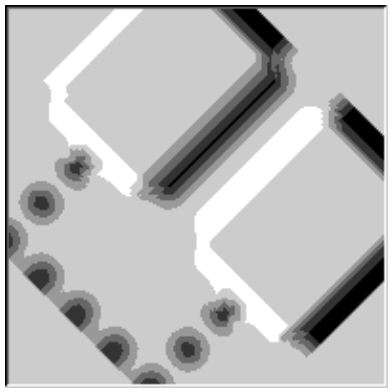


diputree 2.2

Table of Contents

Table of Contents

Table of Contents

Introduction

What are applets ?

Applets are small components that can reside within a web document.
They're just like any other component in your web document (such as paragraphs, forms, tables, graphics, etc.).

The difference lies in the fact that applets can have behavior, respond to user and/or system events and trigger events themselves. You can think of them as little applications embedded within your web document.

The advantages of applets over static pages/components are ample:

- your users are more acquainted with interactive interfaces
- you have more power over the overall interface you implement on your web site
- it can give your web site a firm structure, without imposing too much constraints on your web design

dipu applets are configurable

With version 2.2, you have total control over the applet’s representation and functionality.

Functionality

- diputree is scriptable (in real time)
- a ‘site wide’ consistent look an feel
- keyboard support
- user definable icons (downloadable icons)
- configurable space between container and non-container icons
- configurable colors for text, background and lines
- background images are now supported
- more control over the size of the scrollbars
- lines, expanders and icons can be switched off
- the root of the tree can be hidden
- pop up tips are supported (state dependent!)
- status bar information messages
- automatic expansion and/or contraction of containers (state dependent!)
- frame destination can be set for every individual link
- multiple frame destinations (unlimited)
- ‘opened’ and ‘closed’ icons (state dependent)
- fonts can be set for every individual entry
- text can be aligned for every individual entry
- standard icons are compiled in and do not require any additional downloads
- configurable margins
- all possible URL combinations are supported, even partial and relative URL’s
- default values for all options exist
- it uses java 1.0.2 so it will run on every browser supporting java

New functionality in version 2.2

- a configuration program (a graphical user interface to configure the applet)
- a completely new event framework (internal & external handlers)
- conditional loading through external handlers
- javascript call-back through external handlers
- scroll bars are now implemented in diputree for more compatibility
  (java's own scrollbar implementation is buggy)
- extensive error reporting through the java console

dipu applets have an extremely small footprint

Even with all this functionality it is less than 15KB in jar format or 32Kb in class format,
NO ADDITIONAL DOWNLOADS ARE REQUIRED!

With a 33.6Kb modem and Netscape 4+ or ie4+ it only takes 4 seconds to download the applet!

updates and more information

For updates and information please refer to http://www.dipu.com .

1. The tables

Introduction

using applet parameters

```
<APPLET CODE="diputree.class" ARCHIVE="diputree.jar" WIDTH=150
HEIGHT=250>

<PARAM name=recordseparator value="^">

<PARAM name=fieldseparator value="|">

<PARAM name=options value="
^containershiftx|20
^containershifty|20

.
.
.
^tipwait|1000
^tipfontprofile|tipdefaultfont ">
</APPLET>
```

The parameter tag contains the following parts

- <PARAM
  this part of the tag indicates that a new parameter definition starts here
- name="the_name_of_your_parameter"
  this part of the tag declares the name of the parameter
- value="the value of the parameter"
  this part of the tag contains the actual value of the parameter
- >
  this part of the tag indicates that the parameter definition ends here

passing database tables

The applet uses the parameter tag for passing complete database tables from the applet definition in your web page to the applet.

Such a parameter can contain multiple records and every record can contain multiple fields.

So first you have to define which character will serve as a record separator and which character will serve as a field separator. This is done through the parameters ‘recordseparator’ and ‘fieldseparator’. These are the only 2 single value parameters, they are meta-parameters (parameters who describe other parameters)

So if we want to define 4 persons whose names are Sven, Danny, Jessica and Serge, with their home address Oostende, Baal, Mechelen and Mechelen and with the ages of 25,28,25 and 29, then we have to define the following parameter (assuming recordseparator is ^ and fieldseparator is |)

```
^Sven|Oostende|25
^Danny|Baal|28
^Jessica|Mechelen|25
^Serge|Mechelen|29
```

using default values

Rule:

> You always need to enter the fieldseparator, even if you are using a default value for a field.

Exception:

> You can omit the fieldseparators when a default field or the end of the record follows every default field.

examples

First the general rule

Let us assume that the default city is set to Mechelen.
Now our parameter looks like this:

```
^Sven|Oostende|25
^Danny|Baal|28
^Jessica||25
^Serge||29
```

For Serge and Jessica we used the default city. But notice that although we used the default value for the city field, we still had to use the field (we left it blank!)

The exception

Now assume that the default age is 25.
This would make the parameter look like this:

```
^Sven|Oostende
^Danny|Baal|28
^Jessica
^Serge||29
```

Notice that Jessica does not require any fieldseparator, because a default field or the end of the record follows every default field (jessica has 2 default values). Now the parser (the code that reads in the fields) can not be mistaken, he knows that all the ‘non defined’ fields are default fields.

loading tables from external files

You can instruct the applet to fetch the table from an external file, instead of ‘hardwiring’ the fields of your table in a parameter tag in your web page.

