071808)

[4.2. To get the contents of the file 105](#_Toc453071809)

[4.3. Synchronous data exchange 105](#_Toc453071810)

[4.4. Cross-domain 106](#_Toc453071811)

[4.4.1. To monitor SAjax 106](#_Toc453071812)

[4.4.2. To monitor IAjax 107](#_Toc453071813)

[4.5. File Upload 108](#_Toc453071814)

[4.5.1. Selecting upload file with ComboInput 108](#_Toc453071815)

[4.5.2. Upload by IAjax 109](#_Toc453071816)

[4.6. A request wrapper for real application 109](#_Toc453071817)

[4.7. XML Data 110](#_Toc453071818)

[4.8. An overall example 110](#_Toc453071819)

[Chapter 5. Distributed UI 112](#_Toc453071820)

[5.1. Shows dialog from a remote file 112](#_Toc453071821)

[5.2. xui.Module and xui.ModuleFactory 113](#_Toc453071822)

[5.2.1. xui.ModuleFactory config 113](#_Toc453071823)

[5.2.2. xui.Module.Load 114](#_Toc453071824)

[5.2.3. newCom and getCom 115](#_Toc453071825)

[5.2.4. xui.UI.Tag 116](#_Toc453071826)

[5.2.5. Destroy com 116](#_Toc453071827)

[5.2.6. If com exists in memory 117](#_Toc453071828)

[Chapter 6. Some fundamental things 117](#_Toc453071829)

[6.1. Pop-up window 117](#_Toc453071830)

[6.1.1. alert window 117](#_Toc453071831)

[6.1.2. confirm window 117](#_Toc453071832)

[6.1.3. prompt window 118](#_Toc453071833)

[6.1.4. pop window 118](#_Toc453071834)

[6.2. Asynchronous execution 118](#_Toc453071835)

[6.2.1. asyRun 118](#_Toc453071836)

[6.2.2. resetRun 119](#_Toc453071837)

[6.3. Skin switcher 119](#_Toc453071838)

[6.3.1. Switch skin for whole application 119](#_Toc453071839)

[6.3.2. Change skin for a single control 119](#_Toc453071840)

[6.4. Language switcher 120](#_Toc453071841)

[6.5. DOM Manipulation 121](#_Toc453071842)

[6.5.1. Node generation and insertion 121](#_Toc453071843)

[6.5.2. Attributes and CSS 121](#_Toc453071844)

[6.5.3. className 122](#_Toc453071845)

[6.5.4. Dom events 123](#_Toc453071846)

[6.5.5. Node Drag&Drop 124](#_Toc453071847)

[6.5.5.1. Drag&Drop profile 125](#_Toc453071848)

[6.5.5.2. Events in Drag&Drop 126](#_Toc453071849)

[6.6. xui.Template 127](#_Toc453071850)

[6.6.1. example 1 127](#_Toc453071851)

[6.6.2. example 2 128](#_Toc453071852)

[6.6.3. A SButton based on xui.Template 129](#_Toc453071853)

[6.7. About Debugging 130](#_Toc453071854)

[6.7.1. The code package for debugging 130](#_Toc453071855)

[6.7.2. Debugging Tools 130](#_Toc453071856)

[6.7.3. Monitor Tools 131](#_Toc453071857)

[Chapter 7. Some typical issues 132](#_Toc453071858)

[7.1. Layout 132](#_Toc453071859)

[7.1.1. Docking 132](#_Toc453071860)

[7.1.2. xui.UI.Layout 132](#_Toc453071861)

[7.1.3. Relative Layout 133](#_Toc453071862)

[7.2. UI Control’s Drag&Drop 135](#_Toc453071863)

[7.2.1. Drag&Drop control among containers 135](#_Toc453071864)

[7.2.2. List sorting 1 135](#_Toc453071865)

[7.2.3. List sorting 2 136](#_Toc453071866)

[7.3. Form 137](#_Toc453071867)

[7.3.1. Form 1 137](#_Toc453071868)

[7.3.2. DataBinder 138](#_Toc453071869)

[7.4. Custom UI Styles 140](#_Toc453071870)

[7.4.1. Custom one instance only - 1 140](#_Toc453071871)

[7.4.2. Custom one instance only - 2 140](#_Toc453071872)

[7.4.3. Custom one instance only - 3 141](#_Toc453071873)

[7.4.4. Custom one instance only - 4 141](#_Toc453071874)

[7.4.5. Custom one instance only - 5 141](#_Toc453071875)

[7.4.6. Custom one instance only - 6 142](#_Toc453071876)

[7.4.7. Custom style for an UI Class 142](#_Toc453071877)

[7.4.8. Custom style for all UI Class - skin 143](#_Toc453071878)

[7.4.8.1. First: Copy 143](#_Toc453071879)

[7.4.8.2. Second: Little by little, modify pictures and CSS 143](#_Toc453071880)

[The end 144](#_Toc453071881)

# 

# Preface

CrossUI is a Cross-Browser JavaScript framework with cutting-edge functionality for rich web application.

CrossUI RAD Tools enables developers to rapidly develop and package the exactly same code and UI into Web Apps, Native Desktop Apps for Windows, OS X, Linux and UNIX on 32-bit and 64-bit architectures as well as Mobile Apps for iPhone, iPad, Windows Phone, webOS, BlackBerry, and Android devices. With this powerful RAD Tool, developers can build cross-platform applications just like what they do in VB or Delphi.

**Develop Once, Deploy Anywhere!**

**Features & Resources:**

1. Rich client-side API, works with any backend or static HTML pages.
2. Web services (JSON/XML/SOAP) can be directly bound.
3. More than 40 common components, including Tabs, Dialog, TreeGrid, TimeLine and many other web GUI components.
4. Wide cross-browser compatibility, IE6+, firefox1.5+, opera9+, safari3+ and Google Chrome.
5. Full API Documentation with tons of samples.
6. Ever Increasing Code Snippets.
7. PHP/C #/JAVA Back-end service codes are available.
8. CrossUI is Open Source under LGPL3 license;
9. CrossUI RAD (commercial license) can reduce development time significantly.

**This guide focuses on CrossUI Framework itself, and contains some info about CrossUI RAD Tools. In this guide book, all the examples will be demoed in browsers. But those examples are cross-platform; you can package them with CrossUI RAD Tools, and deploy them anywhere.**

If you have any good suggestions, you can contact me at linb[at]crossui.com.

Go to <http://www.crossui.com/Forum> for the more information.

# Preparation

First of all, note that all instances of this tutorial are based on CrossUI version 1.0. Therefore, our first task is to download the 1.0 release package, and to establish the local environment.

## Download the package

CrossUI framework zip package can be downloaded from

<http://www.crossui.com/download.html> or

<https://github.com/linb/crossui> .

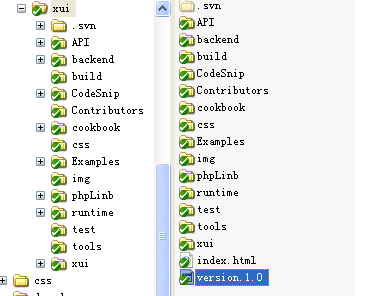
It’s the latest stable version, but not the latest code. I suggest you get the latest code from our SVN.

For those who are not familiar with SVN, should learn how to use SVN first. After all, a lot of open-source projects use SVN to manage code. SVN requires a client program to connect to what is called a "repository" where the files are stored. On commonly used SVN client is called TortoiseSVN, which is freely available. Other clients exist, but TortoiseSVN is recommended due to its simplicity of use.

**Version 1.4 repository URL:** <https://github.com/linb/CrossUI/tree/master/xui1.4> **.**

## The package folder

If you downloaded package from Google group, extract the package to a local folder. If you fetch the code from SVN, does not need to extract.

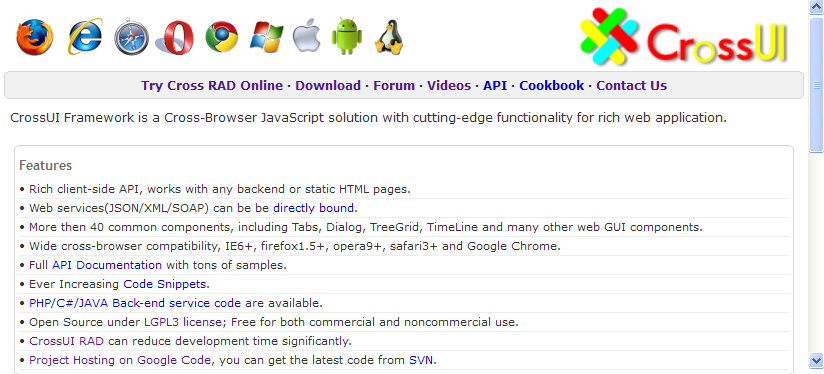


*The contents of the package folder*

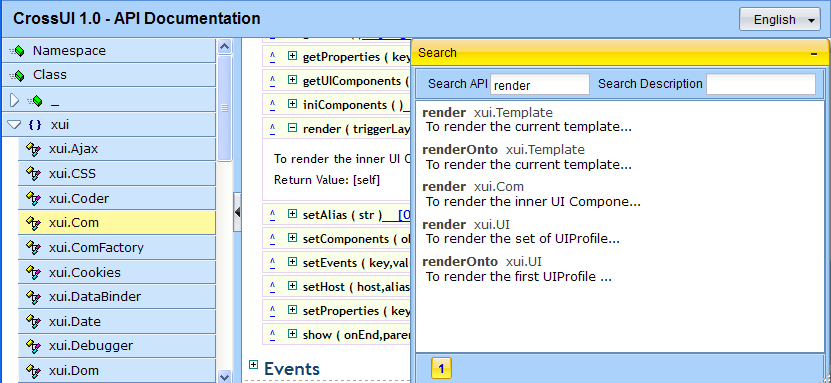
By default, most of the examples in the package can be run in local disk directly, but a small number of examples need php background environment, or MySQL database. In this case, you need to prepare Apache server (version 2 and above), php (version 5 and above) and MySQL (version 5 and above). And, copy the package to apache web directory.

## Glance at examples and API

If your Apache/php environment works well, after you copied the package folder to Apache’s web directory (this tutorial assumes that your root directory is http://localhost/CrossUI/), you should be able to open the page with your browser: http://localhost/CrossUI/.



You can browse http://localhost/CrossUI/Examples/ for examples, and http://localhost/CrossUI/API/ for API Documentation.



API description

API search window

**A simple glance at API is strongly recommended. Learn about how to search a specific API, and how to run the inner code snippet.**

# Hello World

## The first application

As many would expect or not expect, the first example is "Hello World".

Now, create a new folder “mycases” in the package folder (again, this tutorial assumes that your root directory is http://localhost/CrossUI/), add a sub folder “chapter1” in it, and create a file named “helloworld.html” in “chapter1”. Enter the following code:

<!DOCTYPE html>

<html>

<head>

<meta http-equiv="content-type" content="text/html; charset=utf-8" />

<meta http-equiv="Content-Style-Type" content="text/css" />

<meta http-equiv="imagetoolbar" content="no" />

<script type="text/javascript" src="../../runtime/xui/js/xui-all.js"></script>

<title>CrossUI Case</title>

</head>

<body>

<script type="text/javascript">

xui.main(function(){

xui.alert("Hi", "Hello World");

});

</script>

</body>

</html>

recommended

Include lib file

main function

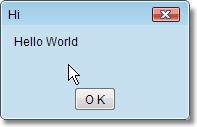
**chapter1/helloworld.html**

***You can find all the source code for each example in this tutorial in the zip package. The CrossUI Cookbook Zip Package with examples can be downloaded from***

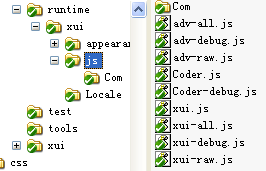
[***http://www.crossui.com/download.html***](http://www.crossui.com/download.html) ***.***

You can double-click the helloworld.htm1 to open the file.

Or open URL [http://localhost/CrossUI/chapter1/helloworld.html](http://localhost/jsLinb/cases/chapter1/helloworld.html) in your browser (Firefox or chrome is recommended here). And you can see the following result:



File “xui-all.js” contains all standard controls (Button, Input, CombInput, Tabs, TreeBar, and TreeGrid etc.). This file can be found in “runtime/js” folder.



## Render onto a html node

“Just replace DIVs with your controls.” A project manager said. "Our web page engineer is responsible to design an html file including a DIV with a unique ID, and JavaScript engineer is responsible to build an advanced UI control, and replace that DIV."

The following example in file chapter1/renderonto.html:

<!DOCTYPE html>

<html>

<head>

<meta http-equiv="content-type" content="text/html; charset=utf-8" />

<meta http-equiv="Content-Style-Type" content="text/css" />

<meta http-equiv="imagetoolbar" content="no" />

<script type="text/javascript" src="../../runtime/xui/js/xui-debug.js"></script>

<title>CrossUI Case</title>

</head>

<body>

<div id="grid" style="position:absolute;left:100px;top:100px;width:300px;height:200px;"></div>

<script type="text/javascript">

xui.main(function(){

var grid = new xui.UI.TreeGrid();

grid.setGridHandlerCaption('grid')

.setRowNumbered(true)

.setHeader(['col 1','col 2','col 3'])

.setRows([

['a1','a2','a3'],

['b1','b2','b3'],

['c1','c2','c3'],

['d1','d2','d3'],

['e1','e2','e3'],

['f1','f2','f3']

]);

grid.renderOnto('grid');

});

</script>

</body>

</html>

Render onto that DIV

Sets rows

Sets columns

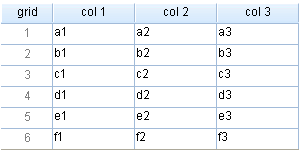
Show line number

Sets grid caption

The DIV with id “grid”

**chapter1/renderonto.html**

The result is:



**There are two ways to get the same result;** codes were in renderonto2.html and renderonto3.html.

renderonto2.html :

xui.main(function(){

(new xui.UI.TreeGrid({

gridHandlerCaption:'grid',

rowNumbered:true,

header:['col 1','col 2','col 3'],

rows:[['a1','a2','a3'],['b1','b2','b3'],['c1','c2','c3'],

['d1','d2','d3'],['e1','e2','e3'],['f1','f2','f3']]

})).renderOnto('grid');

});

Gives properties directly

**chapter1/renderonto2.html**

renderonto3.html :

xui.main(function(){

xui.create('TreeGrid',{

gridHandlerCaption:'grid',

rowNumbered:true,

header:['col 1','col 2','col 3'],

rows:[['a1','a2','a3'],['b1','b2','b3'],['c1','c2','c3'],

['d1','d2','d3'],['e1','e2','e3'],['f1','f2','f3']]

}).renderOnto('grid');

});

Using xui.create

**chapter1/renderonto3.html**

These three approaches generated the same result. You can use any of those in your project according to your habits. But the first approach (using new and setXX) is recommended.

## Do it in CrossUI RAD Tools

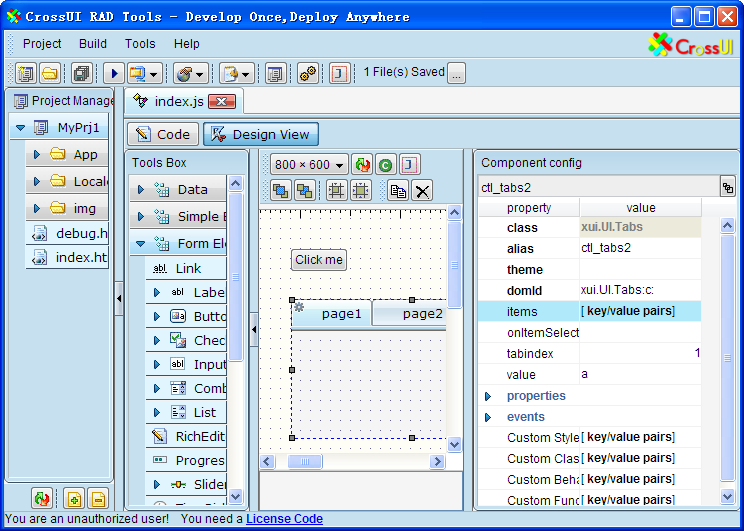
CrossUI RAD can reduce development time significantly, especially on UI layout.

There are two types of designer in CrossUI: online version and desktop version. Desktop version is integrated with many advanced features: document management, package, deployment, and so on.

In order to do the following exercises, you need to download CrossUI RAD Tools desktop version from <http://www.crossui.com/download.html>.

If you don’t want to download that, you can go to <http://www.crossui.com/RAD/Builder.html>, and do the following exercises online.

RAD Tools desktop:



Controls box

Canvas

Debug it

Switch views

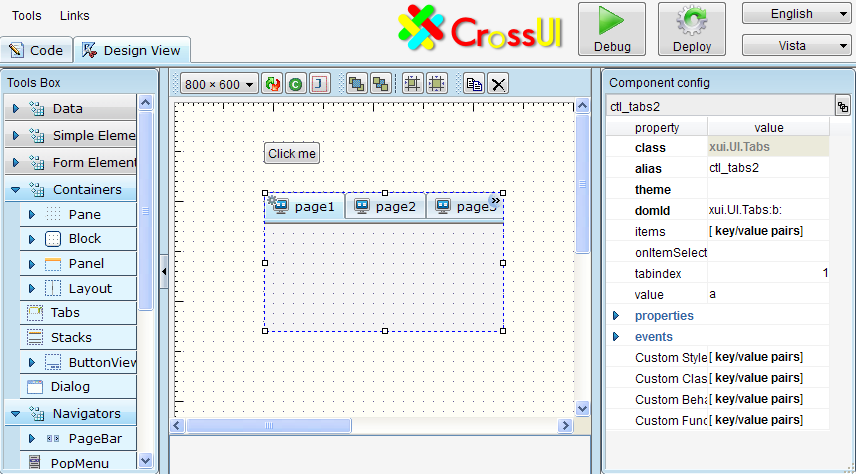
Properties and events

Dblclick for API

Controls tree

Alias

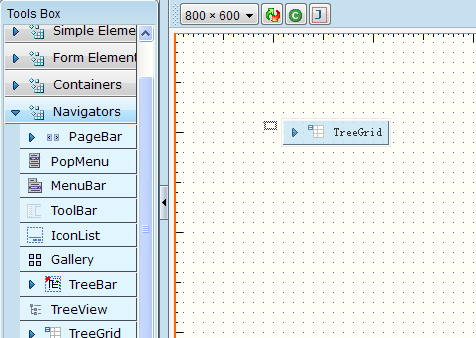
RAD Tools online:



Debug it in browser

Now, we are trying to create the previous section’s grid example in RAD Tools Designer.

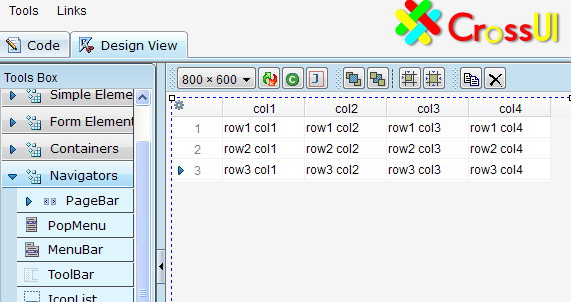
* 1. Open the navigators group in “Tools Box”, and drag the “TreeGrid” control to the Canvas area.



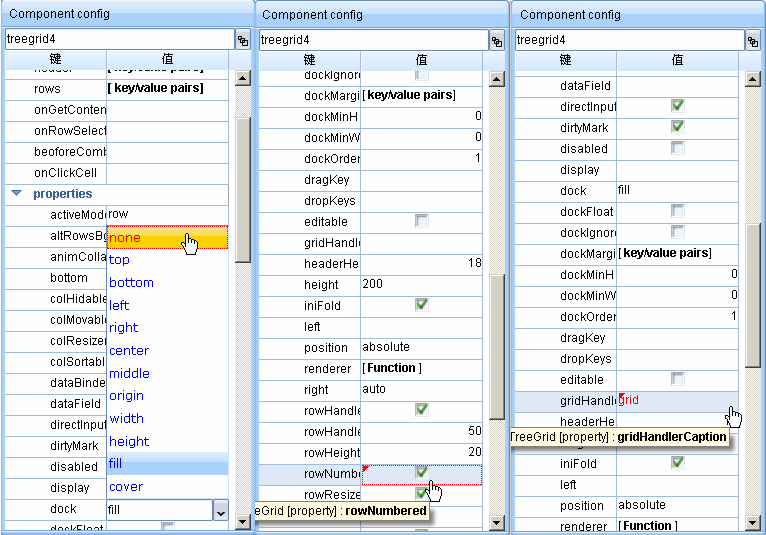
Expand the Navigators group

Drag a TreeGrid to canvas from here

* 1. Click to select the “treegrid” (It’s selected by default)



* 1. Sets this grid’s properties according to the following picture.
* Sets dock to ‘none’;
* Sets rowNumbered to false;
* Sets gridHandlerCapion to ‘grid’.