- Use a prefixed URL in the value part of the parameter tag.
  A prefixed URL is an URL with the characters "url:" prefixed to it.
  So just add "url:" in front of the URL where your values file is located.
  You can use fully qualified or relative URL’s.
- Put your table in a text file.
  The text file should look exactly like the value part of the parameter tag when no external files are used.
  So you can just cut and paste what was in the value part of the parameter and paste it in an empty file.

examples:

- The web page
  <APPLET CODE="diputree.class" ARCHIVE="diputree.jar" WIDTH=150 HEIGHT=250>
  <PARAM name="entries" value="url:http://www.mysite.com/mydirectory/entries.txt">
  </APPLET>
- The external file (entries.txt)
  ^parent||Hi, I'm the parent
  ^child-1|Hi, I'm child 1|parent
  ^child-2|Hi, I'm child 2|parent

a site wide consistent look and feel

If you use the same files for the options, fonts and images parameters your site will look and feel the same everywhere! Updates in a file will be reflected site wide.

The entries table

This is a 'database table' table. Please refer to passing database tables for a detailed explanation of the database table format. In all the following examples we assume that the recordseparator is ^ and the fieldseparator is | .

Columns 5,6,7 and 8 are external keys who point to records in other database tables. Field 5 is an external key for the links table, columns 6 and 7 are external keys for the images table and column 8 is an external key for the fonts table.

The entries table defines the hierarchical structure of your tree.

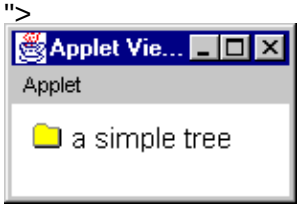
Column	Name	Type	Default	Range
0	Key	String	No default, obligatory field	
1	Name	String	No default, obligatory field	
2	Parent's key	String	No default, obligatory field	
3	Type	Character	'c'	'c' = container, 'n' = non container
4	State	Character	'c'	'c' = closed, 'o' = opened
5	List of linksprofiles	Space separated string(s)		
6	Name of the icon in the images table for closed/non-selected	String	"closedfolder" where type is c , "closeddocument" where type is n	
7	Name of the icon in the images table for openened/selected	String	"openedfolder" where type is c , "openeddocument" where type is n	
8	Font profile	String	"defaultfont"	
9	Font alignment (relative to its icon)	String	"middle"	"top", "middle", "bottom"
10	Tip/expansion (when state is closed)	String		"auto" will enable auto expansion of containers

11	Tip/contraction (when state is opened)	String		‘auto’ will enable auto contraction of containers
12	Status message (will be displayed on the browser’s status bar)	String		

- Column 0 is the (primary) key. This key will be used when referring to this record.
- Column 1 is the name that will be displayed in your tree.
- Column 2 points to the key of the parent of this entry. The hierarchy starts with the entries that have the empty string "" or "-start-" as its parent. Every other entry must be a child or a grandchild from these entries. Only container entries can have children.
- Column 3 defines the type of the entry. A container can contain other containers or noncontainers, noncontainers are final and can not have any children.
- Column 4 defines the state of the entry. It defines if the container’s contents is visible (opened) or not visible (closed). For a noncontainer it defines which icon should be displayed.
- Column 5 defines the linkprofiles that should be activated when the user clicks on the entry. Multiple linkprofiles are supported and should be separated with a space character. So clicking on an entry can trigger multiple links to different frames.
- You can define your own icons. Column 6 contains the icon for the opened state, column 7 for the closed state.
- Column 8 and 9 define the font and alignment used for the displayed name.
- Column 10 and 11 define the contents of the tip that should be displayed if the pointing device is located above this entry and is inactive for a given period of time. Column 10 will be displayed if the state is closed, column 11 when the state is opened. If column 10 or 11 contains the string "auto" then automatic expansion of containers is activated.
- Column 12 defines the contents of the status message that should be displayed if the pointing device is moved above this entry.

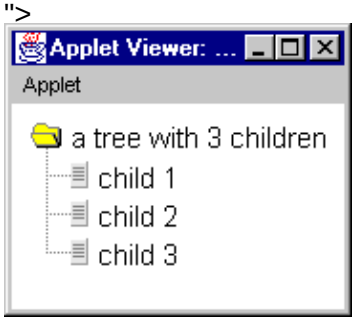
examples

```
<PARAM name=entries value="
^root|a simple tree|-start-
">
```

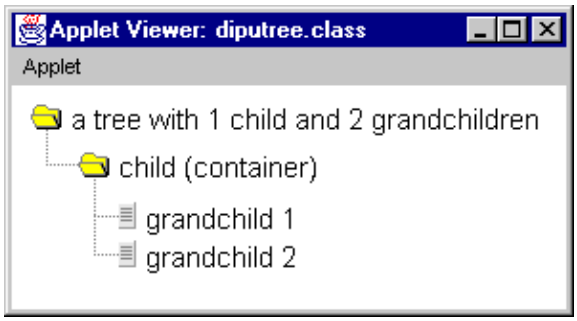



```
 <PARAM name=entries value="
^root|a tree with 3 children|-start-|c|o
^child1|child 1|root|n
^child2|child 2|root|n
^child3|child 3|root|n
```

">

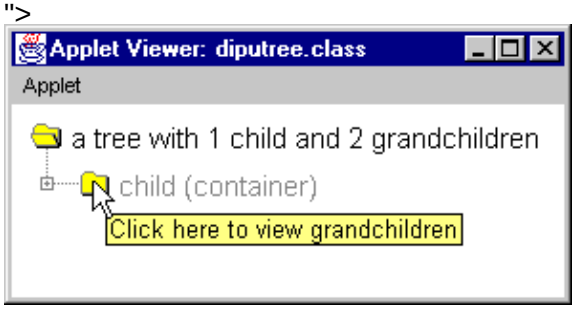


<PARAM name=entries value="
^root|a tree with 1 child and 2 grandchildren|-start-|c|o
^child1|child (container)|root|c|o
^grandchild1|grandchild 1|child1|n
^grandchild2|grandchild 2|child1|n
">



using pop up tips

<PARAM name=entries value="
^root|a tree with 1 child and 2 grandchildren|-start-
^child1|child (container)|root|c|c|||||Click here to view grandchildren
^grandchild1|grandchild 1|child1|n
^grandchild2|grandchild 2|child1|n
">



The links table

This is a 'database table' parameter. Please refer to passing database_tables for a detailed explanation of the database table format. In all the following examples we assume that the recordseparator is ^ and the fieldseparator is | .

Field nr. 5 in the entries parameter contains an external key(s) referring to a record(s) in this table. Each key is separated by a space.

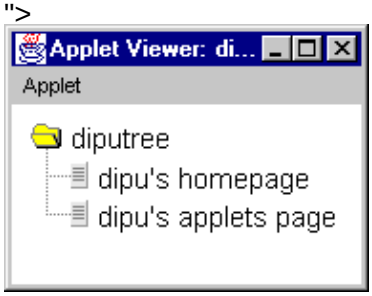
When the entry is clicked all the link profiles will be activated. So it is possible to change the contents of multiple frames!

Record definition				
field	**Name**	**Type**	**Default**	**Range**
0	Key	string	no default, obligatory field	
1	Destination URL	URL	no default, obligatory field	Fully qualified URL's as well as relative URL's are supported (more info on url's).
2	Destination frame	String	frame in the options parameter	

- field 0 is the (primary) key. This key will be used when referring to this record.
- field 1 is the URL of the web document that should be displayed.
- field 2 is the destination frame.

examples

```
<PARAM name=entries value="
^root|diputree|-start-|c|o
^child1|dipu's homepage|root|n|c|homepage
^child2|dipu's applets page|root|n|c|applets
">
 <PARAM name=links value="
^homepage|http://www.dipu.com|dipuframe
^applets|http://www.dipu.com/products
">
```



The images table

This is a 'database table' parameter. Please refer to passing database_tables for a detailed explanation of the database table format. In all the following examples we assume that the recordseparator is ^ and the fieldseparator is | .

Field nr. 6 and 7 in the entries paramter are external keys referring to records in this table.

Record definition				
fields	**Name**	**Type**	**default**	**Range**
0	Key	string	No default, obligatory field	
1	The image's location	URL	No default, obligatory field	Fully qualified URL's as well as relative URL's are supported
2	X coordinate	Integer	0	
3	Y coordinate	Integer	0	
4	Width	Integer	Width of the image	
5	Height	Integer	height of the image	

- fields 0 is the (primary) key. This key will be used when referring to this record.
- fields 1 is the URL of the image.
- fields 2, 3, 4 and 5 define a crop region. If you only need a part of the image, then you can define a crop region

background

A special key is "background", it allows you to use an image as your background. If the image is smaller than the surface of the applet, then the background image will be tiled.

example

```
<PARAM name=images value="
^background|clouds.jpg
">
```

 the background image

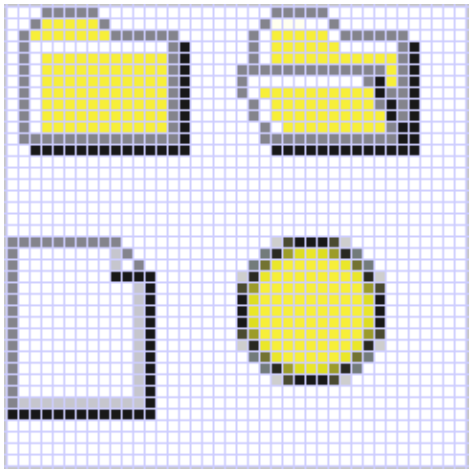
Notice that folders are not ‘transparent’. The java language doesn’t allow creating transparent off-screen images. In the next paragraph will we demonstrate how you can use external icons with transparent backgrounds.

custom images for icons

You can use 1 file per icon or pack different icons in one file. Packing multiple icons in 1 file results in shorter download times because your browser doesn’t need to make multiple internet connections.

We will pack 4 icons with a transparent background in 1 gif image file. Please refer to the manual of your paint program on how to do this.

If we blow this up we get



Now we can define crop regions for the icons that are contained in this image.

We shall only change the default openedfolder and closedfolder icons with their transparent counterparts.

closedfolder starts at the point (0,0), this is the upper left corner, and is 16 pixels wide and 13 pixels high.

openedfolder starts at point (20,0) and is also 16 pixels wide and 13 pixels high

example

<PARAM name=images value="

```
^background|clouds.jpg
^closedfolder|treetrans.gif|0|0|16|13
^openedfolder|treetrans.gif|20|0|16|13
">
```

more pecial image keys

We already saw "background" as a special image key, but most of the built in images can be replaced.

key	description
closedfolder	the icon of the container when it is in the closed state
openedfolder	the icon of the container when it is in the opened state
closeddocument	the icon of the non-container
contractedexpander	the icon of the expander when its container is in the closed state
expandedexpander	the icon of the expander when its container is in the opened state
closedfolder	the icon of the container when it is in the closed state
openedfolder	the icon of the container when it is in the opened state
closeddocument	the icon of the non-container
contractedexpander	the icon of the expander when its container is in the closed state
expandedexpander	the icon of the expander when its container is in the opened state

The fonts table

This is a 'database table' parameter. Please refer to passing database tables for a detailed explanation of the database table format. In all the following examples we assume that the recordseparator is ^ and the fieldseparator is | .

Field nr. 8 in the entries parameter is an external key referring to a record in this table.