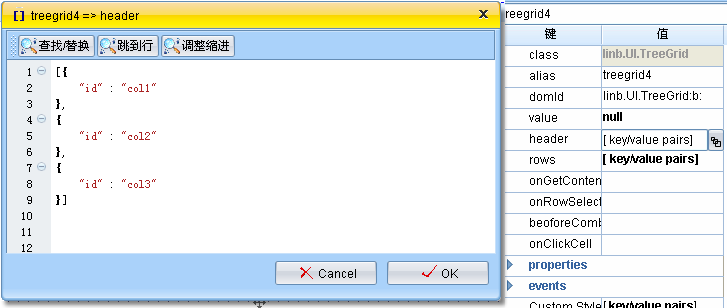


gridHandlerCaption

Click to setting rowNumbered

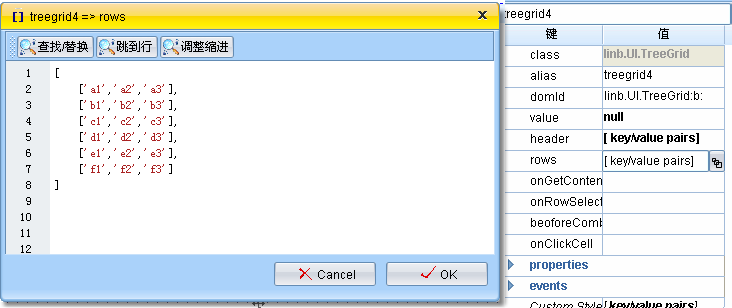
Click to setting dock prop

* 1. Sets header and rows



Click here to pop the header setting window

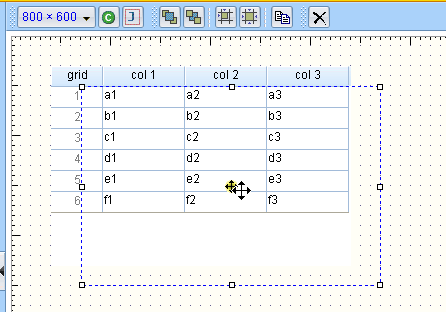
Sets header data



Click here to pop the row setting window

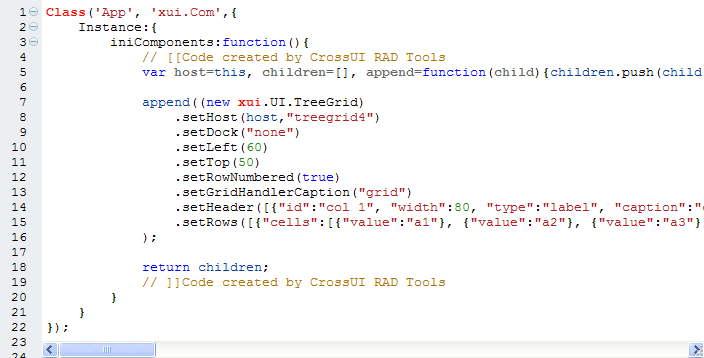
Sets rows data

* 1. Click to select the grid, adjust its position and size



* 1. Now, switch to “Code” view





Code created by CrossUI RAD Tools

Above code is serialized by CrossUI RAD. Header data and rows data will not look the same as your setting.

* 1. Click “Debug” Button to open the test window, you will see the same result with section 2.2.



Debug command

Debug command



* 1. Copy the code from this test page, and paste to a new file designer.grid.html.

<!DOCTYPE html>

<html>

<head>

<meta http-equiv="content-type" content="text/html; charset=utf-8" />

<meta http-equiv="Content-Style-Type" content="text/css" />

<meta http-equiv="imagetoolbar" content="no" />

<title>Web application powered by XUI framework</title>

</head>

<body>

<div id='loading'><img src="../../runtime/loading.gif" alt="Loading..." /></div>

../runtime/loading.gif" alt="Loading..." /></div>

<script type="text/javascript" src="../../runtime/xui/js/xui-all.js"></script>

<script type="text/javascript">

Class('App', 'xui.Module',{

Instance:{

iniComponents:function(){

// [[code created by CrossUI RAD Tools

var host=this, children=[], append=function(child){children.push(child.get(0))};

append((new xui.UI.TreeGrid)

.host(host,"treegrid4")

.setDock("none")

.setLeft(60)

.setTop(50)

.setRowNumbered(true)

.setGridHandlerCaption("grid")

.setHeader([{"id":"col 1", "width":80, "type":"label", "caption":"col 1"}, {"id":"col 2", "width":80, "type":"label", "caption":"col 2"}, {"id":"col 3", "width":80, "type":"label", "caption":"col 3"}])

.setRows([{"cells":[{"value":"a1"}, {"value":"a2"}, {"value":"a3"}], "id":"j"}, {"cells":[{"value":"b1"}, {"value":"b2"}, {"value":"b3"}], "id":"k"}, {"cells":[{"value":"c1"}, {"value":"c2"}, {"value":"c3"}], "id":"l"}, {"cells":[{"value":"d1"}, {"value":"d2"}, {"value":"d3"}], "id":"m"}, {"cells":[{"value":"e1"}, {"value":"e2"}, {"value":"e3"}], "id":"n"}, {"cells":[{"value":"f1"}, {"value":"f2"}, {"value":"f3"}], "id":"o"}])

);

return children;

// ]]code created by CrossUI RAD Tools

}

}

});

xui.Module.load('App', function(){

xui('loading').remove();

});

</script>

</body>

</html>

Include lib file in body

Class created by CrossUI RAD.

You can save this part of code to **App/js/index.js**

Load UI in asynchronous mode

If no App Class in memory, by default, CrossUI framewok will load the Class from **App/js/index.js** file .

Showing a loading picture

chapter1/designer.grid.html

## Application loading process

In section 2.3, we put all html and JavaScript code in a single file. For a bigger application, it’s not a wise solution. A real application may be include dozens of classes. For a developer, maintaining each class in a separate file is always a must.

OK. Let’s separate “designer.grid.html” into two files 🡪 designer.grid.standard.html, and **App/js/index.js**.

designer.grid.standard.html is:

<!DOCTYPE html>

<html>

<head>

<meta http-equiv="content-type" content="text/html; charset=utf-8" />

<meta http-equiv="Content-Style-Type" content="text/CSS" />

<meta http-equiv="imagetoolbar" content="no" />

<title>Web application powered by XUI framework</title>

</head>

<body>

<div id='loading'><img src="../runtime/loading.gif" alt="Loading..." /></div>

<script type="text/javascript" src="../../runtime/xui/js/xui-all.js"></script>

<script type="text/javascript">

xui.Module.load('App', function(){

xui('loading').remove();

});

</script>

</body>

</html>

At last, remove loading picture

Load App class from App/js/index.js asynchronously

**chapter1/designer.grid.standard.html**

App/js/index.js is:

Class('App', 'xui.Module',{

Instance:{

iniComponents:function(){

// [[code created by CrossUI RAD Tools

var host=this, children=[], append=function(child){children.push(child.get(0))};

append((new xui.UI.TreeGrid)

.host(host,"treegrid4")

.setDock("none")

.setLeft(60)

.setTop(50)

.setRowNumbered(true)

.setGridHandlerCaption("grid")

.setHeader([{"id":"col 1", "width":80, "type":"label", "caption":"col 1"}, {"id":"col 2", "width":80, "type":"label", "caption":"col 2"}, {"id":"col 3", "width":80, "type":"label", "caption":"col 3"}])

.setRows([{"cells":[{"value":"a1"}, {"value":"a2"}, {"value":"a3"}], "id":"j"}, {"cells":[{"value":"b1"}, {"value":"b2"}, {"value":"b3"}], "id":"k"}, {"cells":[{"value":"c1"}, {"value":"c2"}, {"value":"c3"}], "id":"l"}, {"cells":[{"value":"d1"}, {"value":"d2"}, {"value":"d3"}], "id":"m"}, {"cells":[{"value":"e1"}, {"value":"e2"}, {"value":"e3"}], "id":"n"}, {"cells":[{"value":"f1"}, {"value":"f2"}, {"value":"f3"}], "id":"o"}])

);

return children;

// ]]code created by CrossUI RAD Tools

}

}

});

**chapter1/App/js/index.js**

When we open **designer.grid.standard.html** in Browser, the loading process will be:

After DOM is ready, CrossUI lib will handle UI Class 🡪 App

Open html/URL in Browser

Browser loading html code (<1k)

**Browser load “loading.gif”(small)，and show it (Users will see the loading picture)**

Browser load xui-all.js simultaneously

[xui.Module.load] load code from App/js/index.js file asynchronously

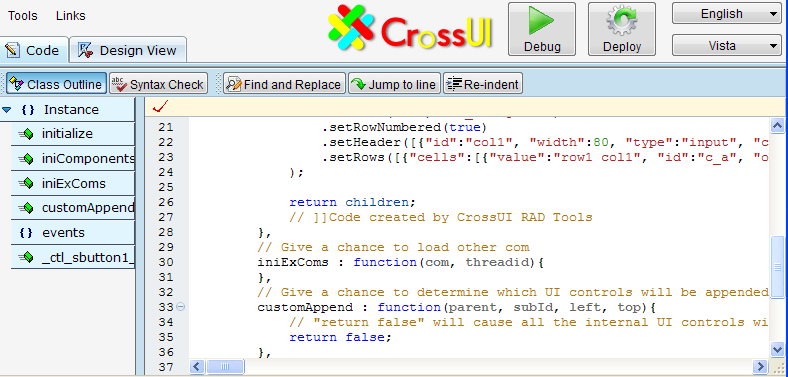
[xui.Module.load] new an instance of Class App, and render it into DOM

If Class code is bigger, it’ll take more time. In the meantime, users will see the waiting message (loading.gif picture or other custom showing)

## Code Editor

By the way, if you use CrossUI RAD online, in order to get better performance, Firefox and chrome are recommended here.

There are two views in Builder: “Design view” and “Code” view. In the online version, the default view is “Design view”. Click “Code” tab to switch to “Code” view.



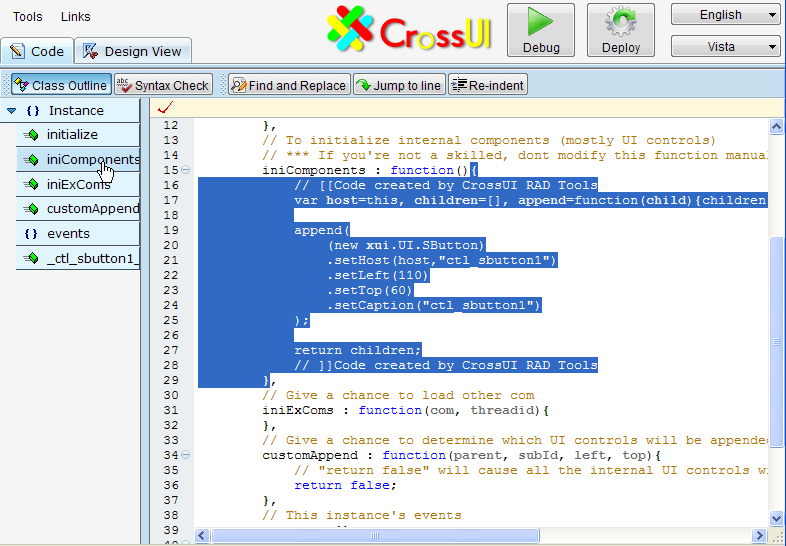
Class outline

Commands

Editor window

### Highlight code from Outline window

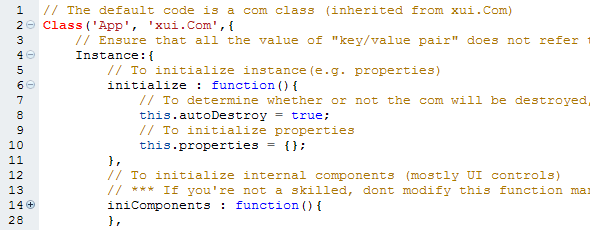
“Class Outline” is located in the left side of “Code” view. By clicking any member or method name in “Class Outline”, RAD Tools will highlight its code in “Editor window”, and scroll “Editor window” to show the code.



Click to show this function

### Code Folding

To make your code view more clear to read and understand, CrossUI RAD lets you fold certain parts of it. Click the left side “plus” or “minus” will fold or expand the block code.



To fold initialize function

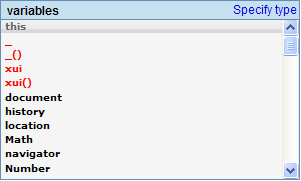
To expand iniComponents function

Note: Due to some browser’s poor performance, please try not to frequent collapse or expand the large body function or object.

### Code Intellisense

Three types Code Intellisense are supported.

* When context does not recognize the input string;
* Type dot after a recognizable variable
* When dbclick a recognizable variable

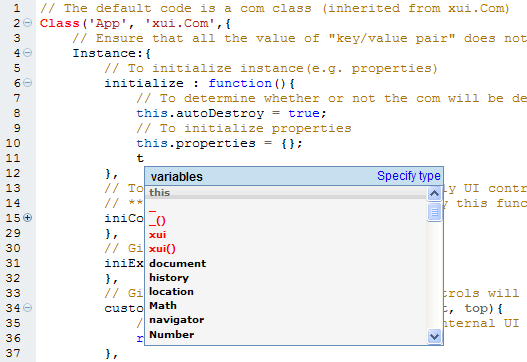


Keyboard actions for Code Intellisense pop Window:

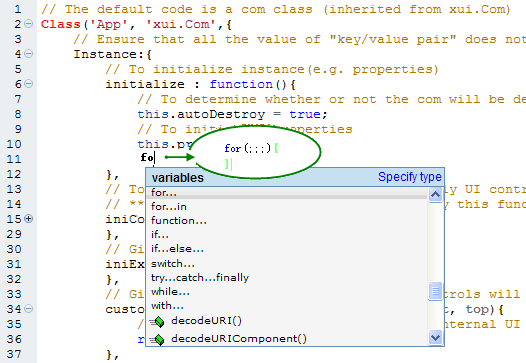
* + “up”: Focus to next item in code list
  + “down”: Focus to previous item in code list
  + “enter”: Select the current focused item, and input to editor window
  + “esc”: Close the pop window
  + Other visible chars: Find and focus the first matched item

#### When context doesn’t recognize the input string

When you input a string, if editor doesn’t recognize this string, it will pop a list window including local variables, global variables, global functions and JavaScript reserved keywords. In the below picture, type ‘t’ will trigger editor to pop a list window, “this” is the default focused item.



If the input string is “fo”, the “for loop statement” will be the default focused item.

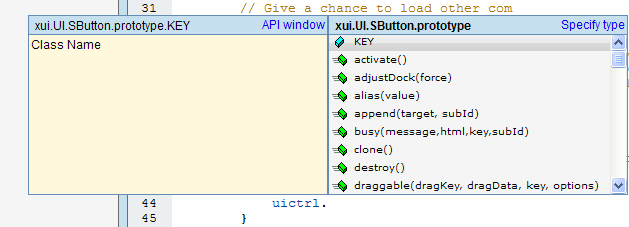


After press "Enter"

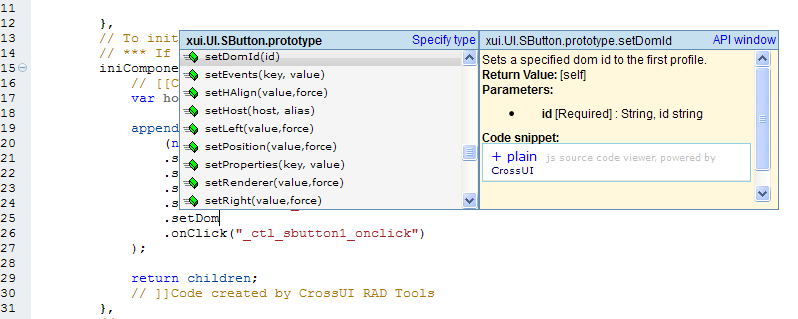
In this case, "Enter" keypress will cause “for loop statement” code to be inserted into the editor automatically.

#### Type dot after a recognizable variable

After an editor recognizable variable, if you type char “.”, editor will pop an available members and functions list for the variable.



Chainable methods can show Code Intellisense window too.

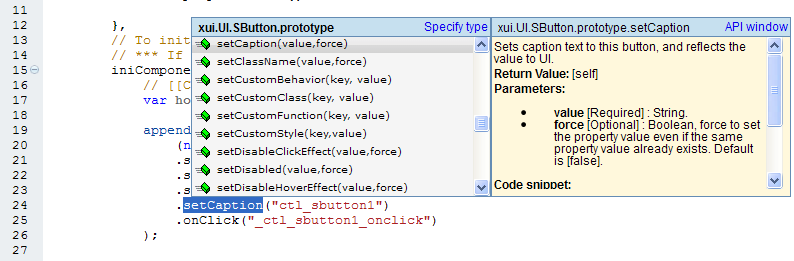


API document

Chainable methods

#### When use dbclick

Double click one variable string will trigger editor to pop the Code Intellisense window.

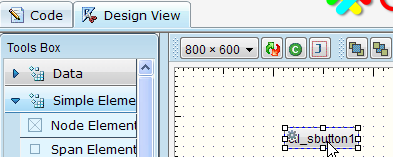


Double click “setCaption”

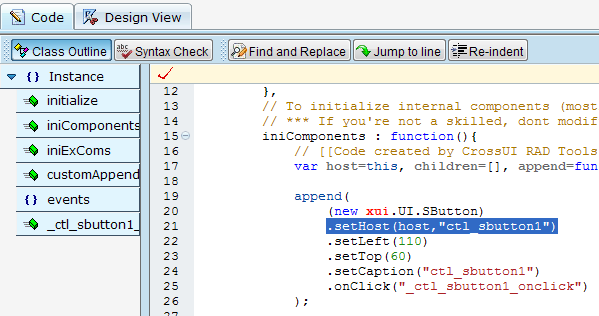
### Find the object definition code

In “Design View”, double click a control will cause:

1. Switch to “Code” view;
2. Highlight the control’s definition code;
3. Scroll the definition code to view.



dblclick

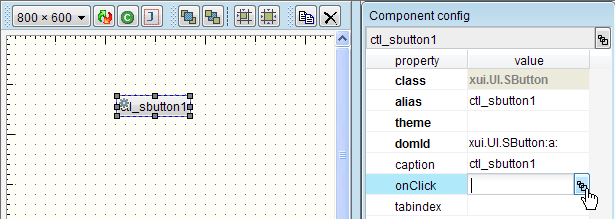


Definition code

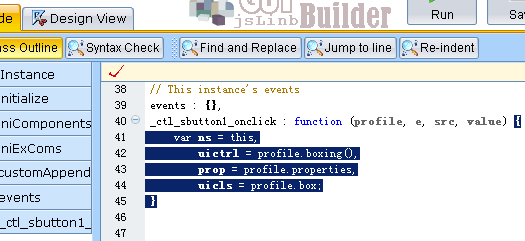
### Generate event code automatically

In the “Design View”, select a control; the right side “Component config” window will be refreshed. Find an event (e.g. onClick event), click its event button will cause:

1. Switch to “Code” view;
2. Create event code, and insert into the editor;
3. Scroll the event code to view.



Click



Event code created

# Controls Facebook

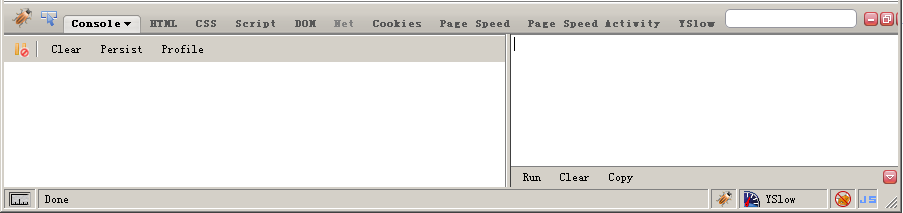
Many beginners are particularly interested in UI controls. In this chapter we’ll give a rough look at the basic controls. Since each control has a lot of functions, here is a brief introduction, it is impossible to explain all the functions. You can browse API to understand the specific function of each control in detail!

## Script testing environment

At first, we have to build a testing environment for executing example codes. About Browsers, Firefox is recommended, if Firefox is not preferred, ie8 or chrome is ok too.

**For Firefox:**

1. You need Firefox and firebug;
2. Ensure all files and folders in cookbook package including “**env.html**” are in **cookbook** dir;
3. Open URL **cookbook**/**env.html** in Firefox;
4. Open firebug console, switch to the multi-line mode

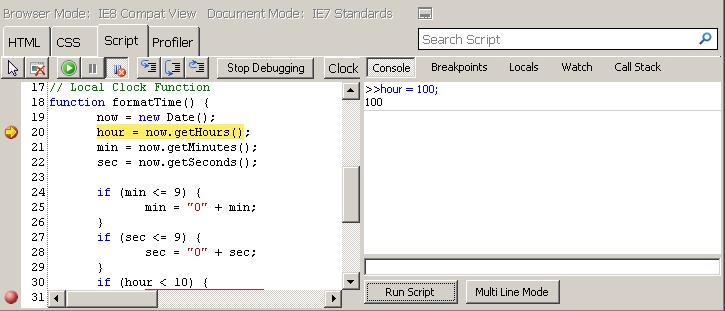


Then, click “run”

Enter the test code here

**For IE8+:**

1. You need IE8;
2. Open URL **cookbook**/**env.html** ;
3. Open developer tools, switch to the multi-line mode



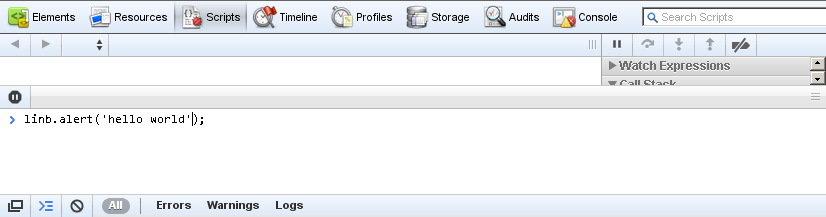
multi line mode is better

Enter the test code here

Then, click “run”

**For Chrome:**

1. You need the latest Chrome;
2. Open URL **cookbook**/**env.html** ;
3. Open developer tools



Enter the test code here, press ‘enter’ to run

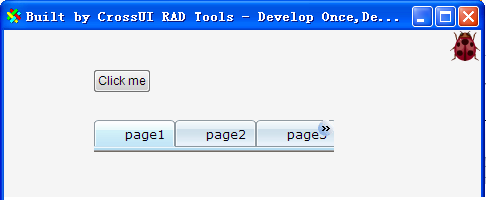
There’s a “Clear” button in **cookbook**/**env.html**, You can click this button to clean up the current page’s DOM. In some cases, you want to clean up both DOM and memory, press ‘F5’ to refresh your browser.



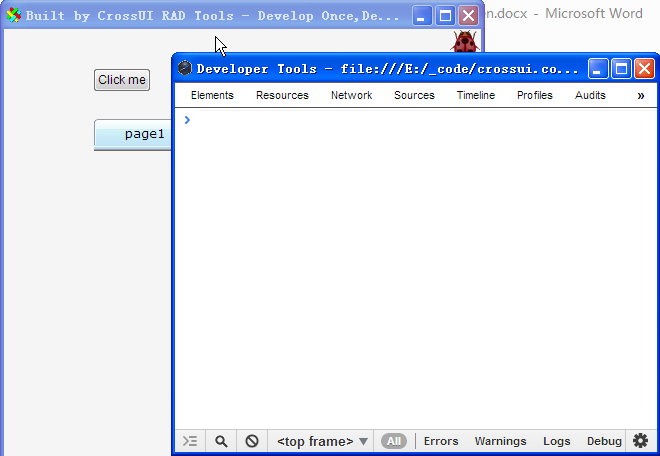
**For CrossUI Desktop version:**



To show debug window



You can press [F12] to show the Developer Tools window.



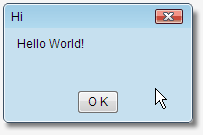
Desktop version’s “development tools window”

## “Hhello world” in env.html

Input the following code into script window, and run it.

xui.alert("Hi", "Hello World!");

**Output:**



Click “Clear” button to clean the DOM.

If you are in CrossUI Desktop, execute the following line in console:

xui ("body"). empty();

## Control creation and runtime update

There are three approaches to create CrossUI control.

// Approach 1

xui.create("SButton", {

caption: "Using xui.create function",

position: "relative"}

).show();

// Approach 2

(new xui.UI.SButton({

caption: "Using new and key/value pairs",

position: "relative"

})).show();

// Approach 3

(new xui.UI.SButton())

.setCaption("Using new and get/set")

.setPosition("relative")

.show();

We use new/setXX mode in RAD Tools

The above three approaches will create entirely consistent UI.

You can use setXXX function to update the control after it was rendered into DOM (runtime update).

var dlg=xui.create("Dialog", {caption: "runtime "}).show();

\_.asyRun(function(){

dlg.setCaption("updated");

},500);

\_.asyRun(function(){

dlg.setMaxBtn(false);

},1000);

\_.asyRun(function(){

dlg.setStatus("max");

},1500);

\_.asyRun(function(){

dlg.destroy();

},2000);

To modify caption

Create a Dialog

To hide the max button

To modify status

To destroy it

## Button related

This section relates to the following controls: xui.UI.Link, xui.UI.SButton, xui.UI.Button, xui.UI.SCheckBox and xui.UI.CheckBox.

### onClick event

**Input:**

var btn=new xui.UI.SButton();

btn.setCaption("Click Me")

.onClick(function(){

xui.alert("Hi","You are great!");

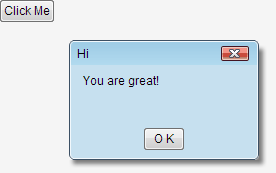
});

btn.show();

Adds onClick evnet

Sets caption

**Output:**



Click it

**Input:**

var btn=new xui.UI.Button();

btn.setCaption("Click Me")

.onClick(function(){

xui.alert("Hi","You are great!");

});

btn.show();

\_.asyRun(function(){

btn.setHeight(80)

.setShadow(true)

.setType("drop")

},1000);

Adds onClick event

Sets caption

Execute code after 1 second

**Output:**



**NOTE**

**xui.UI.SButton / SLabel / SCheckbox** are enough for most cases; Only if you need more complex feature, you should use those complex control: **xui.UI.Button / Label / Checkbox**.

### Boolean Controls

There are three controls can represent and modify Boolean value:

**Input:**

var btn= (new xui.UI.Button({position: "relative", caption:"Button", type:"status"})).show();

var scb= (new xui.UI.SCheckBox({position: "relative", caption:" SCheckBox"})).show();

var cb= (new xui.UI.CheckBox({position: "relative", caption:" CheckBox"})).show();

\_.asyRun(function(){

btn.setValue(true,true);

scb.setValue(true,true);

cb.setValue(true,true);

},1000);

Sets position to ‘relative’

Sets values to true after 1 second

**Output:**



### Link Control

You can take xui.UI.Link as a simple button.

**Input:**

var btn=new xui.UI.Link();

btn.setCaption("Click Me")

.onClick(function(){

xui.alert("Hi","You are great!");

});

btn.show();

// href property

xui.create("Link",{href: <http://www.crossui.com>, target: "\_blank"}).show(null,null,100,100)

// href was disabled when return false

xui.create("Link",{href: "http://www.longboo.com"}).show(null,null,200,100)

.onClick(function(){

return false;

});

Adds onClick event

Sets caption

**Output:**



Goes to crossui.com

Shows alert window

No action

## Label related

This section relates to the following controls: xui.UI.SLabel, xui.UI.Label and xui.UI.Div. These three controls can be used as “label”, xui.UI.SLabel is the simplest one, but it’s enough for most cases; If you need more complex feature like shadow, resizer or border, you should choose xui.UI.Label; Or if you want to input more complex html code in the control, xui.UI.Div is better.

**Input:**

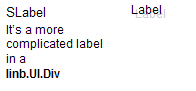
(new xui.UI.SLabel()).setCaption("SLabel").setPosition("relative").show();

(new xui.UI.Label()).setCaption("Label").setPosition("relative").setShadowText(true).show();

(new xui.UI.Div()).setHtml("It’s a more complicated label in a <br /><b>xui.UI.Div</b>")

.setPosition("relative").show();

**Output:**



SLabel

Label

Div

## Input related

This section relates to the following controls: xui.UI.Input, xui.UI.ComboInput and xui.UI.RichEditor. xui.UI.ComboInput is an enhanced version of xui.UI.Input, it can input/edit value through a pop window; xui.UI.RichEditor is a rich text input/edit control.

### setValue/setValue/getUIValue/setUIValue

From the users point of view, value controls (all derived from the xui.absValue control) in CrossUI has two values has two values: the “UI value”(**getUIValue/setUIValue**) and the “control value”(**getValue/setValue**).

“UI value” does not always equal to “control value”. For example, for an empty input control

1. Keyboard input “**abc**”: “UI value” is “**abc**”, “control value” is **empty**;
2. Calls “updateValue” function: “UI value” is “**abc**”, “control value” is “**abc**”;
3. Calls “setValue(‘**bcd’**)”: “UI value” is “**bcd**”, “control value” is “**bcd**”;
4. Calls “setUIValue(‘**efg’**)”: “UI value” is “**efg’**”, “control value” is “**bcd**”
5. Calls “resetValue(‘**x**’)”: “UI value” is “**x**”, “control value” is “**x**”;

var input = (new xui.UI.Input()).show();

xui.message(input.getUIValue()+":"+input.getValue());

\_.asyRun(function(){

input.setUIValue('uivalue');

xui.message(input.getUIValue()+":"+input.getValue());

},2000);

\_.asyRun(function(){

input.updateValue();

xui.message(input.getUIValue()+":"+input.getValue());

},4000);

A new Input

Sets UI value

Updates UI value to control value

You can go to <http://www.crossui.com/xui/Examples/comb/DataBinder/index.html> for more information about it.

### Dirty Mark

If the control’s dirtyMark property is set to true, when “UI value” does not equal to “control value”, a “Dirty Mark” will appear. The “Dirty Mark” will disappear when “UI value” equals to “control value”.

.

Dirty Mark

var input = (new xui.UI.Input()).show();

\_.asyRun(function(){

input.setUIValue('uivalue');

},1000);

\_.asyRun(function(){

input.updateValue();

input.setDirtyMark (false);

},2000);

\_.asyRun(function(){

input.setUIValue('uivalue 2');

},3000);

Dirty Mark appears

Dirty Mark disappears

Nothing happen

If DirtyMark is disabled

### Password Input

Sets Input’s type property to “password”.

**Input:**

var input = (new xui.UI.Input({type: 'password'})).show();

\_.asyRun(function(){

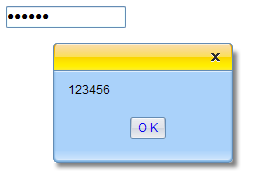
input.setUIValue('123456').updateValue();

xui.pop(input.getValue());

},1000);

Sets type

**Output:**



### Multi-lines

Sets Input’s multiLine property to true.

**Input:**

var input = (new xui.UI.Input()).setMultiLines(true).setHeight(100).show();

\_.asyRun(function(){

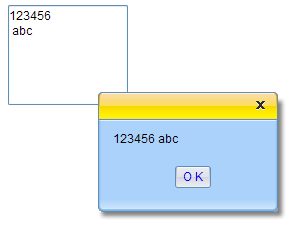
input.setUIValue('123456 \n abc').updateValue();

xui.pop(input.getValue());

},1000);

Sets multiLine to true

**Output:**



### Input validation

#### valueFormat property

“valueForamt” property represents a regular expression.

**Input:**

var input = (new xui.UI.Input())

.setValueFormat("^-?\\d\\d\*$")

.show();

Number only

Executes the above code, input some charts, and let it lose the mouse focus, the “Error Mark” will appear.



#### beforeFormatCheck event

**Input:**

var input = (new xui.UI.Input())

.beforeFormatCheck(function(profile,value){

if(value!=parseFloat(value).toString())

return false;

})

.show();

Number only

In above methods, “beforeFormatCheck” has priority. That means, when "beforeFormatCheck" returns ‘false’, "valueFormat" property will be ignored.

### Dynamic input validation

In previous section examples, “Error Mark” appears only when the control loses focus. If you want to a real-time input validation , you need to set dynCheck property to true.

var input = (new xui.UI.Input())

.setDynCheck(true)

.setValueFormat("^-?\\d\\d\*$")

.show();

Sets dynCheck

### Error Mark

#### Default Error Mark

The default “Error Mark” is an icon at the right side of Input.



The default Error Icon

#### Validation Tips

There are three tool tips in xui.UI.Input control:

* tips: the default tool tips
* tipsOK: the valid tool tips
* tipsErr: the invalid tool tips

**Input:**

var input = (new xui.UI.Input())

.setTips("default tips")

.setTipsErr("invalid tips ")

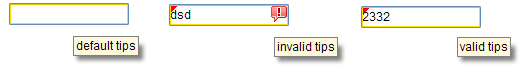
.setTipsOK("valid tips")

.setValueFormat("^-?\\d\\d\*$")

.show();

Sets those tips

**Output:**



#### Binding Validation

You can bind the validation tips to a xui.UI.Div, xui.UI.SLabel or xui.UI.Span.

**Input:**

var slbl= (new xui.UI.SLabel({position:'relative'})).setCustomStyle({KEY:'padding-left:10px'});

var input = (new xui.UI.Input({position:'relative'}))

.setValueFormat("^-?\\d\\d\*$")

.setTipsBinder(slbl)

.setDynCheck(true)

.setTips(" default tips")

.setTipsErr(" invalid tips ")

.setTipsOK(" valid tips")

input.show();

slbl.show();

Show SLabel here

Sets tipsBinder

**Output:**



#### Custom Error Mark

We can custom “Error Mark” in beforeFormatMark event.

**Input:**

var input = (new xui.UI.Input())

.setValueFormat("^-?\\d\\d\*$")

.beforeFormatMark(function(profile,err){

if(err)

xui.alert("Invalid input!","Only number allowed!",function(){

profile.boxing().activate();

});

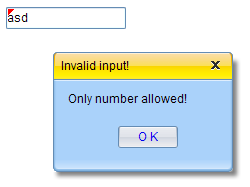
return false;

}).show();

Return false to ignore the default action

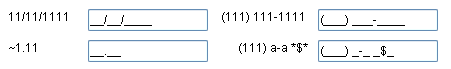
Customs information and aciton

**Output:**



### Mask Input

Mask Input examples:



In **chapter2\Input\index.html**

There is a mask property in xui.UI.Input control. It’s a string. In this string,

* ‘~’ represents [+-]
* ‘1’ represents [0-9]
* ‘a’ represents [A-Za-z]
* ‘u’ represents [A-Z]
* ‘l’ represents [a-z]
* ‘\*’ represents [A-Za-z0-9]
* Other visible char represents itself

**Input:**

var input = (new xui.UI.Input())

.setMask("(1111)11111111-11")

.show();

**Output:**



**NOTE**

**chapter2\Input\index.html** is an overall example for Input.

### xui.UI.ComboInput

xui.UI.ComboInput is an advanced Input.

#### Pop list for selection

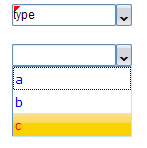
When type property was set to “combobox”, “listbox” or “helpinput”, click the command button will trigger to pop a list window for selection.

listbox

(new xui.UI.ComboInput({items:[ "a","b","c"]})).show();

(new xui.UI.ComboInput({type: "listbox", top:"100", items:[ "a","b","c"]})).show();

combobox



readonly

Keyboard input

#### combobox, listbox and helpinput

There’s an items property in xui.UI.ComboInput (And all list related controls have this property too). Usually, we set items as a simple single layer array (like "[ia’, ‘ib’, ‘ic’]"). Framework will convert this simple array to inner format:

[

{

id : "ia",

caption : "ia"

},

{

id : "ib",

caption : "ib"

},

{

id : "ic",

caption : "ic"

}

]

* 1. combobox: Not readonly. The pop List shows “caption”; Input box shows “caption”; getValue returns “caption”.
  2. listbox: Readonly. The pop List shows “caption”; Input box shows “caption”; getValue returns “id.
  3. helpinput: Not readonly. The pop List shows “caption”; Input box shows “id”; getValue returns “id”.

**Input:**

var items=[

{

id : "id1",

caption : "caption1"

},{

id : "id2",

caption : "caption2"

},{

id : "id3",

caption : "caption3"

}

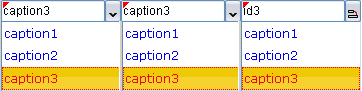
];

xui.create('ComboInput',{position:'relative',items:items}).show();

xui.create('ComboInput',{position:'relative',items:items,type:'listbox'}).show();

xui.create('ComboInput',{position:'relative',items:items,type:'helpinput'}).show();

**Output:**



#### Date Piker

Sets type property to “date”.

**Input:**

var ctrl=xui.create('ComboInput')

.setType('date')

.setValue(new Date)

.show();

\_.asyRun(function(){

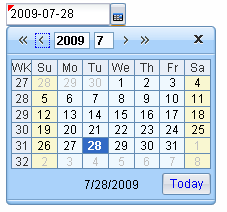
alert("The value is a timestamp string:"+ctrl.getValue());

alert("You can convert it to date object:"+new Date(parseInt(ctrl.getValue())));

});

Date object or timestamp string

**Output:**



#### Time Picker

Sets type property to “time”.

**Input:**

var ctrl=xui.create('ComboInput')

.setType('time')

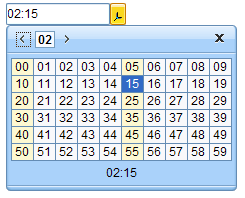
.setValue('2:15')

.show();

xui.alert("The value is a string : "+ctrl.getValue());

Sets string

**Output:**



#### Color Picker

Sets type property to “color”.

**Input:**

var ctrl=xui.create('ComboInput')

.setType('color')

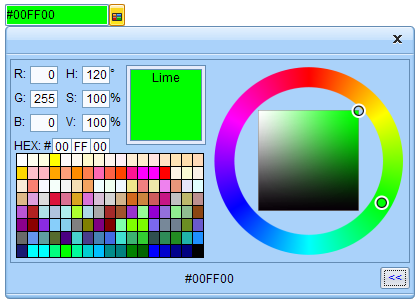
.setValue('#00ff00')

.show();

xui.alert("The value is a string : "+ctrl.getValue());

Sets string

**Output:**



#### File Picker

Sets type property to “upload”.

**Input:**

var ctrl=xui.create('ComboInput')

.setType('file')

.show();

**Output:**



Note: use getUploadObj function to get the file’s handler

ctrl.getUploadObj()

#### Getter

Sets type property to “getter”.

**Input:**

var ctrl=xui.create('ComboInput')

.setType('getter')

.beforeComboPop(function(profile){

profile.boxing().setUIValue(\_.id())

})

.show();

Sets value in beforeComboPop event

**Output:**



#### Custom Pop Window

Sets type property to “cmdbox”, or “popbox”.

**Input:**

var ctrl=xui.create('ComboInput')

.setType('popbox')

.beforeComboPop(function(profile){

var dlg=new xui.UI.Dialog, tb;

dlg.append(tb=new xui.UI.TreeBar({items:["a","b",{id:"c",sub:["c1","c2","c3"]}]}));

tb.onItemSelected(function(profile,item){

ctrl.setUIValue(item.id);

dlg.destroy();

});

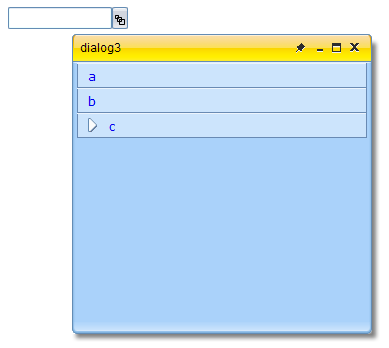
dlg.show(null,true,100,100)

})

.show();

Shows custom pop window in beforeComboPop event

**Output:**



The custom pop window

#### Command Buttons

You can use commandBtn property to add an command button into ComboInput control. The following types are available for commandBtn property:

* “none”: no command button
* “save”: It’s a save button
* “add” : It’s a add button
* “remove” : It’s a remove button
* “delete” : It’s a delete button
* “custom” : custom button ( sets imageClass or mage,/imagePos to custom it)

**Input:**

(new xui.UI.ComboInput).setPosition('relative').setCommandBtn('none').show();

(new xui.UI.ComboInput).setPosition('relative').setCommandBtn('save').onCommand(function(){ xui.alert('save event'); }).show();

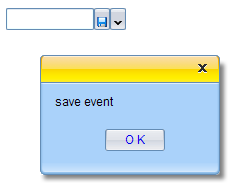
(new xui.UI.ComboInput).setPosition('relative').setCommandBtn('add').onCommand(function(){ xui.alert('add event'); }).show();

(new xui.UI.ComboInput).setPosition('relative').setCommandBtn('remove').onCommand(function(){ xui.alert('remove event'); }).show();

(new xui.UI.ComboInput).setPosition('relative').setCommandBtn('delete').onCommand(function(){ xui.alert('delete event'); }).show();

(new xui.UI.ComboInput).setPosition('relative').setCommandBtn('save').onCommand(function(){ xui.alert('save event'); }).show();

**Output:**



**NOTE**

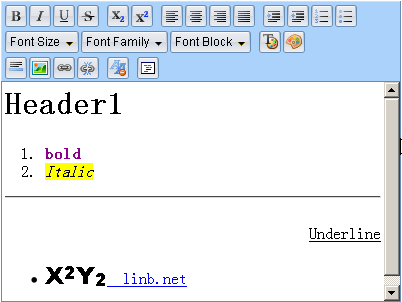
**chapter2\ComboInput\index.html** is an overall example for ComboInput.

### RichEditor

**Input:**

(new xui.UI.RichEditor()).show();

**Output:**



## List related

This section relates to the following controls: xui.UI.List, xui.UI.RadioBox and xui.UIIconList and xui.UI.Gallery.

### A Simple one

xui.create("List")

.setItems(["Item one","Item two","Item tree"])

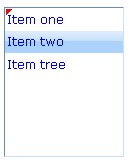
.onItemSelected(function(profile,item){

xui.message(item.id);

})

.show();

onItemSelected event



### A little bit complicated

var renderer=function(o){

return '<span style="width:40px">'+o.col1+"</span>" + ' <span style="width:60px">'+o.col2+"</span>" + ' <span style="width:40px">'+o.col3+"</span>";

};

xui.create("List")

.setWidth(160)

.setItems([{

id:"a",

col1:'Name',

col2:'Gender ',

col3:'Age',

renderer:renderer,

itemStyle:'border-bottom:solid 1px #C8E1FA;font-weight:bold;'

},

{

id:"b",

col1:'Jack',

col2:'Male',

col3:'23',

renderer:renderer

},

{

id:"c",

col1:' Jenny',

col2:'Female',

col3:'32',

renderer:renderer

}]

)

.beforeUIValueSet(function (profile, ov, nv){

return nv!=="a"

})

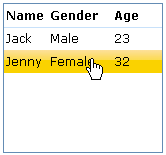
.show();

Gives a render function

Extra variables

For the header item

**Result:**



A Grid List

The above special render function applies to any control’s caption property (e.g. xui.UI.Button, xui.UI.Label); and any control’s sub item caption property (e.g. xui.UI.List, xui.UI.TreeBar) .

xui.create("SCheckBox")

.setCaption("caption")

.setRenderer(function(prop){return prop.caption+"+"+this.key})

.show();



### RadioBox

xui.UI.RadioBox is derived from xui.UI.List.

**Input:**

xui.create("RadioBox")

.setItems(["a","b","c"])

.onItemSelected(function(profile,item){

xui.message(item.id);

})

.show();

**Output:**



### IconList and Gallery

Both are derived from xui.UI.List.

**Input:**

xui.create("IconList")

.setItems([{id:'a',image:'img/a.gif'},{id:'b',image:'img/b.gif'},{id:'c',image:'img/c.gif'}])

.onItemSelected(function(profile,item){

xui.message(item.id);

})

.show();

**Output:**



**Input:**

xui.create("Gallery")

.setItemWidth(64).setItemHeight(64)

.setItems([{id:'a',image:'img/a.gif',caption:'User',comment:'It’s a user'},{id:'b',image:'img/b.gif',caption:'Computer',comment:'It’s a computer'},{id:'c',image:'img/c.gif',caption:'Tree',comment:'It’s a tree'}])

.onItemSelected(function(profile,item){

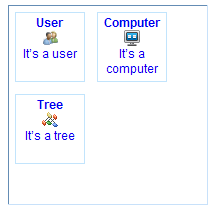
xui.message(item.id);

})

.show();

Item’s height

**Output:**



### Item selection

You can use “**setUIValue**” function to select an item in List, or use “**fireItemClickEvent**” function to get the same result. “fireItemClickEvent” function will trigger “onItemSelected” event, “setUIValue” won’t.

var ctrl=xui.create("List")

.setItems(["Item one","Item two","Item tree"])

.onItemSelected(function(profile,item){

xui.message(item.id);

})

.show();

\_.asyRun(function(){

ctrl.fireItemClickEvent("Item two");

},1000);

\_.asyRun(function(){

ctrl.setUIValue("Item one");

},2000);

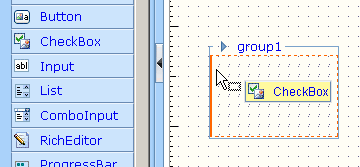
Trigger onItemSelected event

### Container related

This section relates to the following controls: xui.UI.Group, xui.UI.Pane , xui.UI.Panel, xui.UI.Block.

xui.UI.Dialog, xui.UI.Layout and xui.UI.Tabs /Stacks/ButtonViews are container controls too, we will give examples of these controls in separate sections.

Container is those controls that can have child controls. In CrossUI RAD Designer, you can drag a child control and drop it into a container control. Just like this,



**Input 1:**

(new xui.UI.Group)

.append(new xui.UI.SButton)

.show();

append

**Input 2:**

var con = new xui.UI.Group;

con.show();

(new xui.UI.SButton).show(con);

show

**Input 3:**

xui.create({

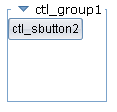
key:"xui.UI.Group",

children:[[{key:"xui.UI.SButton"}]]

}).show();

In children object

**Output:**



### Pane and Panel

xui.UI.Pane is a single node control. It’s derived from xui.UI.Div. xui.UI.Panel has a border and a title bar.