Record definition				
fields	**Name**	**Type**	**Default**	**Range**

0	Key	String	No default, obligatory field	
1	Font name	String	No default, obligatory field	All fonts supported by the system.
2	Font style	Integer	No default, obligatory field	0 = plain, 1 = bold, 2 = italic ; 3 = bold & italic
3	Font size	Integer	No default, obligatory field	

- fields 0 is the (primary) key. This key will be used when referring to this record.
- fields 1 defines the name of the font. Because the amount of available fonts depends on the underlying OS, it is safer to use generic font names or names of font families like: "SansSerif", "Serif" and "Monospaced".
- fields 2 defines the style applied to the font.
- fields 3 defines the size of the font.

font profiles

fontprofile and tipfontprofile from the options parameter

In the options parameter we defined the fontprofile and tipfontprofile options. Which implies that the fontprofile(s) defined by these options are required too.

So we need at least 2 fontprofiles (actually we only need 1 fontprofile because fontprofiles can be shared).

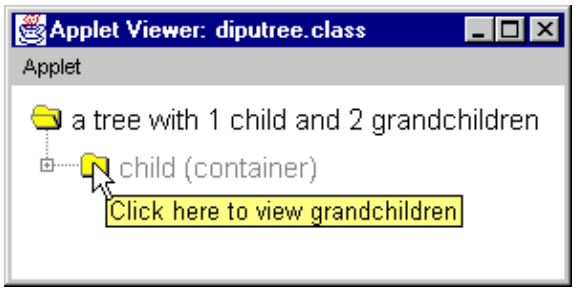
built in font profiles

defaultfont and tipdefaultfont are built in font profiles. These 2 profiles can be used without the need to define them first in the fonts parameter.
You can override these built-in fontprofiles.

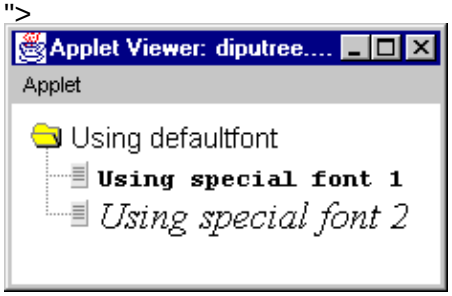
examples

(assumptions: fontprofile = defaultfont, tipfontprofile = tipdefaultfont)

```
<PARAM name=fonts value="
^defaultfont|SansSerif|0|15
^tipdefaultfont|SansSerif|0|13
">
```




```
 <PARAM name=entries value="
^root|Using defaultfont|-start-|c|o
^child1|Using special font 1|root|n|||||font1
^child2|Using special font 2|root|n|||||font2
">
 <PARAM name=fonts value="
^defaultfont|SansSerif|0|15
^tipdefaultfont|SansSerif|0|13
^font1|Monospaced|1|13
^font2|Serif|2|20
">
```



The options table

This is a 'database table' parameter. Please refer to passing database_tables for a detailed explanation of the database table format. In all the following examples we assume that the recordseparator is ^ and the fieldseparator is | .

options records are mappings

The options table is special kind of database table. It has records that contain mappings between keys and their associated values.
We will use the terms "key" and "value" instead of "field 0" and "field 1".

all options are optional

All options are optional. diputree has reasonable defaults for every option in the options parameter. Although defaults exist you should not count on the values of these defaults, they may change over time. So, if you want to implement a special ‘look and feel’ you should declare your own options.

colors

backgroundcolor, linecolor, textcolor, selectedcolor, selectedbackgroundcolor, tipcolor, tipbackgroundcolor, mouseovercolor, underlinecolor, foldercolor, folderhighcolor, folderlowcolor, doccolor and boxcolor

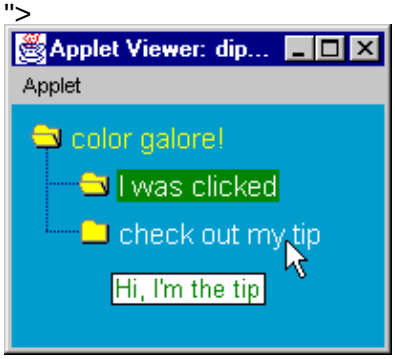
Colors are represented by a hexadecimal value. The first 2 digits are for the red component, the next 2 for the green component and the final 2 for the blue component.
The value of a color component can range from 00 to FF (00 = no intensity, FF = full intensity)
e.g. FFFFFF is the hexadecimal representation of white, 000000 is black.

key	**value**	
name	**type**	**description**
backgroundcolor	hexadecimal integer	the color of the background
linecolor	hexadecimal integer	the color of the lines connecting the icons together
textcolor	hexadecimal integer	the color of the text
selectedcolor	hexadecimal integer	the color of the entry when it has been selected (clicked)
selectedbackgroundcolor	hexadecimal integer	the background color of the entry when it has been selected (clicked)

tipcolor	hexadecimal integer	the color of the pop up tip
tipbackgroundcolor	hexadecimal integer	the background color of the pop up tip
mouseovercolor	hexadecimal integer	the color of an entry when the mouse moves over it
underlinecolor	hexadecimal integer	the color of the line shown under an entry when the mouse moves over it
foldercolor	hexadecimal integer	the color of the default folder
folderhighcolor	hexadecimal integer	the color of the default folder (high intensity)
folderlowcolor	hexadecimal integer	the color of the default folder (low intensity)
doccolor	hexadecimal integer	the color of the default document
boxcolor	hexadecimal integer	the color of expander box

example

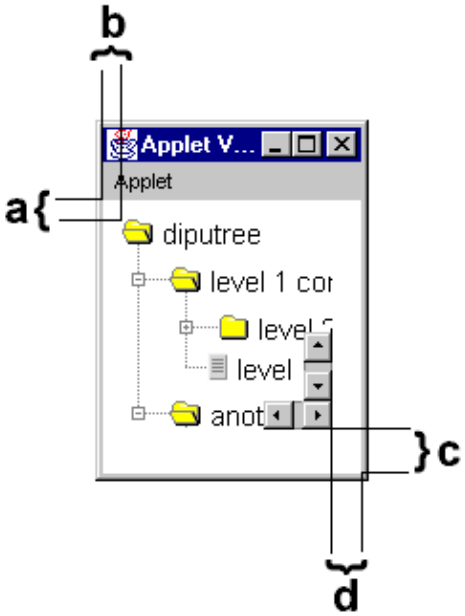
```
<param name="options" value="
^backgroundcolor|0099CC
^linecolor|000080
^textcolor|FFFF00
^selectedcolor|FFFFFF
^selectedbackgroundcolor|008000
^tipcolor|008000
^tipbackgroundcolor|FFFFFF
^mouseovercolor|FFFFFF
">
```



applet margins

marginleft and marginright

key	value	
name	**type**	**description**
marginup	decimal integer	the upper margin
margindown	decimal integer	the lower margin
marginleft	decimal integer	the left margin
marginright	decimal integer	the right margin



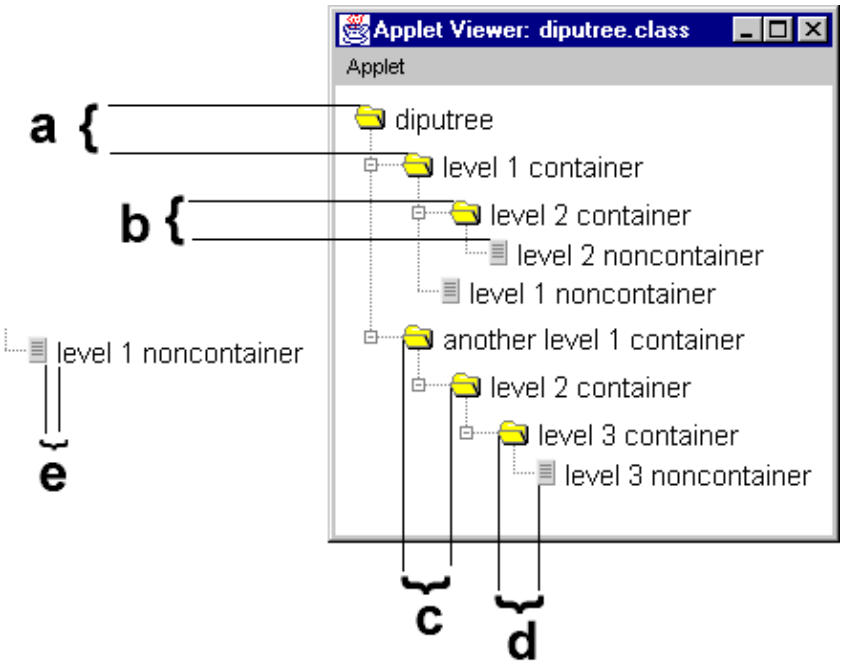
A: marginup, B: marginleft, C: margindown, D: marginright

tree margins

textshiftx, containershiftx, containershifty, noncontainershiftx and noncontainershiftx

key	value	
name	**type**	**description**
textshiftx	decimal integer	the value that the text should shift in the horizontal direction, relative to its icon
containershiftx	decimal integer	the value that a child container should shift in the horizontal direction, relative to its parent container.
containershifty	decimal integer	the value that the container should shift in the vertical direction, relative to the previous icon (be it a

		container or a noncontainer).
noncontainershiftx	decimal integer	the value that a noncontainer should shift in the horizontal direction, relative to its container.
noncontainershifty	decimal integer	the value that a noncontainer should shift in the vertical direction, relative to the previous icon (be it a container or a noncontainer)



a:containershifty,b:noncontainershifty, c:containershiftx, d: noncontainershiftx e: textshiftx

scrollbars

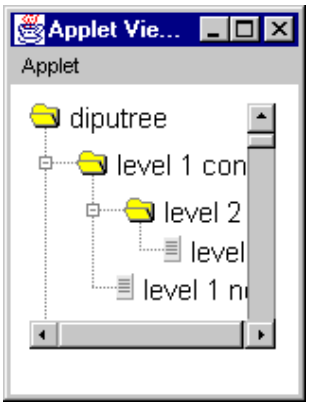
scrollbarhorizontaldivider, scrollbarverticaldivider, scrollbarwidth, scrollbuttonsize, unitdivider, blockdivider and scrollspeed

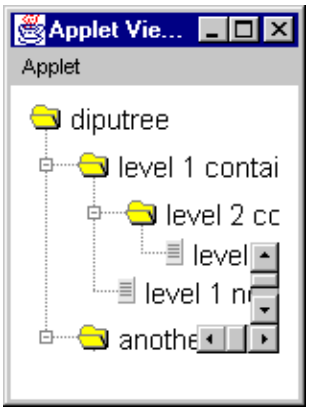
The initial size (100%) and the scrollbardivider value control the representation of the scrollbar. It can be divided by an integer value of 1 or greater.
e.g. a divider value of 3 will give a 33% coverage of the scrollbar.

key	**value**	
name	**type**	**description**
scrollbarhorizontaldivider	decimal integer	the divider value for the horizontal scrollbar (value can not be 0)
scrollbarverticaldivider	decimal integer	the divider value for the vertical scrollbar (value can not be 0)

scrollbarwidth	decimal integer	the width of the scrollbar in pixels
scrollbuttonsize	decimal integer	the width and height of the buttons
unitdivider	decimal integer	the amount (divider value) that should be scrolled when the user wants to scroll by 1 unit (by clicking on the scroll button)
blockdivider	decimal integer	the amount (divider value) that should be scrolled when the user wants to scroll by 1 unit (by clicking on the scroll button)
scrollspeed	decimal integer	the amount of time between 2 consecutive scrollings in milliseconds

Example





<param name="options" value="
^scrollbarhorizontaldivider|1
^scrollbarverticaldivider|1
">

<param name="options" value="
^scrollbarhorizontaldivider|3
^scrollbarverticaldivider|3
">

frame

Your web document can be divided into different frames. Every frame acts as a container for another web document, effectively creating a hierarchy of (contained) subdocuments.