**Input:**

(new xui.UI.Pane)

.append(new xui.UI.SButton)

.show()

You can’t see output, It’s transparent

**Input:**

(new xui.UI.Panel)

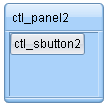
.setDock("none")

.append(new xui.UI.SButton)

.show()

Sets dock to ‘none’

**Output:**



### Block

**Input:**

xui.create("Block",{position:'relative',borderType:'none'}).show()

xui.create("Block",{position:'relative',borderType:'flat'}).show()

xui.create("Block",{position:'relative',borderType:'inset'}).show()

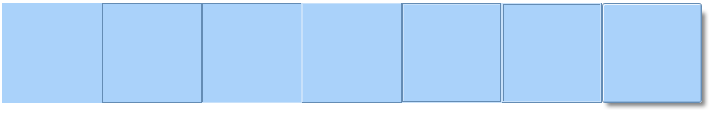
xui.create("Block",{position:'relative',borderType:'outset'}).show()

xui.create("Block",{position:'relative',borderType:'groove'}).show()

xui.create("Block",{position:'relative',borderType:'ridge'}).show()

xui.create("Block",{position:'relative',borderType:'none',border:true,shadow:true,resizer:true}).show()

**Output:**



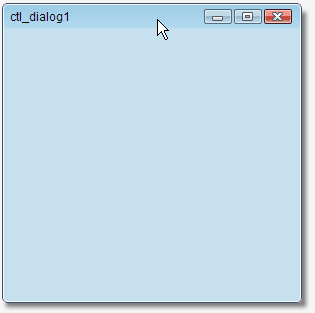
## Dialog related

### Normal state

**Input:**

(new xui.UI.Dialog).show()

**Output:**



**Input:**

var dlg = (new xui.UI.Dialog).show();

var panel;

\_.asyRun(function(){

dlg.append(panel=new xui.UI.Panel)

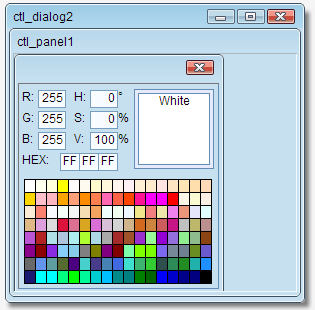
},1000);

\_.asyRun(function(){

panel.append(new xui.UI.ColorPicker)

},2000);

**Output:**



### Min and Max status

**Input:**

var dlg = (new xui.UI.Dialog). setStatus("min").show();

\_.asyRun(function(){

dlg.setStatus("normal");

},1000);

\_.asyRun(function(){

dlg.setStatus("max");

},2000);

**Output:**



### Modal Mode

**Input:**

var dlg = (new xui.UI.Dialog).show();

dlg.append(panel=new xui.UI.SButton({

caption: "Pop a modal dialog"

},

{onClick:function(){

xui.create("Dialog",{

width:200,

height:100,

html:"The second modal dialog"

}).**showModal**(dlg);

}}

))

(new xui.UI.Dialog)

.setHtml("The first modal dialog")

.show(null,true);

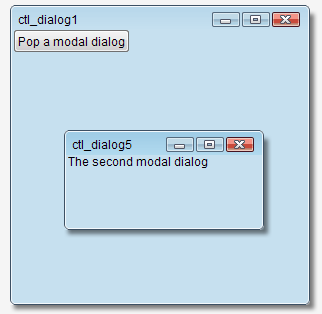
Parent is html body

Parent is dlg

Sets caption

onClick event

**Output:**



## Layout Control

**Input:**

var block=xui.create("Block").setWidth(300).setHeight(300);

var layout=xui.create("Layout",{items:[

{id:'before',

pos:'before',

size:100,

cmd:true

},{id:'after',pos:'after',size:100}

]});

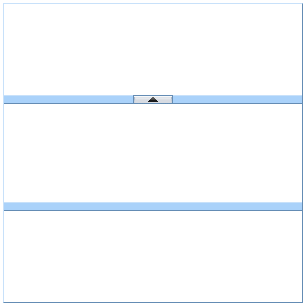
block.append(layout).show();

Size to 100

Has a command button

Append into a block

**Output:**



Fold/Expand command button

Main container

Before container

After container

**Input:**

var block=xui.create("Block").setWidth(400).setHeight(100);

var layout=xui.create("Layout",{items:[

{id:'before',

pos:'before',

size:100,

cmd:true,

folded:true,

max:120,

min:80

},{

id:'after',

pos:'after',

cmd:true,

locked:true,

size:50

},{

id:'after2',

pos:'after',

size:50

}],type: 'horizontal'});

block.append(layout).show();

Max size is120

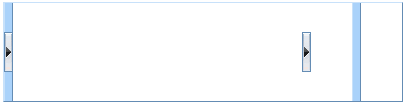
Min size is 80

horizontal

Default status is fold

Size locked

**Output:**



Without command button, with split bar

fold

With command button, without split bar

**NOTE**

**chapter2\Layout\index.html** is an overall example for Layout.

## Multi-pages Controls

Three multi-pages controls: xui.UI.Tabs, xui.UI.Stacks and xui.UI.ButtonViews.

**Input:**

var block=xui.create("Block").setWidth(400).setHeight(100);

var pages=xui.create("Tabs",{

items:["page1","page2","page3"],

value:"page2"

});

block.append(pages).show();

\_.asyRun(function(){

pages.append(new xui.UI.SButton,"page2")

},1000);

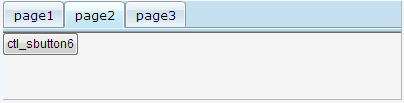
3 pages

The default page

Adds a SButton to 2th page

Append to a block

**Output:**



### noPanel property

For xui.UI.Tabs and xui.UI.ButtonViews, when “noPanel” property was set to true, they no longer are the container control. So, don’t append any children control to tabs in this case.

**Input:**

var block=xui.create("Block").setWidth(400).setHeight(300).show();

var items=["page1","page2","page3"];

xui.create("Tabs",{

items:items,

value:"page2",

position:'relative',

width:'auto',

height:'auto',

dock:'none',

noPanel:true

}).show(block);

xui.create("ButtonViews",{

items:items,

value:"page2",

position:'relative',

width:'auto',

height:32,

barSize:30,

dock:'none',

noPanel:true

}).show(block);

Auto width

Set position to ‘relative’

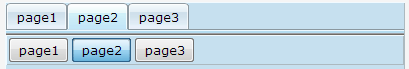
No container

Auto height

No container

Set height to buttonview

**Output:**

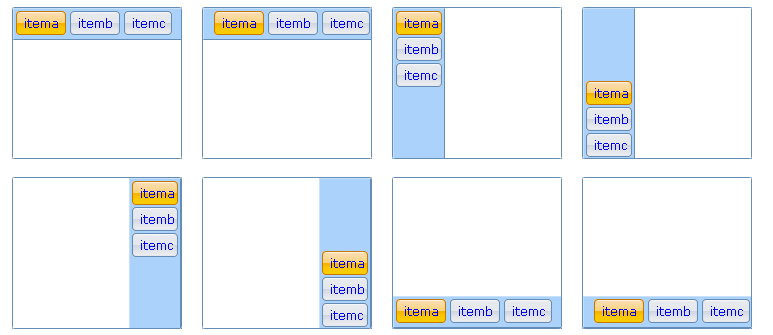


### ButtonViews types

There are three properties used to define the ButtonViews’ layout:

* **barLocation**: Used to set the location of the command button bar.   
  In ‘top’,’bottom’,’left’,’right’.
* **barHAlign**: Used to set command buttons horizontal alignment  
  In ‘left’, ‘right’. Only for **barLocation** is ‘top’ or ‘bottom’
* **barVAlign**: Used to set command buttons vertical alignment  
  In ‘left’, ‘right’. Only for **barLocation** is ‘left’ or ‘right’

The below picture shows all the eight possible ButtonViews layouts:



In **chapter2\ButtonViews\index.html**

**NOTE**

**chapter2\ButtonViews\index.html** is an overall example for ButtonViews.

### Page selection

You can use “**setUIValue**” function to select a page, or use “**fireItemClickEvent**” function to get the same result. “fireItemClickEvent” function will trigger “onItemSelected” event, “setUIValue” won’t.

**Input:**

var block=xui.create("Block").setWidth(400).setHeight(100).show();

var pages=xui.create("Tabs",{

items:["page1","page2","page3"]

})

.onItemSelected(function(profile,item){

xui.message(item.id);

})

.show(block);

\_.asyRun(function(){

pages.fireItemClickEvent("page2");

},1000);

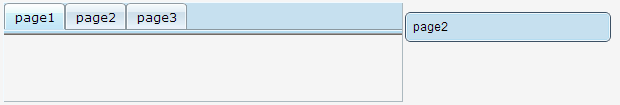
\_.asyRun(function(){

pages.setUIValue("page1");

},2000);

Trigger onItemSelected event

**Output:**



### Pages

#### Close and options Button

Each page can hold a “close” button and a “options” button. Click this button will close the page.

**Input:**

var block=xui.create("Block").setWidth(200).setHeight(400).show(), stacks;

block.append(stacks=new xui.UI.Stacks({

value:'a',

items:[{

id:'a',

caption:'First Page',

closeBtn:true

},{

id:'b',

caption:'Second Page',

optBtn:true

},{

id:'c',

caption:'Third Page'

},{

id:'d',

caption: 'Fourth Page'

}]

}));

stacks.onShowOptions(function(profile,item){

xui.message(" You clicked "+item.caption)

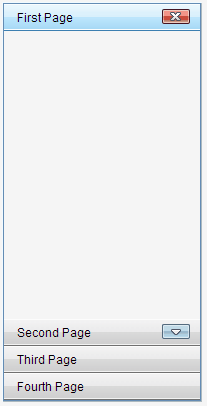
});

Close button

Options button

Click options to trigger onShowOptions event

**Output:**



Close button

Options button

Two events can be fired when “close” button was clicked:

* beforePageClose: Fired before user clicked the close button on a page. If returns false, the page won’t be closed.
* afterPageClose: Fired after user clicked the close button on a page.

#### Add/Remove Pages

**Input:**

var block=xui.create("Block").setWidth(400).setHeight(100).show(), tabs;

block.append(tabs=new xui.UI.Tabs({

value:'a',

items:[{

id:'a',

caption:'First Page'

},{

id:'b',

caption:'Second Page'

}]

}));

\_.asyRun(function(){

tabs.insertItems([{

id:'c',

caption:'Third Page'

},{

id:'d',

caption:'Fourth Page'

}]);

},500);

\_.asyRun(function(){

tabs.insertItems('Fifth Page');

},1000);

\_.asyRun(function(){

tabs.removeItems('d');

},1500);

\_.asyRun(function(){

tabs.removeItems(['b','c']);

},2000);

Close button

Adds two pages

Adds one more

Removes this page

Removes two more

### Dynamic content loading

#### onIniPanelView

var block=xui.create("Block").setWidth(400).setHeight(100).show(),

tabs=new xui.UI.Tabs({

value:'a',

items:[{

id:'a',

caption:'First Page'

},{

id:'b',

caption:'Second PageAgeeablellByRowColtem.caption+lue paires'

},{

id:'c',

caption:'Third Page'

}]

});

tabs.onIniPanelView(function(profile,item){

profile.boxing().getPanel(item.id).append(new xui.UI.SButton)

});

block.append(tabs);

#### beforeUIValueSet/afterUIValueSet

It’s a fine-grained mechanism.

var block=xui.create("Block").setWidth(400).setHeight(100).show(), tabs;

block.append(tabs=new xui.UI.Tabs({

value:'a',

items:[{

id:'a',

caption:'First Page'

},{

id:'b',

caption:'Second Page'

},{

id:'c',

caption:'Third Page'

}]

}));

tabs.beforeUIValueSet(function(profile,ovalue,value){

if(value=='b')

return false;

});

tabs.afterUIValueSet(function(profile,ovalue,value){

if(value=='c'){

var item=profile.getItemByItemId(value);

if(!item.$ini){

profile.boxing().append(new xui.UI.SButton);

item.$ini=true;

}

}

});

Cancel selection

Checks flag

Sets a flag

Gets item object

## Menus and toolbars

### Pop Menu

**Input:**

var pm=xui.create('PopMenu')

.setItems([

{"id":"itema", "caption":"itema", "tips":"item a"},

{"type":"split"},

{"id":"itemb", "type":"checkbox", value:true,"caption":"itemb", "tips":"item b"},

{"id":"itemc", "caption":"itemc", "type":"checkbox", "tips":"item c"},

{"id":"itemd", "caption":"itemd", "tips":"item d",sub:[

{"id":"itemd1", "caption":"itemd1"},

{"id":"itemd2", "caption":"itemd2"}

]},

{"id":"iteme", "caption":"iteme", "tips":"item d", disabled:true}

])

.onMenuSelected(function(profile,item){

xui.message(item.id + (item.type=="checkbox"?" : " + item.value:""))

});

xui.create('SButton')

.onClick(function(profile){

pm.pop(profile.getRoot())

})

.show();

Checkbox type

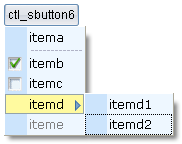
event

For position

Disabled it

Sub pop menu

**Output:**



### MenuBar

**Input:**

var pm=xui.create('MenuBar')

.setItems([{

"id" : "file", "caption" : "File",

"sub" : [{

"id" : "newproject",

"caption" : "New Project"

},

{ "id" : "openproject", "caption" : "Open Project",

"add" : "Ctrl+Alt+O",

"image" : "img/b.gif",

"sub":["option 1","option 2"]

},

{ "id" : "closeproject", "caption" : "Close Project"

},

{"type" : "split"},

{ "id" : "save", "caption" : "Save",

"image" : "img/a.gif"

},

{ "id" : "saveall", "caption" : "Save All",

"add" : "Ctrl+Alt+S",

"image" : "img/c.gif"

}]

},

{ "id" : "tools", "caption" : "Tools",

"sub" : [{ "id" : "command", "caption" : "Command Window"

},

{ "id" : "spy", "caption" : "Components Spy"

}]

},

{ "id" : "build", "caption" : "Build",

disabled:true,

"sub" : [{ "id" : "debug",

"caption" : "Debug"

}]

}]).show()

An icon

Sub pop menu

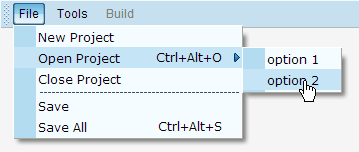
A split

Disabled it

Extra data

Pop menu data

**Output:**



### Toolbars

**Input:**

xui.create('ToolBar',{items:[{

"id" : "align",

"sub" : [

{"id" : "left","caption" : "left"},

{"id" : "center","caption" : "center"},

{type:'split'},

{"id" : "right","caption" : "center"}

]

},{

"id" : "code",

"sub" : [{

"id" : "format","caption" : "format",

label:"label",

image:"img/a.gif",

"dropButton" : true

}]

}]

})

.onClick(function(profile,group,item){

xui.message(group.id + " : " +item.id)

})

.show();

Button group data

With an icon

Group object

A split

Button data

With a label

A drop button

Button object

**Output:**



## TreeBar and TreeView

### Three selection mode

All controls derived from xui.UI.absList have three options mode.

#### No-selection

**Input:**

var block=new xui.UI.Block({width:200,height:200}).show();

xui.create("TreeBar",{items:[{ id : "itema", sub : ["suba","subb","subc","subd"]},

{id : "itemd", sub : ["sub1","sub2","sub3"]}]})

.setSelMode("none")

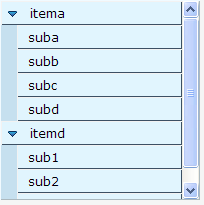
.onItemSelected(function(profile,item){

xui.message(item.id);

}).show(block);

Sets to ‘none’

**Output:**



#### Single-selection

**Input:**

var block=new xui.UI.Block({width:200,height:200}).show();

xui.create("TreeBar",{items:[{ id : "itema", sub : ["suba","subb","subc","subd"]},

{id : "itemd", sub : ["sub1","sub2","sub3"]}]})

.setSelMode("single")

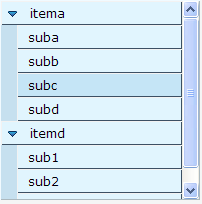
.onItemSelected(function(profile,item){

xui.message(item.id);

}).show(block);

Sets to single

**Output:**



#### Multi-selection

**Input:**

var block=new xui.UI.Block({width:200,height:200}).show();

xui.create("TreeBar",{items:[{ id : "itema", sub : ["suba","subb","subc","subd"]},

{id : "itemd", sub : ["sub1","sub2","sub3"]}]})

.setSelMode("multi")

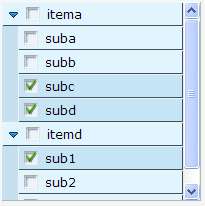
.onItemSelected(function(profile,item){

xui.message(item.id);

}).show(block);

Sets to ‘multi’

**Output:**



### Group Item

**Input:**

var block=new xui.UI.Block({width:200,height:200}).show();

xui.create("TreeBar",{items:[{

id : "itema",

image : "img/a.gif",

sub : ["suba","subb","subc","subd"]

},

{id : "itemb"},

{

id : "itemd",

group:true,

sub : ["sub1","sub2","sub3"]

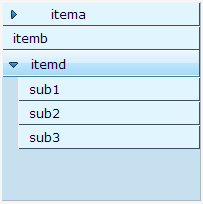
}]}).show(block);

With an icon

It’s a group

Sub items

**Output:**



Group item

### Expand all nodes by default

**Input:**

var block=new xui.UI.Block({width:200,height:200}).show();

xui.create("TreeBar",{

iniFold:false,

items:[{

id : "itema",

image : "img/a.gif",

sub : ["suba"]

},

{

id : "itemd",

group:true,

sub : ["sub1","sub2",{

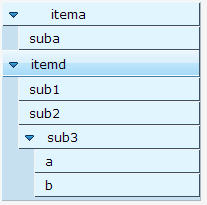
id:"sub3",sub:["a","b"]

}]

}]}).show(block);

Sets iniFold property to false

**Output:**



### Mutex Expand

var block=new xui.UI.Block({width:200,height:200}).show();

xui.create("TreeBar",{

singleOpen:true,

items:[{

id : "itema",

image : "img/a.gif",

sub : ["suba"]

},

{

id : "itemd",

group:true,

sub : ["sub1","sub2",{

id:"sub3",sub:["a","b"]

}]

}]}).show(block);

Mutex Expand

### Dynamic Destruction

var block=new xui.UI.Block({width:200,height:200}).show();

xui.create("TreeBar",{

dynDestory:true,

items:[{

id : "itema",

image : "img/a.gif",

sub : ["suba"]

},

{

id : "itemd",

group:true,

sub : ["sub1","sub2",{

id:"sub3",sub:["a","b"]

}]

}]}).show(block);

Dynamic Destruction

### Dynamically loading

**Input:**

var block=new xui.UI.Block({width:200,height:200}).show();

xui.create("TreeBar",{

singleOpen:true,

dynDestory:true,

items:[{

id : "itema",

sub : true

},

{

id : "itemb",

sub : true

}]})

. onGetContent(function(profile,item,callback){

if(item.id=="itema"){

var rnd=\_();

callback([rnd+"-a",rnd+"-b",rnd+"-c"]);

}

if(item.id=="itemb")

return ["itembsub1","itembsub2","itembsub3"];

})

.show(block);

Mutex Expand

Wants to load children dynamically

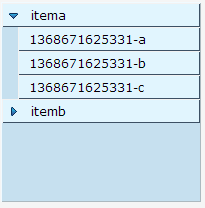
Asynchronous or synchronous callback

Can also be returned directly

Dynamic Destruction

Takes time stamp as a random string

Output:



dynamic created and destroyed

## TreeGrid

### Header and Rows

The header property and rows property in TreeGrid are Array of key/value pairs, like,

[

{id : "xxx1", caption : "xxx1" …,

sub: []

},

{id : "xxx2", caption : "xxx2" …},

…

]

[key/value pairs]

The sub [key/value pairs]

If no id specified, will create one automatically

It can be written as a simplified format,

[

"xx1",

"xx2",

{

id : "xxx3",

sub: ["sub1","sub2"]

}

]

Only id string

Only id string

When call setHeader/setRows, the simplified format can be convert to,

[

{id:"xx1",caption:"xx1"},

{id:"xx2",caption:"xx2"}

{

id : "xxx3", caption : "xxx3",

sub: [

{id:"sub1",caption:"sub1"},

{id:"sub2",caption:"sub2"}

]

}

]

#### Sets standard format

var block=new xui.UI.Block({width:200,height:200}).show();

var tg=new xui.UI.TreeGrid;

tg.setRowHandler(false)

.setHeader([

{id:"col1", caption:"Name"},

{id:"col2", caption:"Age", width:40}

]).setRows([

{id:"row1",cells:[{

value:'Jack',caption:'Jack'

},{

value:23,caption:'23'

}]},

{id:"row2",cells:[{

value:'John',caption:'John'

},{

value:32,caption:'32'

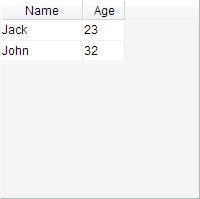
}]}

]).show(block);

Cells data

Column’s width

No row handler



#### Sets simplified format

var block=new xui.UI.Block({width:200,height:200}).show();

var tg=new xui.UI.TreeGrid;

tg.setRowHandler(false)

.setHeader(["Name", "Age"])

.setRows([['Jack', 23], ['John', 32]])

.show(block);

Only value input

Only id input

### getHeader

Calls getHeader function to return the header data. There are three format,

* getHeader(): returns memory data;
* getHeader("data"): returns the standard format data;
* getHeader("min"): returns the simplified format data;

var block=new xui.UI.Block({width:200,height:200}).show();

var tg=new xui.UI.TreeGrid;

tg.setRowHandler(false)

.setHeader(["Name", "Age"])

.setRows([['Jack', 23], ['John', 32]])

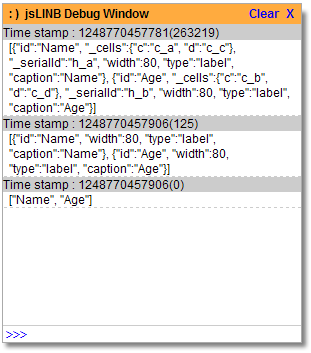
.show(block);

xui.log(tg.getHeader());

xui.log(tg.getHeader("data"));

xui.log(tg.getHeader("min"));

Comparing these three formats



### getRows

Calls getRows function to return the rows data. Similarly, there are three format,

* getRows (): returns memory data;
* getRows ("data"): returns the standard format data;
* getRows ("min"): returns the simplified format data;

var block=new xui.UI.Block({width:200,height:200}).show();

var tg=new xui.UI.TreeGrid;

tg.setRowHandler(false)

.setHeader(["Name", "Age"])

.setRows([['Jack', 23], ['John', 32]])

.show(block);

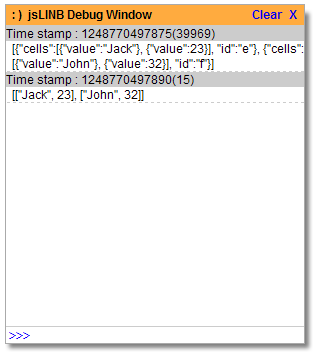
xui.log(tg.getRows("data"));

xui.log(tg.getRows("min"));

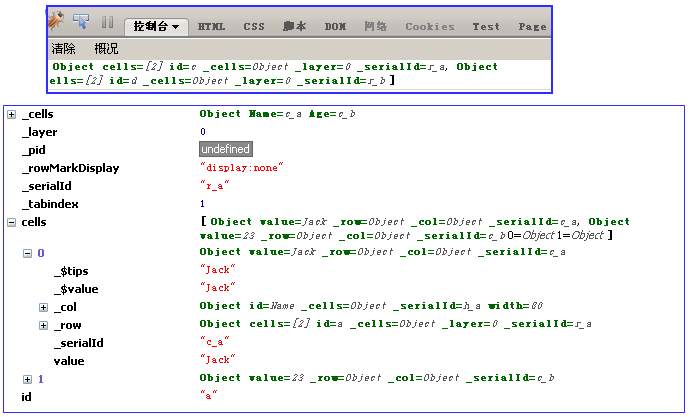
//console.log(tg.getRows());

Comparing these three formats

There is circular reference in memory data, can‘t be directly serialized



The rows memory data in firebug:



### Active Modes

There are three active modes for TreeGrid:

* non-active appearance : activeMode is “none”;
* the row-active appearance: activeMode is “row” ;
* the cell-active appearance: activeMode is “cell”;

#### non-active appearance

**Input:**

var block=new xui.UI.Block({width:200,height:200}).show();

var tg=new xui.UI.TreeGrid;

tg.setRowHandler(false)

.setHeader(["Name", "Age"])

.setRows([['Jack', 23], ['John', 32]])

.setActiveMode("none")

.show(block)

. afterRowActive (function(profile,row){

xui.message(row.id);

})

. afterCellActive (function(profile,cell){

xui.message(cell.value);

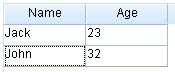
})

Sets to ‘none’

No row handler

Does not trigger events

**Output:**



#### row-active appearance

var block=new xui.UI.Block({width:200,height:200}).show();

var tg=new xui.UI.TreeGrid;

tg.setRowHandler(false)

.setHeader(["Name", "Age"])

.setRows([['Jack', 23], ['John', 32]])

.setActiveMode("row")

.show(block)

.afterRowActive (function(profile,row){

xui.message(row.id);

})

.afterCellActive (function(profile,cell){

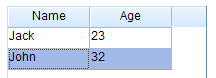
xui.message(cell.value);

})

Sets to “row”

No row handler

Will be fired



non-active appearence

#### cell-active appearance

var block=new xui.UI.Block({width:200,height:200}).show();

var tg=new xui.UI.TreeGrid;

tg.setRowHandler(false)

.setHeader(["Name", "Age"])

.setRows([['Jack', 23], ['John', 32]])

.setActiveMode("cell")

.show(block)

.afterRowActive (function(profile,row){

xui.message(row.id);

})

.afterCellActive (function(profile,cell){

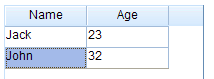
xui.message(cell.value);

})

Sets to “row”

No row handler

Will be fired



non-active appearance

### Selection Mode

There are five selection modes for TreeGrid:

* Non-selection: activeMode is “none”, or selMode is ‘none’
* Single row selection: activeMode is “row”, and selMode is ‘single’
* Multi-rows selection: activeMode is “row”, and selMode is ‘multi’
* Single cell selection: activeMode is “cell”, and selMode is ‘single’
* Multi-cells selection: activeMode is “cell”, and selMode is ‘multi’

#### Non-selection

**Input:**

var block=new xui.UI.Block({width:200,height:200}).show();

var tg=new xui.UI.TreeGrid;

tg.setRowHandler(false)

.setHeader(["Name", "Age"])

.setRows([['Jack', 23], ['John', 32]])

.setSelMode("none")

.show(block)

.afterUIValueSet(function(profile,ovalue,value){

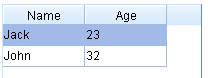
xui.message(value);

});

Non-selection

Won’t be fired

**Output:**



It’s active appearance, not the selection

**Input:**

var block=new xui.UI.Block({width:200,height:200}).show();

var tg=new xui.UI.TreeGrid;

tg.setRowHandler(false)

.setHeader(["Name", "Age"])

.setRows([['Jack', 23], ['John', 32]])

.setActiveMode("none")

.setSelMode("none")

.show(block)

.afterUIValueSet(function(profile,ovalue,value){

xui.message(value);

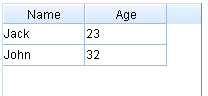
});

Non-selection

Won’t be fired

Non-active

**Output:**



Non-active appearance, non-selection

#### Single row selection

**Input:**

var block=new xui.UI.Block({width:200,height:200}).show();

var tg=new xui.UI.TreeGrid;

tg.setRowHandler(false)

.setHeader(["Name", "Age"])

.setRows([['Jack', 23], ['John', 32]])

.setSelMode("single")

.show(block)

.afterUIValueSet(function(profile,ovalue,value){

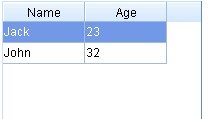
xui.message(value);

});

Sets to ‘single’ mode

Will be fired

**Output:**



Selection appearance

#### Multi-row selection

**Input:**

var block=new xui.UI.Block({width:200,height:200}).show();

var tg=new xui.UI.TreeGrid;

tg.setRowHandlerWidth(24)

.setHeader(["Name", "Age"])

.setRows([['Jack', 23], ['John', 32]])

.setSelMode("multi")

.show(block)

.afterUIValueSet(function(profile,ovalue,value){

xui.message(value);

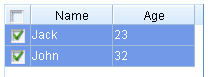
});

Sets to ‘multi’ mode

Row handler’s width

Will be fired

**Output:**



#### Single cell selection

**Input:**

var block=new xui.UI.Block({width:200,height:200}).show();

var tg=new xui.UI.TreeGrid;

tg.setRowHandler(false)

.setActiveMode("cell")

.setHeader(["Name", "Age"])

.setRows([['Jack', 23], ['John', 32]])

.setSelMode("single")

.show(block)

.afterUIValueSet(function(profile,ovalue,value){

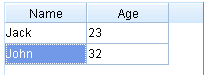
xui.message(value);

});

Sets to ‘single’ mode

Sets to ‘cell’ mode

**Output:**

****

#### Multi-cells selection

**Input:**

var block=new xui.UI.Block({width:200,height:200}).show();

var tg=new xui.UI.TreeGrid;

tg.setRowHandler(false)

.setActiveMode("cell")

.setHeader(["Name", "Age"])

.setRows([['Jack', 23], ['John', 32]])

.setSelMode("multi")

.show(block)

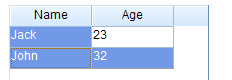
.afterUIValueSet(function(profile,ovalue,value){

xui.message(value);

});

Sets to ‘multi’ mode

**Output:**



### The Tree Grid

**Input:**

var block=new xui.UI.Block({width:200,height:200}).show();

var tg=new xui.UI.TreeGrid;

tg.setRowHandlerWidth(20)

.setHeader([

{id:"col1", caption:"Name"},

{id:"col2", caption:"Age", width:40}

]).setRows([

{id:"row1",cells:['Jack',23]},

{id:"row2",cells:['John',32],

sub:[{id:"row21",cells:['Tom',24]},

{id:"row22",cells:['Bob',25]}

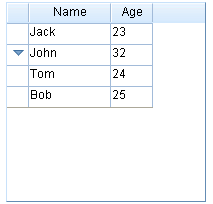
]}

]).show(block)

A row has sub rows

Row header is a must for tree grid

**Output:**



No Indentation here

**Input:**

var block=new xui.UI.Block({width:200,height:200}).show();

var tg=new xui.UI.TreeGrid;

tg.setRowHandlerWidth(20)

.setGridHandlerCaption("Name")

.setRowHandlerWidth(80)

.setHeader([

{id:"col2", caption:"Age", width:40}

]).setRows([

{id:"row1",caption: 'Jack',cells:[23]},

{id:"row2",caption: 'John',cells:[32],

sub:[{id:"row21",caption: 'Tom',cells:[24]},

{id:"row22", caption: 'Bob',cells:[25]}

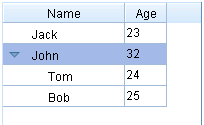
]}

]).show(block)

Row’s caption

Grid handler caption

Output:



Indentation in row handler

### Column config

#### The first column

In order to show the first column, you have to set rowHandler to [true].

**Input:**

var block=new xui.UI.Block({width:200,height:200}).show();

var tg=new xui.UI.TreeGrid;

tg.setRowHandlerWidth(80)

.setGridHandlerCaption("Name")

.setHeader([

{id:"col1", caption:"Age", width:40}

]).setRows([

{id:"row1",caption:'Jack',cells:[23]},

{id:"row2",caption:'John',cells:[32],

sub:[{id:"row21",caption:'Tom',cells:[24]},

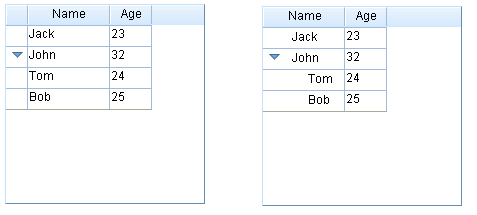
{id:"row22",caption:'Bob',cells:[25]}

]}

]).show(block)

Sets row handler’s width

**Output:**



The first column

The first column

上一节中缩进的例子

#### Column width

**Input:**

var block=new xui.UI.Block({width:240,height:200}).show();

var tg=new xui.UI.TreeGrid;

tg.setRowHandlerWidth(80)

.setGridHandlerCaption("Name")

.setHeader([

{id:"col1", caption:"Age", width:40},

{id:"col2", caption:"Part-time", width:90}

]).setRows([

{id:"row1",caption:'Jack',cells:[23, true]},

{id:"row2",caption:'John',cells:[32, false]}

]).show(block)

\_.asyRun(function(){

tg.updateHeader("col2", {width:70});

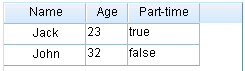
},1000);

Column’s width

The first column’s width

Modify column width dynamically

**Output:**



#### Drag&Drop to modify column width

“colResizer” property in TreeGrid determines whether the column width can be modified with Drag&Drop. Each column can include a “colResizer” property too. The “colResizer” property in column has higher priority than in TreeGrid.

**In CrossUI, "fine-grained Setting has higher priority than coarse-grained" is a base rule.**

**Input:**

下拉辑ckboxleizer(false)ion(profile,item){var block=new xui.UI.Block({width:240,height:200}).show();

var tg=new xui.UI.TreeGrid;

tg.setRowHandlerWidth(80)

.setColResizer(false)

.setGridHandlerCaption("Name")

.setHeader([

{id:"col1", caption:"Age", width:40},

{id:"col2", caption:"Part-time", width:90,colResizer:true}

]).setRows([

{id:"row1",caption:'Jack',cells:[23, true]},

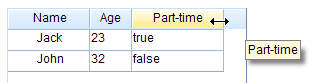
{id:"row2",caption:'John',cells:[32, false]}

]).show(block)

fine-grained setting

coarse-grained setting

**Output:**



Only this column

#### Drag&Drop to modify column position

**Input:**

var block=new xui.UI.Block({width:240,height:200}).show();

var tg=new xui.UI.TreeGrid;

tg.setRowHandlerWidth(80)

.setColMovable(false)

.setGridHandlerCaption("Name")

.setHeader([

{id:"col1", caption:"Age", width:40},

{id:"col2", caption:"Part-time", width:90,colMovable:true}

]).setRows([

{id:"row1",caption:'Jack',cells:[23, true]},

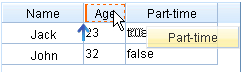
{id:"row2",caption:'John',cells:[32, false]}

]).show(block)

fine-grained setting

coarse-grained setting

**Output:**



#### Default Sorting

**Input:**

var block=new xui.UI.Block({width:240,height:200}).show();

var tg=new xui.UI.TreeGrid;

tg.setRowHandlerWidth(80)

.setColSortable(false)

.setGridHandlerCaption("Name")

.setHeader([

{id:"col1", caption:"Age", width:40},

{id:"col2", caption:"Part-time", width:90,colSortable:true}

]).setRows([

{id:"row1",caption:'Jack',cells:[23, true]},

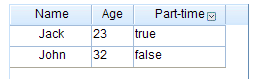
{id:"row2",caption:'John',cells:[32, false]}

]).show(block)

fine-grained setting

coarse-grained setting

**Output:**



Sorting icon

#### Custom Sorting

var block=new xui.UI.Block({width:240,height:200}).show();

var tg=new xui.UI.TreeGrid;

tg.setRowHandlerWidth(80)

.setGridHandlerCaption("Name")

.setHeader([

{id:"col1", caption:"Age", width:40},

{id:"col2", caption:"Part-time", width:90,sortby:function(x,y){return -1}}

]).setRows([

{id:"row1",caption:'Jack',cells:[23, true]},

{id:"row2",caption:'John',cells:[32, false]}

]).show(block)

Custom sorting function

#### Hide columns

**Input:**

var block=new xui.UI.Block({width:240,height:200}).show();

var tg=new xui.UI.TreeGrid;

tg.setRowHandlerWidth(80)

.setColHidable (false)

.setGridHandlerCaption("Name")

.setHeader([

{id:"col1", caption:"Age", width:40, colHidable:true },

{id:"col2", caption:"Part-time", width:90, colHidable:true}

]).setRows([

{id:"row1",caption:'Jack',cells:[23, true]},

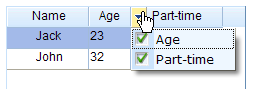
{id:"row2",caption:'John',cells:[32, false]}

]).show(block)

These 2 columns can be hidden

Global setting

**Output:**



#### Setting Cell Types in column header

**Input:**

var block=new xui.UI.Block({width:240,height:200}).show();

var tg=new xui.UI.TreeGrid;

tg.setRowHandlerWidth(80)

.setColSortable(false)

.setGridHandlerCaption("Name")

.setHeader([

{id:"col1", caption:"Age", width:40, type: "number"},

{id:"col2", caption:"Part-time", width:90, type: "checkbox"}

]).setRows([

{id:"row1",caption:'Jack',cells:[23, true]},

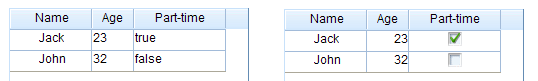
{id:"row2",caption:'John',cells:[32, false]}

]).show(block)

Checkbox type

Number type

**Output:**



#### column header style

**Input:**

var block=new xui.UI.Block({width:240,height:200}).show();

var tg=new xui.UI.TreeGrid;

tg.setRowHandlerWidth(80)

.setColSortable(false)

.setGridHandlerCaption("Name")

.setHeader([

{id:"col1", caption:"Age", width:40, type: "number"},

{id:"col2", caption:"Part-time", width:90, type: "checkbox", headerStyle: "font-weight:bold;"}

]).show(block)

Sets bold

**Output:**



#### column header icon

**Input:**

var block=new xui.UI.Block({width:240,height:200}).show();

var tg=new xui.UI.TreeGrid;

tg.setRowHandlerWidth(80)

.setColSortable(false)

.setGridHandlerCaption("Name")

.setHeader([

{id:"col1", caption:"Age", width:40, type: "number"},

{id:"col2", caption:"Part-time", width:90, type: "checkbox", renderer: function(h){

return "<img style='vertical-align:middle' src='img/a.gif'> "+ h.caption;

}}

]).show(block)

Renderer function

**Output:**



#### Update column header dynamically

var block=new xui.UI.Block({width:240,height:200}).show();

var tg=new xui.UI.TreeGrid;

tg.setGridHandlerCaption("Name")

.setHeader([

{id:"col1", caption:"Age", width:40, type: "number"},

{id:"col2", caption:"Part-time", width:90, type: "checkbox"}

]).setRows([

{id:"row1",caption:'Jack',cells:[23, true]},

{id:"row2",caption:'John',cells:[32, false]}

]).show(block)

\_.asyRun(function(){

tg.updateHeader('col2','Full-time')

},1000)

\_.asyRun(function(){

tg.updateHeader('col2',{caption:'Part-time', width:40, headerStyle:'font-weight:bold', colResizer:false, colSortable:false, colMovable:true, colHidable:true})

},2000)

Updates caption only

Those properties are updatable

### Row config

#### Row height

**Input:**

var block=new xui.UI.Block({width:240,height:200}).show();

var tg=new xui.UI.TreeGrid;

tg.setGridHandlerCaption("Name")

.setHeader([

{id:"col1", caption:"Age", width:40, type: "number"},

{id:"col2", caption:"Full-time", width:90, type: "checkbox"}

]).setRows([

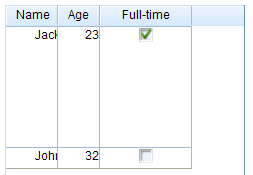
{id:"row1",caption:'Jack',cells:[23, true], height:120},

{id:"row2",caption:'John',cells:[32, false]}

]).show(block)

Sets row height

**Output:**



#### Drag&Drop to modify row height

**Input:**

下拉辑ckboxleizer(false)ion(profile,item){var block=new xui.UI.Block({width:240,height:200}).show();

var tg=new xui.UI.TreeGrid;

tg.setRowHandlerWidth(80)

.setRowResizer(false)

.setGridHandlerCaption("Name")

.setHeader([

{id:"col1", caption:"Age", width:40},

{id:"col2", caption:"Full-time", width:90}

]).setRows([

{id:"row1",caption:'Jack',cells:[23, true]},

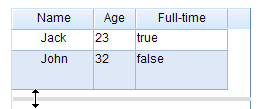
{id:"row2",caption:'John',cells:[32, false],rowResizer:true}

]).show(block)

This row has rowResizer

Global disabled rowResizer

**Output:**



#### Setting cell type in row

**Input:**

var block=new xui.UI.Block({width:240,height:200}).show();

var tg=new xui.UI.TreeGrid;

tg.setRowHandlerWidth(80)

.setColSortable(false)

.setGridHandlerCaption("Name")

.setHeader([

{id:"col1", caption:"Age", width:40, type: "number"},

{id:"col2", caption:"Part-time", width:90, type: "checkbox"}

]).setRows([

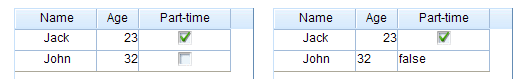
{id:"row1",caption:'Jack',cells:[23, true]},

{id:"row2",caption:'John',cells:[32, false],type: "label"}

]).show(block)

Sets all cells in this row type to ‘'label'

**Output:**



#### Row style

**Input:**

var block=new xui.UI.Block({width:240,height:200}).show();

var tg=new xui.UI.TreeGrid;

tg.setRowHandlerWidth(80)

.setGridHandlerCaption("Name")

.setHeader([

{id:"col1", caption:"Age", width:40, type: "number"},

{id:"col2", caption:"Full-time", width:90, type: "checkbox"}

]).setRows([

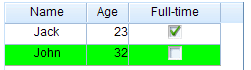
{id:"row1",caption:'Jack',cells:[23, true]},

{id:"row2",caption:'John',cells:[32, false],rowStyle: "background-color:#00ff00"}

]).show(block)

Sets styles

**Output:**



#### Row numbers

**Input:**

var block=new xui.UI.Block({width:240,height:200}).show();

var tg=new xui.UI.TreeGrid;

tg.setRowHandlerWidth(80)

.setRowNumbered(true)

.setGridHandlerCaption("Name")

.setHeader([

{id:"col1", caption:"Age", width:40, type: "number"},

{id:"col2", caption:"Full-time", width:90, type: "checkbox"e paires, width:90, type: "checkbox"}

]).setRows([

{id:"row1",caption:'Jack',cells:[23]},

{id:"row2",caption:'John',cells:[32],

sub:[{id:"row21",caption:'Tom',cells:[24]},

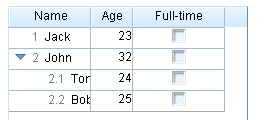
{id:"row22",caption:'Bob',cells:[25]}

]}

]).show(block)

To show row numbers

**Output:**



Default format line numbers

#### Custom row numbers

**Input:**

var block=new xui.UI.Block({width:240,height:200}).show();

var tg=new xui.UI.TreeGrid;

tg.setRowHandlerWidth(80)

.setRowNumbered(true)

.setGridHandlerCaption("姓名")

.setHeader([

{id:"col1", caption:"Age", width:40, type: "number"},

{id:"col2", caption:"Full-time", width:90, type: "checkbox"e paires, width:90, type: "checkbox"}

]).setRows([

{id:"row1",caption:'Jack',cells:[23]},

{id:"row2",caption:'John',cells:[32],

sub:[{id:"row21",caption:'Tom',cells:[24]},

{id:"row22",caption:'Bob',cells:[25]}

]}

])

.setCustomFunction('getNumberedStr',function(no){

var a=no.split('.');

a[0]={1:'I',2:'II'}[a[0]];

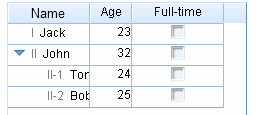
return a.join('-')

})

.show(block)

Custom function

Output:



#### Alternate Row Colors

**Input:**

var block=new xui.UI.Block({width:240,height:200}).show();

var tg=new xui.UI.TreeGrid;

tg.setRowHandlerWidth(80)

.setAltRowsBg (true)

.setGridHandlerCaption("Name")