You can assign a name to each frame individually. By default your browser assigns names to some special frames, such as: _blank (a new frame), _self (this frame), _parent (this frame's parent) and _top (the uppermost parent frame, the root of the frame hierarchy).

key	**value**	
name	**type**	**description**

frame	String	the name of the default frame where the pages will be displayed, can be overridden in the links parameter.

- (value: string)

fontprofile

key	**value**	
name	**type**	**description**
fontprofile	String	the default font profile, can be overridden in the entries parameter

showstatus

key	**value**	
name	**type**	**description**
showstatus	integer boolean	showing status messages can be switched on or off (0 = off, 1 = on)

mouse movements

mouseover and underline

When the mouse moves over an entry, the entry can respond by highlighting or underlining itself.

key	**value**	
name	**type**	**description**
mouseover	integer boolean	highlighting due to the mouse being rolled over can be switched on or off (0 = off, 1 = on)
underline	integer boolean	underlining due to the mouse being rolled over can be switched on or off (0 = off, 1 = on)

tips

tipwait and tipfontprofile

Pop up tips are small messages that pop up whenever there is no user action during a specified period.

key	**value**	
name	**type**	**description**
tipwait	decimal integer	the period before a tip pops up defined in milliseconds (1000ms = 1sec)
tipfontprofile	decimal integer	the default font profile to use for the tips. If the tip extends beyond the applet's boundaries the font size is decremented until it can fit within the boundaries (the default font profile for pop up tips is tipdefaultfont).

selected

The key of the entry that should be selected when the applet is first loaded.

key	**value**	
name	**type**	**description**
selected	string	the key from the entries table, that should be set as the selected record.

externalhandlers

Not all browsers support external handlers. If this option is set, then the applet shall check during initialization if external handlers are supported.

If this feature is supported then the applet will leave this option as is (a value of 1), if this feature is not support then the applet will unset this option (new value will be 0).

In your script you can get the value of this option to determine if external handlers are enabled.

IMPORTANT

External handlers require the MAYSCRIPT attribute to be present in the applet definition.

key	**value**	
name	**type**	**description**
externalhandlers	boolean integer	instructs the applet to test if external handlers are supported (values: 0 = don't test an consequently don't allow external handlers

		1= test for external handlers support, if support is not available set value to 0)

asyncload

For utter compatibility you can disable (a value of 0) the asynchronous loading of images. This will prevent the applet from showing a "please wait while loading" message during image loading.

key	**value**	
name	**type**	**description**
asyncload	boolean integer	tells the applet to enable asynchronous loading of images (values: 0 = don't load asynchronously, 1= load asynchronously,)

baseurl

If you use relative URL's then baseurl is used to complete your URL.
So if all your link URL's are located in a directory " /mydirectory" at your web site "http://www.mywebsite.com" than you can define a baseurl "http://www.mywebsite.com/mydirectory".
Now you only have to use the filename part of your URL instead of the complete URL in your links parameter.

The default URL for baseurl is the URL of the directory that contains the applet.
So if your applet is located in "http://www.mysite.com/mydirectory/diputree.class", then the baseurl will be "http://www.mysite.com/mydirectory".

baseurl is used in the links parameter.

key	**value**	
name	**type**	**description**
baseurl	String	the url that will be used to complete relative and/or uncomplete URL references (values: a Fully Qualified URL or a relative URL)

showing icons, lines and expanders

showlines, showicons and showexpanders

You can individually show or hide icons, their lines and/or their expanders.

key	**value**

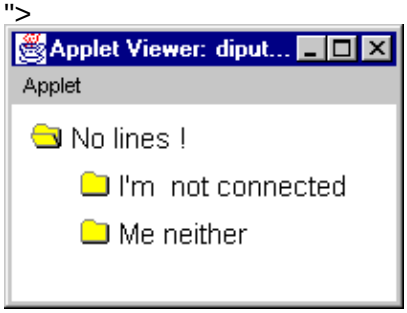
name	type	description
showlines	decimal integer	switch showing lines on or off (Values: 0 = hide, 1 = show).
showicons	decimal integer	switch showing icons on or of (Values: 0 = hide, 1 = show).
showexpanders	decimal integer	switch showing expanders in status bar on or off (Values: 0 = hide, 1 = show).

Special feel when expanders are shown or hidden

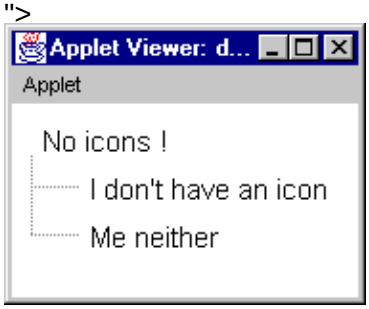
- expanders are shown
  Clicking on an expander will open or close the container, clicking on the icon or the text will show the contents
- expanders are hidden
  Clicking on the icon or the text wil open or close the container AND show the contents

examples

```
<param name="options" value="
^showlines|0
">
```




```
<param name="options" value="
^showicons|0
">
```



grouping

Non-containers can be grouped together. If grouping is disabled entries will be displayed in the order they were entered.

key	value	
name	**type**	**description**
grouping	decimal integer	switch grouping lines on or off (Values: 0 = no grouping, 1 = grouping).

autoclose

If a top level container is opened and autoclose is enabled, then will all other top level containers be closed automatically. This feature is especially useful when top level containers have a large amount of children.

key	value	
name	**type**	**description**
grouping	decimal integer	switch grouping lines on or off (Values: 0 = no grouping, 1 = grouping).