.setHeader([

{id:"col1", caption:"Age", width:40, type: "number"},

{id:"col2", caption:"Full-time", width:90, type: "checkbox"}

]).setRows([

{id:"row1",caption:'Jack',cells:[23]},

{id:"row2",caption:'John',cells:[32],

sub:[{id:"row21",caption:'Tom',cells:[24]},

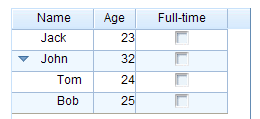
{id:"row22",caption:'Bob',cells:[25]}

]}

]).show(block)

Sets alternate bg color

**Output:**



#### Group

**Input:**

var block=new xui.UI.Block({width:240,height:200}).show();

var tg=new xui.UI.TreeGrid;

tg.setRowHandlerWidth(80)

.setGridHandlerCaption("Name")

.setHeader([

{id:"col1", caption:"Age", width:40, type: "number"},

{id:"col2", caption:"Full-time", width:90, type: "checkbox"}

]).setRows([

{id:"row1",caption:'Jack',cells:[23]},

{id:"row2",caption:'John',cells:[32],group:true,

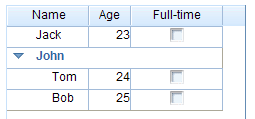
sub:[{id:"row21",caption:'Tom',cells:[24]},

{id:"row22",caption:'Bob',cells:[25]}

]}

]).show(block)

It’s a group row



Group row

#### Preview and Summary region

**Input:**

var block=new xui.UI.Block({width:240,height:200}).show();

var tg=new xui.UI.TreeGrid;

tg.setRowHandlerWidth(80)

.setGridHandlerCaption("Name")

.setHeader([

{id:"col1", caption:"Age", width:40, type: "number"},

{id:"col2", caption:"Full-time", width:90, type: "checkbox"}

]).setRows([

{id:"row1",caption:'Jack',cells:[23], preview: '<strong>Attention:</strong>',summary: '<em>Jack is athe right one</em>'} ,

{id:"row2",caption:'John',cells:[32], preview: 'John is OK',

sub:[{id:"row21",caption:'Tom',cells:[24]},

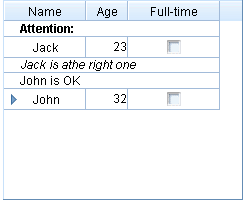
{id:"row22",caption:'Bob',cells:[25]}

]}

]).show(block)

preview

summary



summary

preview

#### Update row dynamically

**Input:**

var block=new xui.UI.Block({width:240,height:200}).show();

var tg=new xui.UI.TreeGrid;

tg.setRowHandlerWidth(80).setGridHandlerCaption("Name")

.setHeader([

{id:"col1", caption:"Age", width:40, type: "number"},{id:"col2", caption:"Full-time", width:90, type: "checkbox"}

]).setRows([

{id:"row1",caption:'Jack',cells:[23, true]},

{id:"row2",caption:'John',cells:[32],

sub:[{id:"row21",caption:'Tom',cells:[24]},

{id:"row22",caption:'Box',cells:[25]}

]}

]).show(block)

\_.asyRun(function(){

tg.updateRow('row2', 'Jerry')

},1000)

\_.asyRun(function(){

tg.updateRow('row2',{caption:'Group', height:30, rowStyle:'background-color:#00ff00;', rowResizer:false, group:true, preview:'preview', summary:'summary'})

},2000)

\_.asyRun(function(){

tg.updateRow('row1', {sub:[{value:"Kate",cells:[24,true]}]})

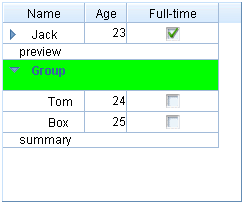
},3000)

Updates row caption only

These properties are updatable

Updates all sub rows

**Output:**



### Cell config

#### Cell types

These types are support:

* + ‘label’: readonly text;
  + ‘button’: the button;
  + ‘input’: single line input;
  + ‘textarea’: multi lines input;
  + ‘number’: number only input;
  + ‘currency’: currency only input;
  + ‘progress’: the progress appearance;
  + ‘combobox’: combo input;
  + ‘listbox’: readonly combo input;
  + ‘getter’: for getting data;
  + ‘helpinput’: help data input;
  + ‘cmdbox’: command box input;
  + ‘popbox’: pop box input;
  + ‘time’: time input;
  + ‘date’: date input;
  + ‘color’: color input;

**Input:**

var block=new xui.UI.Block({width:240,height:200}).show();

var tg=new xui.UI.TreeGrid;

tg.setRowHandlerWidth(80)

.setColSortable(false)

.setGridHandlerCaption("Name")

.setHeader([

{id:"col1", caption:"Age", width:40, type: "number"},

{id:"col2", caption:"Full-time", width:90, type: "label"}

]).setRows([

{id:"row1",caption:'Jack',cells:[23, true]},

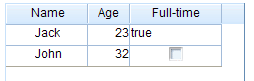
{id:"row2",caption:'John',cells:[32, {value:false, type: "checkbox"}] }

]).show(block)

Setting in cell (has priority)

Setting in column header data

**Output:**



#### Cell style

**Input:**

var block=new xui.UI.Block({width:240,height:200}).show();

var tg=new xui.UI.TreeGrid;

tg.setRowHandlerWidth(80)

.setColSortable(false)

.setGridHandlerCaption("Name")

.setHeader([

{id:"col1", caption:"Age", width:40, type: "number"},

{id:"col2", caption:"Full-time", width:90, type: "checkbox", cellStyle: "background-color:#00ff00;"}

]).setRows([

{id:"row1",caption:'Jack', cells:[23, true]},

{id:"row2",caption:'John', cellStyle: "background-color:#0000ff;",cells:[

32,

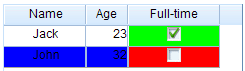
{value:false, cellStyle: "background-color:#ff0000;"}

] }

]).show(block)

**Output:**

Setting in row



Setting in column header

Setting in cell

#### Update cell dynamically

**Input:**

var block=new xui.UI.Block({width:240,height:200}).show();

var tg=new xui.UI.TreeGrid;

tg.setRowHandlerWidth(80).setGridHandlerCaption("Name")

.setHeader([

{id:"col1", caption:"Age", width:40, type: "number"},{id:"col2", caption:"Full-time", width:90, type: "checkbox"}

]).setRows([

{id:"row1",caption:'Jack',cells:[23, true]},

{id:"row2",caption:'John',cells:[32]}

]).show(block)

\_.asyRun(function(){

tg.updateCellByRowCol('row2','col1', 18)

},1000)

\_.asyRun(function(){

tg.updateCellByRowCol('row2','col1',{value:18,cellStyle:'background-color:#00ff00;'})

},2000)

\_.asyRun(function(){

tg.updateCellByRowCol ('row2','col1', {type:"listbox",value:"20", editorListItems:["20","30","40"], editable:true})

},3000)

Updates value only

These properties are updatable

Updates cell type

### Editable

“editable” property in TreeGrid determines whether the TreeGrid is editable or not . Each column / row / cell has this property too. Those setting follow “Fine-grained priority principle”.

* TreeGrid’s editable =>false; cell’s editable=>true: only this cell is editable
* TreeGrid’s editable =>false; column header’s editable=>true: only this column is editable
* TreeGrid’s editable =>false; row’s editable=>true: only this row is editable
* TreeGrid’s editable =>true; cell’s editable=>true: only this cell is uneditable
* TreeGrid’s editable =>true; column header’s editable=>false: only this column is uneditable
* TreeGrid’s editable =>true; row’s editable=> false: only this row is uneditable

It should be noted that, cells in Row handler are uneditalbe; cells with ‘label’ or ‘button’ type are uneditable.

#### Editable TreeGrid

**Input:**

var block=new xui.UI.Block({width:240,height:200}).show();

var tg=new xui.UI.TreeGrid;

tg.setRowHandler(false)

.setEditable(true)

.setHeader([

{id:"col1", caption:"Name", width:60, type: 'input'},

{id:"col2", caption:"Age", width:40, type: "number"},

{id:"col3", caption:"Gender", width:40, type: "listbox", editorListItems:[{id:'male',caption:'Male'},{id:'female', caption:'Female'}]}

]).setRows([

['Jack',23, {value:'male',caption:'Male'}],

['John',25, {value:'female',caption:'Female'}]

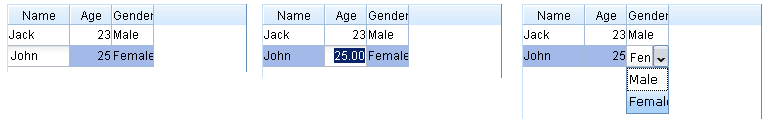
]).show(block)

Value and caption

List for editor

Sets editable

**Output:**



#### Editable column

**Input:**

var block=new xui.UI.Block({width:240,height:200}).show();

var tg=new xui.UI.TreeGrid;

tg.setRowHandler(false)

.setEditable(false)

.setHeader([

{id:"col1", caption:"Name", width:60, type: 'input'},

{id:"col2", caption:"Age", width:40, type: "number"},

{id:"col3", caption:"Gender", width:40, type: "listbox", editable:true, editorListItems:[{id:'male',caption:'Male'},{id:'female', caption:'Female'}]}

]).setRows([

['Jack',23, {value:'male',caption:'Male'}],

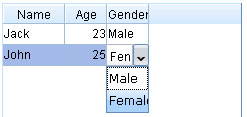
['John',25, {value:'female',caption:'Female'}]

]).show(block)

This column is editable

Sets uneditable

**Output:**

****

#### Editable row

**Input:**

var block=new xui.UI.Block({width:240,height:200}).show();

var tg=new xui.UI.TreeGrid;

tg.setRowHandler(false)

.setEditable(false)

.setHeader([

{id:"col1", caption:"Name", width:60, type: 'input'},

{id:"col2", caption:"Age", width:40, type: "number"},

{id:"col3", caption:"Gender", width:40, type: "listbox", editorListItems:[{id:'male',caption:'Male'},{id:'female', caption:'Female'}]}

]).setRows([

['Jack',23, {value:'male',caption:'Male'}],

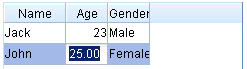
{cells:['John',25, {value:'female',caption:'Female'}],editable:true}

]).show(block)

This row is editable

Sets uneditable

**Output:**



#### Editable cell

**Input:**

var block=new xui.UI.Block({width:240,height:200}).show();

var tg=new xui.UI.TreeGrid;

tg.setRowHandler(false)

.setEditable(false)

.setHeader([

{id:"col1", caption:"Name", width:60, type: 'input'},

{id:"col2", caption:"Age", width:40, type: "number"},

{id:"col3", caption:"Gender", width:40, type: "listbox", editorListItems:[{id:'male', caption:'Male'}, {id:'female', caption:'Female'} ]}

]).setRows([

['Jack',23, {value:'male', caption:'Male'}],

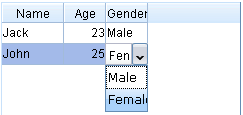
['John',25, {value:'female', caption:'Female', editable:true }]

]).show(block)

Only this cell is editable

Sets uneditable

**Output:**

****

#### The Editor

When a cell is set to editable, “active this cell” will show a corresponding editor. There are the following editors for different cell types.

* + ‘label’: readonly; no editor
  + ‘button’: readonly; no editor
  + ‘input’: normal xui.UI.Input control
  + ‘textarea’: multi lines xui.UI.Input control
  + ‘number’: number only xui.UI.Input control
  + ‘currency’: currency only xui.UI.Input control
  + ‘progress’: xui.UI.ComboInput control, spin
  + ‘combobox’: xui.UI.ComboInput control, combobox
  + ‘listbox’: xui.UI.ComboInput control, listbox
  + ‘getter’: xui.UI.ComboInput control, getter
  + ‘helpinput’: xui.UI.ComboInput control, helpinput
  + ‘cmdbox’: xui.UI.ComboInput control, cmdbox
  + ‘popbox’: xui.UI.ComboInput control, popbox
  + ‘time’: xui.UI.ComboInput control, time
  + ‘date’: xui.UI.ComboInput control, date
  + ‘color’: xui.UI.ComboInput control, color

**Input:**

var block=new xui.UI.Block({width:300,height:340}).show();

var tg=new xui.UI.TreeGrid;

tg.setRowHandler(false)

.setEditable(true)

.setHeader(["Type","Cell UI"]).setRows([

{cells:['label',{type:'label',value:'label'}]},

{cells:['button',{type:'button',value:'button'}]},

{cells:['input',{type:'input',value:'input'}]},

{cells:['textarea',{type:'textarea',value:'textarea'}]},

{cells:['number',{type:'number',value:'1.23'}]},

{cells:['currencty',{type:'number',value:'21.23'}]},

{cells:['progress',{type:'progress',value:'0.85'}]},

{cells:['combobox',{type:'combobox',value:'combobox'}]},

{cells:['listbox',{type:'listbox',value:'listbox'}]},

{cells:['getter',{type:'getter',value:'getter'}]},

{cells:['helpinput',{type:'helpinput',value:'helpinput'}]},

{cells:['cmdbox',{type:'cmdbox',value:'cmdbox'}]},

{cells:['popbox',{type:'popbox',value:'popbox'}]},

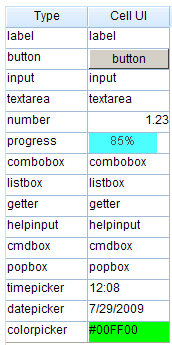
{cells:['time',{type:'time',value:'12:08'}]},

{cells:['date',{type:'date',value:(new Date).getTime()}]},

{cells:['color',{type:'color',value:'#00ff00'}]}

]).show(block)

**Output：**

****

#### Custom the editor

**Input:**

var block=new xui.UI.Block({width:300,height:340}).show();

var tg=new xui.UI.TreeGrid;

tg.setRowHandler(false)

.setEditable(true)

.setHeader(["Type","Cell UI "]).setRows([

{cells:['email',{type:'email',value:'a@b.com'}]},

{cells:['popwnd',{type:'popwnd',value:'value'}]}

])

.beforeIniEditor(function(profile, cell, cellNode){

var t=cell.type;

if(t=='email'){

var editor = new xui.UI.Input({valueFormat:"^[\\w\\.=-]+@[\\w\\.-]+\\.[\\w\\.-]{2,4}$"});

return editor;

}

if(t=='popwnd'){

var dlg=xui.prompt('Specify it','Update',cell.value, function(value){

if(cell.value!==value)

profile.boxing().updateCell(cell, value);

});

dlg.getRoot().cssPos(cellNode.offset());

return false;

}

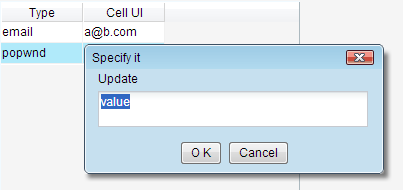
}).show(block);

Return false for advanced custom editor

Return the custom editor

xui.UI.Input or CombInput

**Output:**



The custom editor

### Add/Remove rows

**Input:**

var block=new xui.UI.Block({width:240,height:200}).show();

var tg=new xui.UI.TreeGrid;

tg.setRowHandler(false)

.setEditable(true)

.setHeader([

{id:"col1", caption:"Name", width:60, type: 'input'},

{id:"col2", caption:"Age", width:40, type: "number"}

]).setRows([

{id:'row1',cells:['Jack',23]},

{id:'row2',cells:['John',25]}

]).show(block);

\_.asyRun(function(){

tg.insertRows([[]])

},1000);

\_.asyRun(function(){

tg.insertRows(["Tom",30])

},2000);

\_.asyRun(function(){

tg.insertRows([{id:'row3',cells:['Jerry',19]},['Mark',31]])

},3000);

\_.asyRun(function(){

tg.removeRows('row1')

},4000);

\_.asyRun(function(){

tg.removeRows(['row2','row3'])

},5000);

\_.asyRun(function(){

tg.insertRows([{id:'row4',cells:['Jack',23]}],null,null,true)

},6000);

\_.asyRun(function(){

tg.insertRows([['John',23]],null,'row1',false)

},7000);

Adds a empty row

Adds a new row

Adds two rows

Removes a row by id

Removes two row by ids

Adds a row to the top

Adda a row next to ’row1’

**NOTE**

**chapter2/TGDynamic\index.html** is an overall example for ThreeGrid

**chapter2/TreeGrid.Paging\index.html** is another example for multi pages

## Other standard controls

### ProgressBar

**Input:**

xui.create('ProgressBar')

.setCaptionTpl("{value}% finished!")

.setValue(80)

.show();

Sets text display template

percentage

**Output:**



### Slider

**Input:**

xui.create('Slider')

.setPosition('relative')

.setSteps(100)

.setValue("20:50")

.show()

xui.create('Slider')

.setSteps(10)

.setType("vertical")

.setIsRange(false)

.setValue(2)

.setHeight(200)

.show();

Sets step to 100

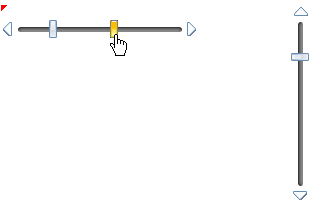
A single slider

**Sets the locations for the two slider**

Sets step to 10

Vertical slider

**Output:**



Drag it to move

Vertial slider

### Image

xui.create("Image")

.setSrc("img/a.gif")

.afterLoad(function(){

xui.message("The picture is loaded.");

})

.show();

xui.create("Image")

.setSrc("img/b.gif")

.beforeLoad(function(){

return false;

})

.show()

The image was loaded successful

Cancel loading process

### PageBar

**Input:**

var onclick=function(profile,page){

profile.boxing().setPage(page);

};

// a PageBar

xui.create('PageBar')

.setValue("1:5:12")

.onClick(onclick)

.show();

// another PageBar

xui.create('PageBar')

.setValue("1:5:12")

.setTop(100)

.setCaption("")

.setPrevMark("<<")

.setNextMark(">>")

.setTextTpl("[ \* ]")

.onClick(onclick)

.show();

Set current page

onClick event

1.Min page; 2.current page; 3.max page

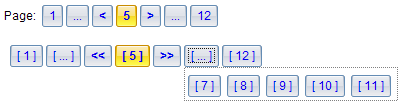
Caption label

Prev command label

Next command label

Page label templates, \* is the variable value

**Output:**



# Data exchanging(Ajax)

CrossUI is a client-side solution, it can work with any backend (php, .Net, Java, python) or static HTML pages. Client-side and backend is completely decoupled. Client-side does not need to care what kind of technique is used in the backend. Client-side sends request to, and gets response from a given backend service(e.g. JSON service, REST service) .

There are three IO class in CrossUI:

* xui.Ajax: An AJAX wrapper for xmlHttp object. It’s features:
  + Can only access the same domain by default;
  + Works both synchronous and asynchronous;
  + Works both ‘get’ and ‘post’ methods;
  + Returns string.
* xui.SAjax: An AJAX wrapper for “script tag”. It’s features:
  + Cross domain;
  + Asynchronous only;
  + Cannot post data;
  + Returned content is packaged as javascript’s Object

inb.SAjax send request data includes a “callback” parameter (default is “*xui.SAjax.NO.\_1*”).

**Server’s return data must be the following format:**

*xui.SAjax.NO.\_1* ({/\*JSON \*/})

* xui.IAjax: An AJAX wrapper for “iframe”. It’s features:
  + Cross domain;
  + Asynchronous only;
  + Can update file;
  + Works both ‘get’ and ‘post’ methods;
  + Returned content is packaged as javascript’s Object

inb.IAjax send request data includes a “callback” parameter (default is “window.name”).

**Server’s return data must be the following format:**

<script type='text' id='json'>{/\*JSON\*/}</script>

<script type='text/javascript'>

window.name=document.getElementById('json').innerHTML;

</script>

**“xui.request” function can choose an appropriate class from xui.Ajax, xui.SAjax or xui.IAjax automatically, according to requested domain, ‘GET/POST’ method and other information.**

**NOTE**

Examples in this chapter works only as a http url, do not double-click directly to open.

## Fiddler

In order to understand the data exchanges process better, you need a tool like Fiddler to monitor network traffic.

Go to <http://www.fiddler2.com/fiddler2/> to get Fiddler.

Fiddler can configure IE proxy automatically, but if you are in Firefox, chrome or opera, you need to configure the proxy by manual (Fiddler proxy: 127.0.0.1:8888). Of course, you can find some Firefox proxy plug-ins to help you.

## To get the contents of the file

xui.Ajax can get file contents easily.

xui.Ajax('data/ajax.js', '23233', function(rsp){

xui.message(rsp)

},function(errMsg){

xui.alert(errMsg)

}).start();

Start to send request

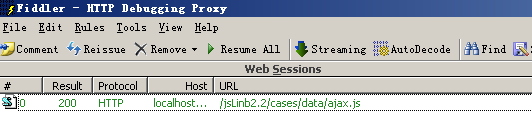
Callback function for fail

Callback function for success

File’s uri address

A random string for avoiding browser cache

**In Fiddler:**



## Synchronous data exchange

Only xui.Ajax support synchronous data exchanging.

var url="chapter3/request.php";

xui.Ajax(url, {

key:'test',

para:{p1:'para 1'}

},function(rsp){

xui.log(rsp);

},function(errMsg){

xui.alert(errMsg)

}, null, {

asy:false

}).start();

synchronous

Request data

**In fiddler:**

**The request:**



**The response:**



This is an asynchronous request:

var url="chapter3/request.php";

xui.Ajax(url, {

key:'test',

para:{p1:'para 1'}

},function(rsp){

xui.log(rsp);

},function(errMsg){

xui.alert(errMsg)

}).start();

## Cross-domain

xui.SAjax and xui.IAjax can be used for calling Cross Domain Web Services. But only xui.IAjax can post data and upload file.