2. Scripting

scripting is programming

You can use scripts as simple 'command batch files' or as a full fledged 'programming language'.

Some refer to scripting as ‘the glue’ that connects different components together. A component is a collection of code and data that represents a ‘higher level’ of functionality. The dipu applets for example are a components.

Because scripting is aimed at a big audience the syntax is kept as simple as possible.

The objects of the script are dynamically typed and they don’t need to be declared, beware of these 'features' of some scripting languages.

- Dynamically typed
  The type of an object can change dynamically. Your object can start it’s life as a string of text, it can change its type to an integer and finally transform into a color type.
  This can be a potential pitfall.
- They don’t need to be declared
  You can start using an object right away, there’s no need to define nor declare it. This really has a potential for pitfalls and some hard ones to find, a small typing error can result in the creation of a second object, although this was not intended!

web scripting

Today there are two standards for scripting in a web document, Javascript and Vbscript.
dipu applets are accessible from both languages.

reflection

If you load a web document in your browser, all the elements contained in your document are ‘reflected’ in the scripting environment. This means that all elements (such as all html tags) that are contained in your document are mapped to objects in your scripting environment.

This is also the case for applets. In Netscape navigator 3 or greater and in Microsoft Internet Explorer 4 or greater you can simply invoke the public methods of an applet.

Before you can invoke a method on an applet, you need to give your applet a name. If not, the browser won't know which applet you are referring to.

In the following examples we will use javascript because both browsers support it.

methods & types

public methods

- addRecords (tablename,records)
- removeRecords (tablename, records)
- getKeys (tablename, separator) returns keys
- getField (tablename, key, column index) returns value
- setField (tablename, key, column index, value)
- getSelected () returns key
- setSelected (key)

parameter and return types

- tablename : string
- records: string
- key: string
- keys: string
- column index: integer
- value: string
- separator: string

With these methods you can add/replace/remove records in the following tables:

- entries

- options
- links
- fonts
- images
- internalactions
- externalactions
- selected
- expanded
- contracted

dipu applets are thread safe, so multiple concurrent accesses to the applet are allowed.

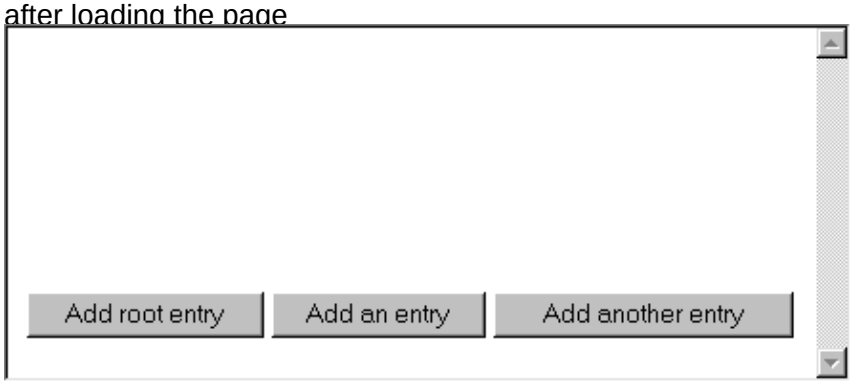
adding or replacing records

With the addRecords(tablename,records) method you can add record(s) to tha applet in the specified table. If there is already a record with the specified key, then the new record will replace the old record.

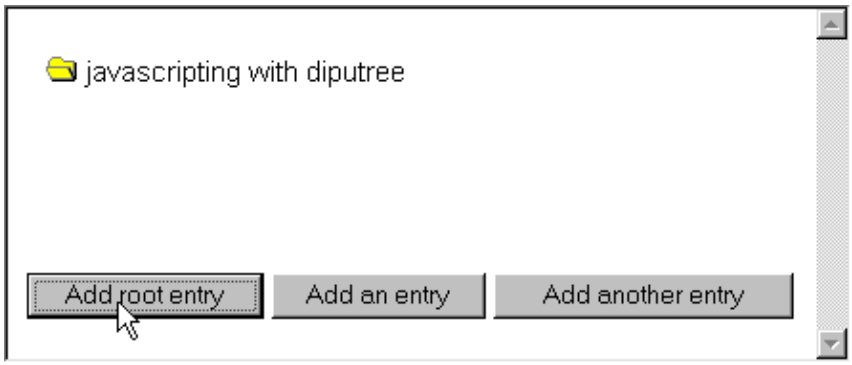
example

```
<html>
<body>
<applet CODE="diputree.class" ARCHIVE="diputree.jar" CODEBASE="../../../"
NAME="diputree" WIDTH="300" HEIGHT="100" MAYSCRIPT>
</applet>
<form>
<input TYPE="BUTTON" VALUE="Add root entry"
onClick="diputree.addRecords('entries','^root|javascripting with
diputree|-start-|c|o');">
<input TYPE="BUTTON" VALUE="Add an entry"
onClick="diputree.addRecords('entries','^one|an entry|root');">
<input TYPE="BUTTON" VALUE="Add another entry"
onClick="diputree.addRecords('entries','^two|another entry|root')">
</form>
</body>
</html>
```