### To monitor SAjax

**Code:**

var url="chapter3/request.php";

xui.SAjax(url, {

key:'test',

para:{p1:'para 1'}

},function(rsp){

xui.log(rsp);

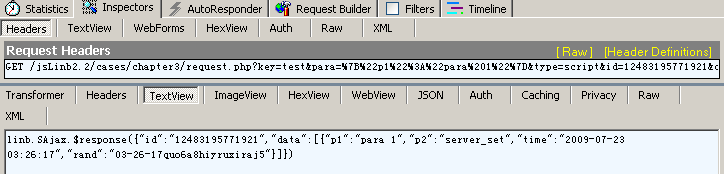
},function(errMsg){

xui.alert(errMsg)

}).start();

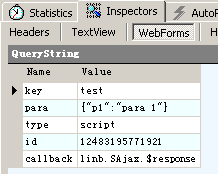
Request data

**In Fiddler:**



The response

GET method



‘script’ tells backend: this is a SAjax request

The request data

### To monitor IAjax

**Code:**

var url="chapter3/request.php";

xui.IAjax(url, {

key:'test',

para:{p1:'para 1'}

},function(rsp){

xui.log(rsp);

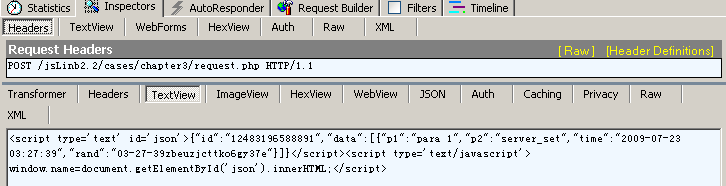
},function(errMsg){

xui.alert(errMsg)

}).start();

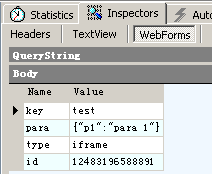
Request data

**In Fiddler:**



The response

POST method



‘iframe’ tells backend: this is an IAjax request

The request data

By default, IAajax use “POST” method, you can specify method in options.

var url="chapter3/request.php";

xui.IAjax(url, {

key:'test',

para:{p1:'para 1'}

},function(rsp){

xui.log(rsp);

},function(errMsg){

xui.alert(errMsg)

},null,{

method: 'get'

}).start();

Switch to GET method

## File Upload

Only xui.UI.IAjax can upload file.

This code in this section is in "chapter3/upload/".

### Selecting upload file with ComboInput

ComboInput

Sets ComboInput’s type property to “file”:



**chapter3/upload/index.html**

### Upload by IAjax

Class('App', 'xui.Module',{

    Instance:{

iniComponents:function(){

// [[code created by CrossUI UI Builder

var host=this, children=[], append=function(child){children.push(child.get(0))};

append((new xui.UI.SLabel)

.setHost(host,"slabel1")

.setLeft(40)

.setTop(44)

.setCaption("Select your file: ")

);

append((new xui.UI.ComboInput)

. setHost(host,"upload")

.setLeft(140)

.setTop(40)

.setWidth(140)

.setReadonly(true)

.setType("upload")

.setValue("Select a file ...")

);

append((new xui.UI.SButton)

. setHost(host,"sbutton3")

.setLeft(290)

.setTop(40)

.setCaption("Upload it")

.onClick("\_sbutton3\_onclick")

);

return children;

// ]]code created by CrossUI UI Builder

},

\_sbutton3\_onclick:function (profile, e, src, value) {

var file=this.upload.getUploadObj();

if(file){

xui.IAjax('../request.php',{key:'upload',para:{},file:file},function(rsp){

xui.alert(rsp.data.message);

},function(errMsg){

xui.alert(errMsg)

}).start();

}

}

    }

});

Created by Designer

Upload control

IAjax upload

Successful return

Getting file content

## A request wrapper for real application

In practical applications, you can choose xui.Ajax, xui.SAjax and xui.IAjax according to the actual situation. Usually, we will wrap a common function or class to handle all data interaction with the backend service. This is an example wrapper. Just for your reference.

request=function(service,

requestData,

onOK,

onStart,

onEnd,

file

){

\_.tryF(onStart);

xui.observableRun(function(threadid){

var options;

if(file){

requestData.file=file;

options={method:'post'};

}

xui.request(service, requestData, function(rsp){

if(rsp){

if(!rsp.error)

\_.tryF(onOK, [rsp]);

else

xui.pop(\_.serialize(rsp.error));

}else{

xui.pop(\_.serialize(rsp));

}

\_.tryF(onEnd);

},function(rsp){

xui.pop(\_.serialize(rsp));

\_.tryF(onEnd);

}, threadid,options)

});

};

Service url address

Request data (key/value pairs)

Callback for successful call

Callback for onStart and onEnd

File to upload

Success

Fail

Fail

Fail

## XML Data

If the server returns xml data, we can use xui.XML to convert the XML data into JSON data.

xui.Ajax('data/ajax.xml', '', function(rsp){

alert (rsp)

var obj = xui.XML.xml2json(xui.XML.parseXML(rsp));

xui.pop(obj.message);

},function(errMsg){

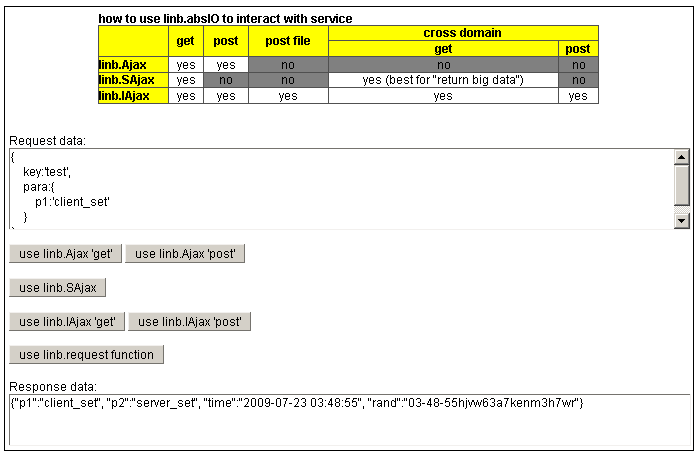
xui.alert(errMsg)

}).start();

XML to JSON

## An overall example

The following is an overall example for data exchanging.



xui.request

Ajax get and post

IAjax get and post

SAjax

Results window

Request data

Chapter3/io/index.html

Notice:

You can download PHP/C#/Java/Node.js backend demo code packages from:

<http://www.crossui.com/download.html> .

# Distributed UI

Sometimes, especially in larger applications, we maybe save a large “not frequently used” UI Class into a separate file. This file will not be loaded at the beginning.

When the application needs to show the UI, the program will automatically load code from the "separate file". It is so called "distributed UI". This "distributed UI" file can be in your server, or in different domain remote servers.

## Shows dialog from a remote file

There’s a file “Module3.js” in folder “chapter4\distributed\App\js\”, “Module3.js” includes a Class named “App.Module3”. Let’s try to call it.

**Input:**

Namespace("App");

xui.include("App.Module3",

xui.getPath("chapter4/distributed/App/js/","Module3.js"),

function(){

var ins=new App.Module3();

ins.show();

},function(){

xui.alert("fail");

}

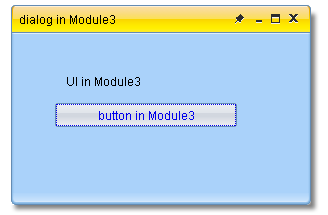
);

Namespace is “App”

Dynamic asynchronous remote file loading

Create instance and show it

**Output:**



Code in Module3.js

And try to load code and create UI from a difference domain.

Namespace("App");

xui.include("App.Module3",

"",

function(){

var ins=new App.Module3();

ins.show();

},function(){

xui.alert("fail");

}

);

A difference domain

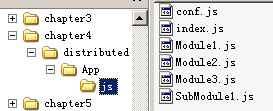
## xui.Module and xui.ModuleFactory

In fact, most of the actual business applications will not load code from a foreign domain. From another perspective, most of "Distributed UI" files are put in the application directory.

In this case, we can use xui.Module and xui.ModuleFactory to load those “distributed UI”. In order to use this approach, all those Classes must be derived from the xui.Module, named according to specified rules, and put into the specified directory.

xui.ModuleFactory implements a management mechanism for the xui.Module. It can follow a specified rule (finding file path from the class name) to load code from a remote file.

There’s an overall example in “chapter4/distributed”, we can browse it for detail.



Distributed UI classes

This folder

Conf file

### xui.ModuleFactory config

In conf.js:

CONF={ComFactoryProfile:

{

module1:{

cls:'App.Module1',

children:{

tag\_SubModule1:'submodule1'

}

},

submodule1:{

cls:'App.SubModule1'

}

}}

module1 is an “Aoo.Module1” Class

submodule1 is a “Aoo.SubModule1” Class

module1 has a child submodule1

Loading this configuration to xui.ModuleFactory:

xui.ModuleFactory.setProfile(CONF.ComFactoryProfile);

### xui.Module.Load

In file index.html,

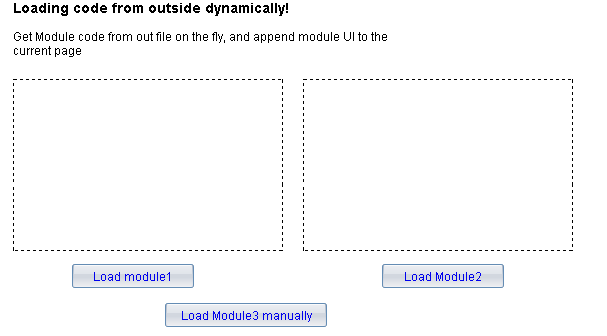
xui.Module.load ('App');

To load and show the firs UI Class

The above code will try to find file named “index.js” from “distributed/App/js/”, create an instance (new App), and show the instance to DOM.

xui.load <=> xui.Module.load

**Output:**



### newCom and getCom

**In index.js, onClick event for “Load module3 manually” button is:**

\_button3\_onclick:function (profile, e, value) {

var host=this;

xui.ModuleFactory.newCom('App.Module3' ,function(){

this.show(xui([document.body]));

});

}

newCom

Shows to HTML body

Callback function

Uses Class Name

[xui.ModuleFactory.newCom(“App.Module3”.. ], will:

* find file “Module3.js” in “distributed/App/js/”
* load code from file “Module3.js” ;
* create new instance,;
* call the callback function.

xui.newCom <=> xui.ModuleFactory.newCom

xui.getCom <=> xui.ModuleFactory.getCom

Note: newCom use “Class Name” to load code.

**onClick event for “Load module1” button is:**

\_button9\_onclick:function (profile, e, value) {

var host=this;

xui.ModuleFactory.getCom('module1',function(){

var ns=this;

host.div16.append(ns.getUIComponents(),false);

});

}

getCom

Shows to the left block

Callback function

From config

[xui.ModuleFactory.newCom(“module1”.. ], will:

* find config from xui.ModuleFactory
* find file “Module1.js” in “distributed/App/js/”
* load code from file “Module1.js” ;
* create new instance,;
* call the callback function.

**onClick event for “Load module2” button is:**

\_button10\_onclick:function (profile, e, value) {

var host=this;

xui.ModuleFactory.getCom('App.Module2',function(){

var ns=this;

host.div16.append(ns.getUIComponents(),false);

});

}

getCom

Shows to the right block

Callback function

“Class Name” works too.

No config needed in this case.

By default, the instance created by "getCom" is singleton, and will be cached in inb.ComFactory.

### xui.UI.Tag

There’s a xui.UI.Tag object in file Module1.js:

host.panelMain.append((new xui.UI.Tag)

.host(host,"tag2")

.setLeft(20)

.setTop(70)

.setWidth(218)

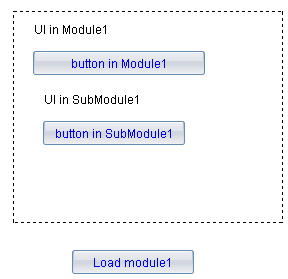
.setHeight(98)

.setTagKey("tag\_SubModule1")

);

Module name in configuration

Here, this Tag object configures size and position properties for module “tag\_SubModule1”. When the instance of Module1 was created, according to the Tag object’ info, system will load the “tag\_SubModule1” automatically, and set size and position properties to it. Then, system will replace the Tag object with “tag\_SubModule1” object, and destroy the Tag object.



The Tag object was replaced by “tag\_SubModule1”

### Destroy com

Call com’s **destroy()** function to destroy the Class instance;

Call **Class.destroy(“class name”)** to destroy the Class itself.

If you used “getCom(‘module name’)” to create an com instance, you have to call “**xui.ModuleFactory.setCom ( ‘module name’, null )**” to clear that cache.

### If com exists in memory

If a com exists in memory already, we can call it directly:

xui('body').append(new App.Acom);

# Some fundamental things

## Pop-up window

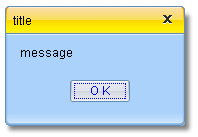
### alert window

xui.alert('title','message',function(){

xui.message('You close this window!')

}, 'O K', 50, 100);

Fired after user close the window

a

### confirm window

xui.confirm('title','confirm?',function(){

xui.message('You confirmed it')

},

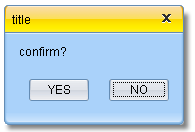
function(type){

xui.message(" You didn’t cofirm it -" + type)

},'YES', 'NO',50,100);

Fired when user click “YES”

Fired when user click “NO” or click close button



### prompt window

xui.prompt('title', 'message','default content',function(msg){

xui.message('You input - ' + msg)

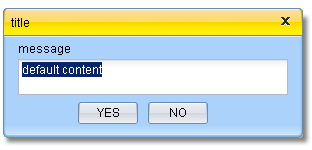
},function(){

xui.message(" You cancel it")

},'YES', 'NO',50,100);

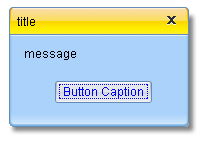
Fired when user click “YES”

Fired when user click “NO” or click close button



### pop window

xui.pop('title', 'message','Button Caption',50,100);



## Asynchronous execution

### asyRun

\_.asyRun is a wrapper for javascript setTimeout.

\_.asyRun(function(arg1,arg2){

xui.pop("this===xui:"+(this===xui), arg1+":"+arg2)

},

500,

['arg1','arg2'],

xui)

scope

parameters

Delay 500 ms

Function body

### resetRun

\_.asyRun is a wrapper for set timeout too. But it has an unique id. When you set another function with the same id, the latter will cover the former.

\_.resetRun('key1',function(arg){

xui.pop("this===xui:"+(this===xui), arg)

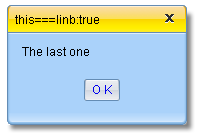
},500,['The previous one'],xui)

\_.resetRun('key1',function(arg){

xui.pop("this===xui:"+(this===xui), arg)

},500,['The last one'],xui)

The latter one will be executed



## Skin switcher

### Switch skin for whole application

There are three default system skins in CrossUI : default, vista and aqua. You can use xui.setTheme to switch the skin.

xui.setTheme("vista")

Dynamically (no page refresh)

### Change skin for a single control

It’s a fine-grained mechanism.

ctl\_button.setTheme("custom")

Only for “ctl\_button”

In this case, developer needs to define CSS class for this "custom".

## Language switcher

xui.alert(xui.getLang())

xui.setLang("cn");

Sets locale, switch locale dynamically (no page refresh)

Gets the current locale key

**Example “chapter5\lang” loading process:**

After DOM is ready, CrossUI framework will handle UI Class 🡪 App

Open url in the Browser

Loading html file (<1k)

**Browser load “loading.gif”(small)，and show it (Users will see the loading picture)**

Browser load xui-all.js simultaneously

[xui.Module.load] load code from App/js/index.js file asynchronously

**Loading lib locale file [CrossUI path] / xui/ Locale / en.js** asynchronously

If Class code is bigger, it’ll take more time. In the meantime, users will see the waiting message (loading.gif picture or other custom showing)

[xui.Module.load] new an instance of Class App, and render it into DOM

File 1

File 2

**Loading application locale file Locale/en.js** asynchronously

File 3

## DOM Manipulation

Class “xui.Dom” is a wrapper for cross-browser DOM Manipulation. It can: create / remove elements; manage elements’ attributes; manage elements’ CSS; manage elements’ events.

### Node generation and insertion

**Input:**

var firstNode;

xui('body').append(firstNode=xui.create('<div style="border:solid 1px">the first div</div>'));

firstNode.append('input')

.last().attr('value','append')

.parent()

.prepend('<button>prepend</button>')

.addPrev('<div style="border:solid 1px">prev div</div>')

.addNext('<div style="border:solid 1px"> div</div>')

Appends to firstNode

It’s the fist node

Returns to firstNode

Prepend a button

Adds to previous sibling

Adds to next sibling

Finds input, and sets

**output:**

append

addNext

addPrev

prepend



### Attributes and CSS

**Input:**

var node;

xui('body').append(node=xui.create('div'));

node.html('content<input value="ini">');

\_.asyRun(function(){

node.css('border','solid 1px');

},1000);

\_.asyRun(function(){

xui.message(node.css('fontSize'))

node.css({background:'#00ff00',fontSize:'16px'});

},2000);

\_.asyRun(function(){

node.attr('style','border:none;font-size:18px;')

},3000);

\_.asyRun(function(){

xui.message(node.last().attr('value'))

node.last().attr('value','updated');

},4000);

Sets contents

Updated input’s value attr

Gets CSS fontSize

Updates all style

Updates fontSize and backgorund

Updates CSS border

Gets input’ value attr

### className

There are five function to handle CSS className:

* hasClass: Determines whether a specified class exists or not
* addClass: Adds classes to the current DOM nodes
* removeClass: Removes classes from the current DOM nodes
* replaceClass: Replaces classes for the current DOM nodes
* tagClass: Adds/Removes a tag to all classes of the current DOM nodes

var node;

xui('body').append(node=xui.create('div'));

\_.asyRun(function(){

node.addClass("cls1 cls2 cls3");

xui.message(node.hasClass('cls2'));

node.text(node.attr('className'));

},1000);

\_.asyRun(function(){

node.removeClass("cls2");

node.text(node.attr('className'));

},2000);

\_.asyRun(function(){

node.replaceClass(/cls/g,"class");

node.text(node.attr('className'));

},3000);

\_.asyRun(function(){

node.tagClass("-mouseover",true);

node.text(node.attr('className'));

},4000);

\_.asyRun(function(){

node.tagClass("-mouseover",false);

node.text(node.attr('className'));

},6000);

Adds classes

Remove tag

Removes

Modifies

Determines whether a class name exists or not

Adds tag

### Dom events

There are three groups of event functions are designed for a DOM event in CrossUI: [before-], [on-] and [after-].

* xui(/\*\*/).onClick([function], ‘label’) => adds the [function] to [onclick]group;
* xui(/\*\*/).onClick([function]) => removes all event functions in [onclick] group, and adds the [function] to [onclick] group;
* xui(/\*\*/).onClick(null, ‘label’) => removes the event function labeled with ‘label’ from the [onclick] group;
* xui(/\*\*/).onClick(null) => removes all event functions in [onclick] group;
* xui(/\*\*/).onClick(null,null,true) => removes all event functions in [beforeclick] group, [onclick] group and [afterclick] group;
* xui(/\*\*/).onClick() => fire event, executes all event functions in [onclick] group in order. If any of those functions returns [false], the remaining functions will be ignored;
* xui(/\*\*/).onClick(true) => fire event, executes all event functions in [beforeclick] group, [onclick] group and [afterclick] group in order;

var node;

xui('body').append(node=xui.create("<button>click me</button>"));

node.onClick(function(){

alert('onClick');

return false;

})

.beforeClick(function(){

alert('beforeClick');

})

.afterClick(function(){

alert('afterClick');

});

node.onClick(true);

\_.asyRun(function(){

node.onClick(null);

node.onClick(true);

},2000);

Adds a onClick event

Fires all click events. Since onClick returns false, afterClick will not be fired.

Removes onClick event;

Adds a beforeClick event

Adds an afterClick event

Fires all click events. Since onClick was removed, afterClick will be fired this time.

### Node Drag&Drop

**Input:**

var btn,div;

xui('body').append(btn=xui.create("<button>drag me</button>"))

.append(div=xui.create("<div style='position:absolute;left:100px;top:100px;border:solid 1px;width:150px; height:150px;display:block;'> drop here </button>"))

btn.draggable(true,{dragType:'icon'},'dragkey','dragdata');

div.droppable(true,'dragkey')

.onDrop(function(){

xui.alert(xui.DragDrop.getProfile().dragData);

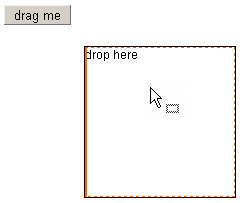
});

Sets draggable

Sets droppable

onDrop event

Output:

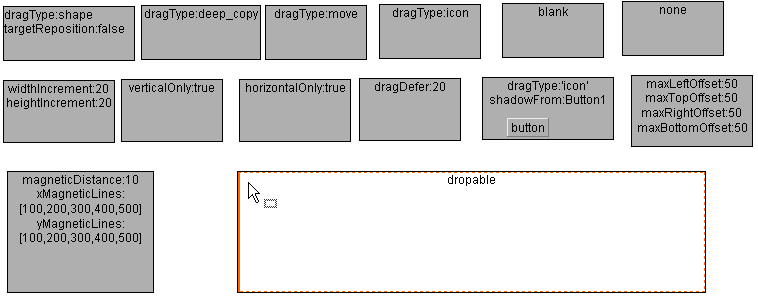


#### Drag&Drop profile

The “draggable” function’s second parameter is Drag&Drop profile object. It’s a key/value pairs. In dragging process, the Drag&Drop profile object can be got by xui.DragDrop.getProfile(). The profile object:

* dragType: ‘move’,’copy’,’deep\_copy’,’shape’,’icon’, ‘blank’ or ‘none’, Default is ‘shape’;
* shadowFrom: DOM element or xui.Dom Object. It’s valid when dragType==‘icon’;
* targetReposition: Boolean, does dd reset the target position, Default is [true];
* dragIcon: String, the drag icon image path, Default is [xui.ini.path+’ondrag.gif’];
* magneticDistance: Number, the magnetic distance, Default is 0;
* xMagneticLines: Array of Number, the magnetic line values in horizontal dir, Default is [];
* yMagneticLines: Array of Number, the magnetic line values in vertical dir, Default is [];
* widthIncrement: Number, the width increment in horizontal dir, Default is 0;
* heightIncrement: Number, the height increment in vertical dir, Default is 0;
* dragDefer: Number, when [xui.DragDrop.startDrag] is called, the real drag action will be triggered after [document.onmousemove] runs [dragDefer] times, Default is 0;
* horizontalOnly:Boolean, drag horizontal dir only, Default is [false];
* verticalOnly: Boolean, drag vertical dir only, Default is [false];
* maxBottomOffset</strong>:Number, the offset between [the restricted bottom] and [the current mouse Y], for mouse restricted region, Default is [null];
* maxLeftOffset</strong>:Number, the offset between [the restricted left] and [the current mouse X], for mouse restricted region, Default is [null];
* maxRightOffset</strong>:Number, the offset between [the restricted right] and [the current mouse X], for mouse restricted region, Default is [null];
* maxTopOffset: Number, the offset between [the restricted top] and [the current mouse Y], for mouse restricted region, Default is [null];
* targetNode: DOM element or xui.Dom Object, the drag target node;
* targetCSS: Number, the drag target node’s CSS key/value Object, Default is [null];
* dragKey: String, the drag key, Default is [null];
* dragData: Object, the drag data, Default is [null];
* targetLeft: Number, the drag target node’s CSS left, Default is [null];
* targetTop: Number, the drag target node’s CSS top, Default is [null];
* targetWidth: Number, the drag target node’s CSS width, Default is [null];
* targetHeight: Number, the drag target node’s CSS height, Default is [null];
* targetOffsetParent: xui.Dom Object, the drag target node offsetParent node, Default is [null];
* dragCursor: ‘none’, ‘move’, ‘link’, or ‘add’, the drag cursor key; [readonly]
* x: Number, current X value of mouse; [readonly]
* y: Number, current Y value of mouse; [readonly]
* ox: Number, original X value of mouse; [readonly]
* oy: Number, original Y value of mouse; [readonly]
* curPos: {left:Number,top:Number}, current CSS pos of the dragging node [readonly]
* offset: {x:Number,y:Number}, offset from now to origin [readonly]
* isWorking: Boolean, is dd working or not? [readonly]
* restrictedLeft: Number, the calculated restricted left value; [readonly]
* restrictedRight: Number, the calculated restricted right value; [readonly]
* restrictedTop: Number, the calculated restricted top value; [readonly]
* restrictedBottom: Number, the calculated restricted bottom value; [readonly]
* proxyNode: xui.Dom Object, the proxy Object; [readonly]
* dropElement: String, the target drop element DOM id. [readonly]

There is an DD overall example in **chapter3/dd/ddProfile.html.**

****

#### Events in Drag&Drop

For that node in dragging,

* onDragbegin
* onDrag
* onDragstop

For that droppable node,

* onDragenter
* onDragleave
* onDragover
* onDrop

**Input:**

var btn,div,elist;

xui('body').append(btn=xui.create("<button>drag me</button>"))

.append(div=xui.create("<div style='border:solid 1px;width:100px;height:100px;'>drop here</button>"))

xui('body').append(elist=xui.create('<div style="position:absolute;left:140px;top:40px;width:600px;height:400px;overflow:auto;"></div>'))

btn.dragable(true,{dragType:'icon'},'dragkey','dragdata')

.onDragbegin(function(){

elist.append('<strong>onDragbegin </strong>');

})

.onDrag(function(){

elist.append('<em>onDrag </em>');

})

.onDragstop(function(){

elist.append('<strong>onDragend </strong>');

})

div.dropable(true,'dragkey')

.onDragenter(function(){

elist.append('<strong>onDragenter </strong>');

})

.onDragover(function(){

elist.append('<em>onDragover </em>');

})

.onDragleave(function(){

elist.append('<strong>onDragleave </strong>');

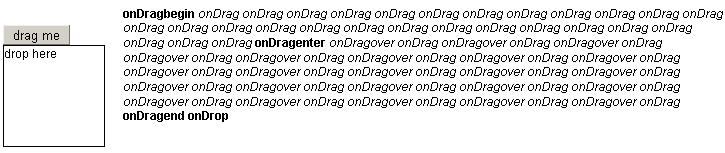
})

.onDrop(function(){

elist.append('<strong>onDrop </strong>');

});

**Output:**



## xui.Template

Xui.Template is a completely independent UI wrapper. It doesn’t depend on xui.UI Class and all its derived Classes.

### example 1

xui.Template includes three aspects: template, properties and events:

**Input:**

var tpl=new xui.Template;

tpl.setTemplate("<div [event]>{con}</div>")

//tpl.setTemplate({root:"<div [event]>{con}</div>"})

.setProperties({

con:'click me'

})

.setEvents({

root:{

onClick:function(){

xui.alert('Hi');

}

}

})

.show()

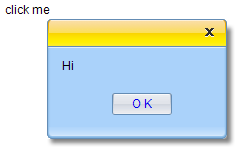
Sets template

Sets properties

Sets events

For event

Output:



### example 2

**Input:**

(tpl=new xui.Template)

.setTemplate({

root : "<div style='width:200px;border:solid 1px;'><h3>{head}</h3><ul>{items}</ul><div style='clear:both;'></div></div>",

items: "<li [event] style='padding:4px;border-top:dashed 1px;'><div ><div><a href='{href}'><img src='{src}'/><div>{price}</div></a></div></div><div><a href='{href}'><h4>{title}</h4><div>{desc}</div></a></div></li>"

})

.setEvents({

items:{

onMouseover:function(profile,e,src){

xui.use(src).css('backgroundColor','#EEE');

//Tips

var item=profile.getItem(src),

tpl=new xui.Template({"root":"<div style='text-align:center;border:solid 1px;background:#fff;'><h4>{title}</h4></div><div>{desc}</div>"},item),

html=tpl.toHtml();

xui.Tips.show(xui.Event.getPos(e),html);

},

onMouseout:function(profile,e,src){

xui(src).css('backgroundColor','transparent');

xui.Tips.hide();

}

}

})

.setProperties({

head:"On sale products",

items:[{ id:"a", href:"#", price:"$ 18.99", title:"product #0", desc:"product #0 is on sale now!" },

{id:"b", href:"#", price:"$ 23.99", title:"product #1", desc:"product #1 is on sale now!" },

{id:"c", href:"#", price:"$ 23.99", title:"product #2", desc:"product #2 is on sale now!" }]

})

.show()

Sets events in “items”

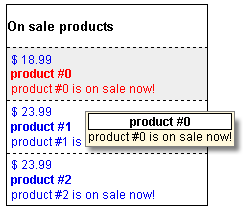
Sub template exists

Here, “root” key is a must

Sets properties in “root”

Sets properties in “items”

**Output:**



### A SButton based on xui.Template

“chapter5\SButton” is an example for creating a xui.UI.SButton like control based on xui.Template.

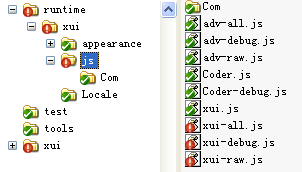
**Output:**



## About Debugging

### The code package for debugging

In folder “runtime/xui/js/”, All files ending with ”-debug.js” are for debugging purpose.



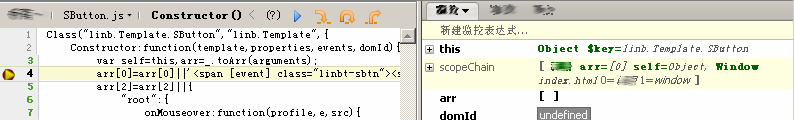
For debugging

For debugging

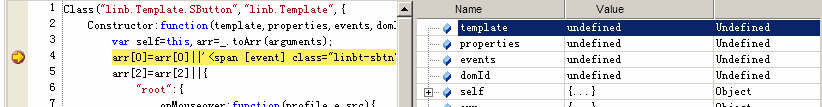
### Debugging Tools

You can use Firebug in Firefox, developer tool in IE8, chrome or opera10 to debug JavaScript.

**FireBug:**

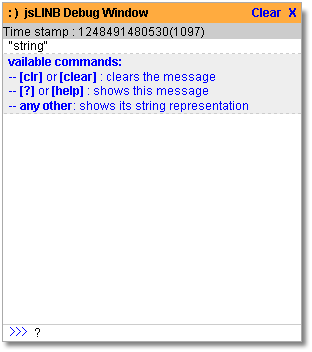


**Developer Tools in IE8:**



### Monitor Tools

CrossUI has a variable monitor tools. You can call xui.log(“xxx”) to show the monitor window:



# Some typical issues

## Layout

### Docking

**Input:**

xui.create('Block',{dock:"top",

height:80,html:'top'

}).show();

xui.create('Block',{dock:"bottom",

height:30,html:'bottom'

}).show();

xui.create('Block',{dock:"left",

width:150,html:'left'

}).show();

xui.create('Block',{dock:"right",

width:150,html:'right'

}).show();

xui.create('Block',{dock:"fill",

html:'main',

dockMargin:{left:10,right:10,top:10,bottom:10}

}).show();

The main area

At top

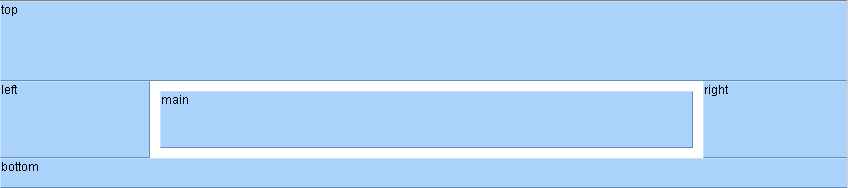
At bottom

Left side

Right side

Sets docking margin

Output:



### xui.UI.Layout

**Input:**

var layout1=xui.create('Layout',{type:'vertical',

items:[{

pos:'before',

id:'top',

size:80

},{

pos:'after',

id:'bottom',

size:30

}]

}).show();

xui.create('Layout',{type:'horizontal',

items:[{

pos:'before',

id:'top',

size:150

},{

pos:'after',

id:'bottom',

size:150

}]

}).show(layout1);

Vertial layout

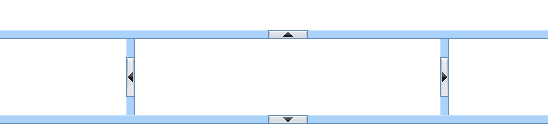
At top

At bottom

Left side

Right side

Horizontal layout



Drag&Drop to resize

### Relative Layout

**Input:**

xui.create('Pane',{position:'relative',

width:"auto",height:80,html:"the top div"

})

.setCustomStyle({"KEY":"border:solid 1px #888"})

.show()

var pane=xui.create('Pane',{position:'relative',

width:"auto",height:"auto",

html:"<strong>auto height</strong>"

})

.setCustomStyle({"KEY":"border:solid 1px #888"})

.show()

xui.create('Pane',{position:'relative',

width:"auto",height:100,

html:"<strong>bottom</strong>"

})

.setCustomStyle({"KEY":"border:solid 1px #888"})

.show()

xui.create("SButton")

.setLeft(140)

.setTop(30)

.setCaption("Add content")

.onClick(function(){

pane.append(xui.create("Pane").setPosition("relative").setHeight(30).append("Input"))

})

.show()

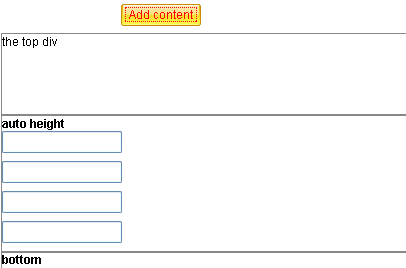
At top

Middle

At bottom

Adds contents

**Output:**



Automatic growth

## UI Control’s Drag&Drop

### Drag&Drop control among containers

**Input:**

var panel1=xui.create('Panel',{position:'relative', dock:'none',width:150}).show();

var panel2=xui.create('Panel',{position:'relative', dock:'none',width:150}).show();

var btn=xui.create('Button',{left:10,top:10}).show(panel1);

var onDrop=function (profile, e, node, key, data) {

var dd = xui.DragDrop.getProfile(), data = dd.dragData;

if(data){

var btn=xui.getObject(data);

profile.boxing().append(btn.boxing());

}

};

btn.draggable('iAny',btn.get(0).getId(),null,{shadowFrom:btn.getRoot()});

panel1.setDropKeys('iAny').onDrop(onDrop);

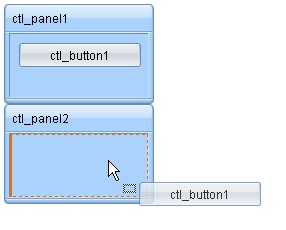
panel2.setDropKeys('iAny').onDrop(onDrop);

Sets onDrop function

Sets draggable

Sets droppable

**Output:**



### List sorting 1

**Input:**

xui.create("List",{

items:["item a","item b","item c","item d"]

})

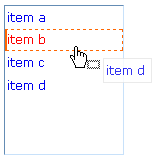
.setDragKey("list")

.setDropKeys("list")

.show()

Sets drag key and drop keys.

**Output:**



### List sorting 2

**Input:**

xui.create("List",{

items:["item a","item b","item c","item d"]

})

.setDragKey("list")

.setDropKeys("list")

.onDropMarkShow(function(profile,e,src,key,data,item){

if(item){

xui.DragDrop.setDragIcon('move');

xui.DragDrop.setDropFace(null);

profile.getSubNodeByItemId('ITEM',item.id).css('borderTop','dashed 1px');

return false;

}

})

.onDropMarkClear(function(profile,e,src,key,data,item){

if(item){

xui.DragDrop.setDragIcon('none');

profile.getSubNodeByItemId('ITEM',item.id).css('borderTop','');

return false;

}

})

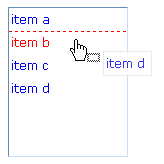
.show()

Custom appearance

Sets drag key and drop keys

Restores appearance

**Output:**



## Form

### Form 1

**Input:**

Class.destroy('App');

Class('App', 'xui.Module',{

Instance:{

iniComponents:function(){

// [[code created by CrossUI UI Builder

var host=this, children=[], append=function(child){children.push(child.get(0))};

append((new xui.UI.SLabel)

.setHost(host,"slabel1").setLeft(80).setTop(60).setWidth(44).setCaption("Name:") );

append((new xui.UI.SLabel)

. setHost(host,"slabel2").setLeft(80).setTop(90).setCaption("Age:").setWidth(44));

append((new xui.UI.Input)

.setHost(host,"iName").setLeft(130).setTop(60).setValueFormat("[^.\*]").setValue("Jack"));

append((new xui.UI.ComboInput)

.setHost(host,"iAge").setLeft(130).setTop(90).setType("spin").setIncrement(1).setMin(20).setMax(60).setValue("35"));

append((new xui.UI.SCheckBox)

.setHost(host,"cFull").setLeft(130).setTop(130).setCaption("Full time"));

append((new xui.UI.SButton)

.setHost(host,"submit").setLeft(130).setTop(170).setCaption("SUBMIT").onClick("\_submit\_onclick"));

return children;

// ]]code created by CrossUI UI Builder

},

\_submit\_onclick:function (profile, e, src, value) {

if(!this.iName.checkValid()){

xui.alert('You must specify Name');

return;

}

var name=this.iName.updateValue().getValue(), age=this.iAge.updateValue().getValue(),

full=this.cFull.updateValue().getValue();

xui.alert(\_.serialize({name:name,age:age,full:full}))

}}

});

(new App).show();

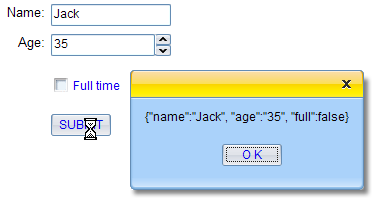
Form validation

Code created by Designer

event

Collects data

**Output:**



### DataBinder

There are two types of data in DataBinder:

* + 1. The inner data property:

setData function: to get the inner data property;

getData function: to set the inner data property;

* + 1. The bound UI controls (if it exists):

updateValue function: to update the bound UI controls “UI value” to inner value, and removed dirty marks;

checkValue function: to check the bound UI controls;

getDirtied function: to get the dirtied values from the bound UI controls;

updateDataToUI function: to update the inner data to the bound UI controls;

updateDatafromUI function: to update the inner data from the bound UI controls;

**The bound UI (If it exists)**

updateValue / checkValue / getDirtied

**Inner data property**

updateDatafromUI

updateDataToUI

setData

getData

**Input:**

Class.destroy('App');

Class('App', 'xui.Module',{

Instance:{

iniComponents:function(){

// [[code created by CrossUI UI Builder

var host=this, children=[], append=function(child){children.push(child.get(0))};

append((new xui.DataBinder).setHost(host,"binder").setName("binder"))

append((new xui.UI.SLabel)

.setHost(host,"slabel1").setLeft(80).setTop(60).setWidth(44).setCaption("Name:") );

append((new xui.UI.SLabel)

.setHost(host,"slabel2").setLeft(80).setTop(90).setCaption("Age:").setWidth(44));

append((new xui.UI.Input) .setDataBinder("binder").setDataField("name")

.setHost(host,"iName").setLeft(130).setTop(60).setValueFormat("[^.\*]").setValue("Jack"));

append((new xui.UI.ComboInput) .setDataBinder("binder").setDataField("age")

.setHost(host,"iAge").setLeft(130).setTop(90).setType("spin").setIncrement(1).setMin(20).setMax(60).setValue("35"));

append((new xui.UI.SCheckBox) .setDataBinder("binder").setDataField("isfull")

.setHost(host,"cFull").setLeft(130).setTop(130).setCaption("Full time"));

append((new xui.UI.SButton)

.setHost(host,"submit").setLeft(130).setTop(170).setCaption("SUBMIT").onClick("\_submit\_onclick"));

return children;

// ]]code created by CrossUI UI Builder

},

\_submit\_onclick:function (profile, e, src, value) {

if(!this.binder.checkValid()){

xui.alert('One or some invalid fields exits!');

return;

}

this.binder.updateDataFromUI();

xui.alert(\_.serialize(this.binder.getData()))

}}

});

(new App).show();

Form validation

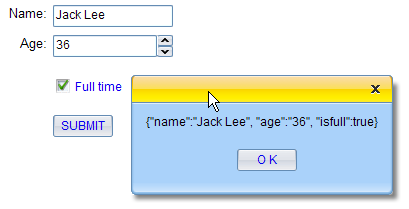
Code created by Designer

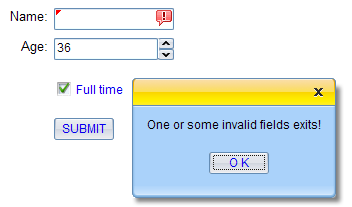
Sets dataBinder and dataField to each control

Collects data

Adds a DataBinder, sets name property

**Output:**





## Custom UI Styles

### Custom one instance only - 1

**Input:**

xui.CSS.remove("id","my\_css");

xui.CSS.addStyleSheet(".xui-sbutton-custom-focus{font-weight:bold;color:#ff0000;}", "my\_css");

(new xui.UI.SButton)

.setCaption("Use setCustomClass ")

.setTheme ("custom")

.show();

Sets theme to this instance

Adds CSS. You should put those into a CSS file in your real application

Theme key words

**Output:**



### Custom one instance only - 2

**Input:**

(new xui.UI.SButton)

.setCaption("Use setCustomStyle")

.setCustomStyle({

FOCUS:"font-weight:bold;color:#ff0000;"

})

.show();

Custom FOCUS node style

**Output:**



### Custom one instance only - 3

**Input:**

xui.CSS.remove("id","my\_css");

xui.CSS.addStyleSheet(".my-class{font-weight:bold;color:#ff0000;}", "my\_css");

(new xui.UI.SButton)

.setCaption("Use setCustomClass ")

.setCustomClass({

FOCUS:"my-class"

})

.show();

Custom FOCUS node className

Adds CSS. You should put those into a CSS file in your real application

**Output:**



### Custom one instance only - 4

**Input:**

xui.CSS.remove("id","my\_css");

xui.CSS.addStyleSheet("#myctrl1 .xui-sbutton-focus{font-weight:bold;color:#ff0000;}", "my\_css");

(new xui.UI.SButton)

.setCaption("Use domId")

.setDomId("myctrl1")

.show();

Gives a domId

Adds CSS. You should put those into a CSS file in your real application

**Output:**



### Custom one instance only - 5

**Input:**

(new xui.UI.SButton)

.setCaption("Use getSubNode and css ")

.onRender(function(profile){

profile.getSubNode('FOCUS').css({

fontWeight:'bold',

color:'#ff0000'

});

})

.show()

After it was rendered into DOM

**Output:**



### Custom one instance only - 6

**Input:**

xui.CSS.remove("id","my\_css");

xui.CSS.addStyleSheet(".my-listitem{font-weight:bold;color:#ff0000;}", "my\_css");

xui.create('List',{items:[{

id:"item 1",

itemStyle:"border:dashed 1px #00ff00;margin:2px;"

},{

id:"item 2",

itemStyle:"border:dashed 1px #0000ff;margin:4px;"

},{

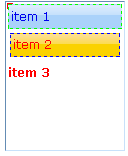
id:"item 3",

itemClass:"my-listitem"

}]}).show()

Adds CSS. You should put those into a CSS file in your real application

**Output:**



### Custom style for an UI Class

**Input:**

xui.CSS.remove("id","my\_css");

xui.CSS.addStyleSheet(".xui-sbutton-focus{font-weight:bold;color:#ff0000;}", "my\_css");

(new xui.UI.SButton({position:'relative'})).show();

(new xui.UI.SButton({position:'relative'})).show();

(new xui.UI.SButton({position:'relative'})).show();

(new xui.UI.SButton({position:'relative'})).show();

(new xui.UI.SButton({position:'relative'})).show();

All instances were changed

Adds CSS. You should put those into a CSS file in your real application

**Output:**

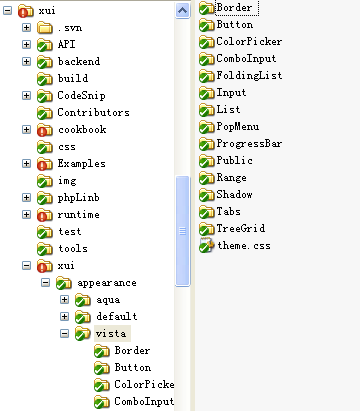


### Custom style for all UI Class - skin

There are three system built-in skins in CrossUI: default, vista and aqua. You can use xui.setTheme to switch the skin. You can also add your own custom skin easily. Only two steps:

#### First: Copy

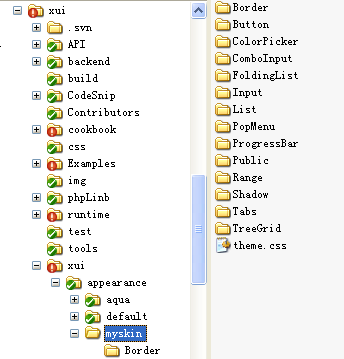
All skins are in “runtime/xui/appearance”, you can create an new folder (e.g. ‘myskin’), and copy all directories and files in an existing skin folder to it.



A skin folder

Skin pictures folder

Skin CSS file



Copy from vista folder

#### Second: Little by little, modify pictures and CSS

For example, we modifies corner.gif file in Button folder.



After that,

**Input:**

xui.create('Button').show();

\_.asyRun(function(){

xui.setTheme('myskin')

},2000);

**Output:**



# The end

Crossui.com

All rights reserved.