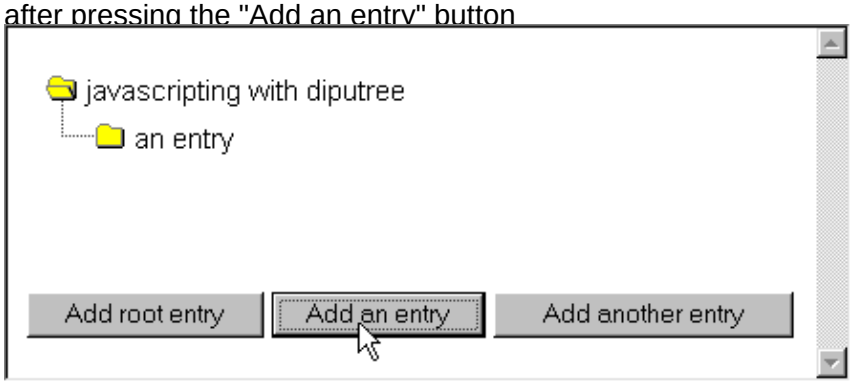
after loading the page



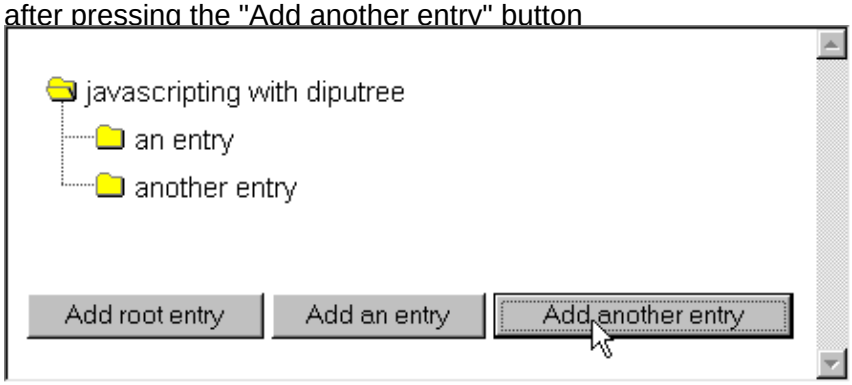
after pressing the "Add root entry" button



after pressing the "Add an entry" button



after pressing the "Add another entry" button



removing records

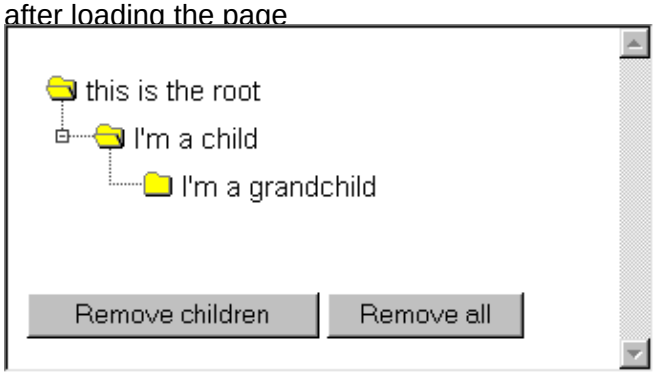
With the removeRecords(tablename,records) method you can remove record(s) from the applet in the specified table.

Example

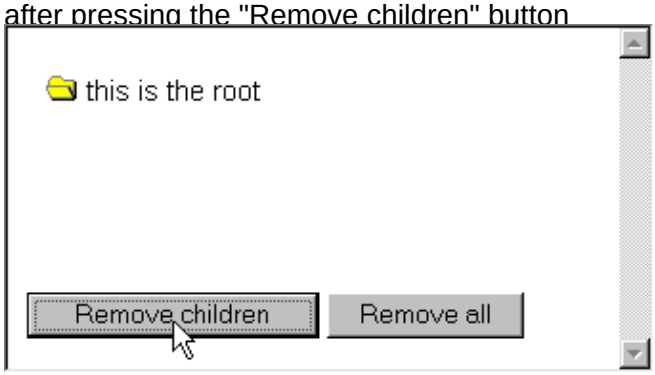
```
<html>
<body>
<applet CODE="diputree.class" ARCHIVE="diputree.jar" NAME="diputree"
WIDTH="300" HEIGHT="100" MAYSCRIPT>
<param name="entries"
value="
^root|this is the root|-start-|c|o
^child|I'm a child|root|c|o
^grandchild|I'm a grandchild|child
```

```
">
</applet>
<form>
<input TYPE="BUTTON" VALUE="Remove children"
onClick="diputree.removeRecords('entries','^child');">
<input TYPE="BUTTON" VALUE="Remove all"
onClick="diputree.removeRecords('entries','^root');">
</form>
</body>
</html>
```

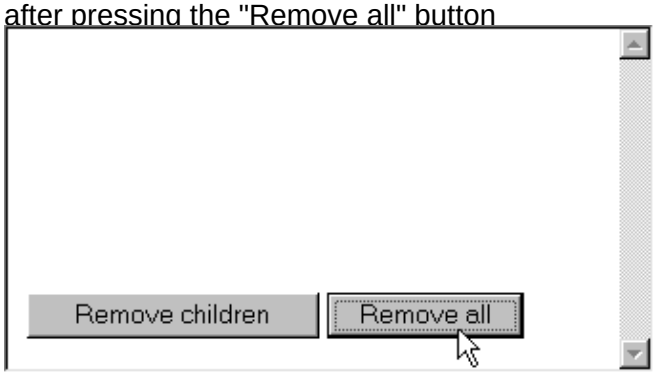
after loading the page



after pressing the "Remove children" button



after pressing the "Remove all" button



getting the key's from a table

With the getKeys(tablename, separator) method you can retrieve all the keys who are present in a table.

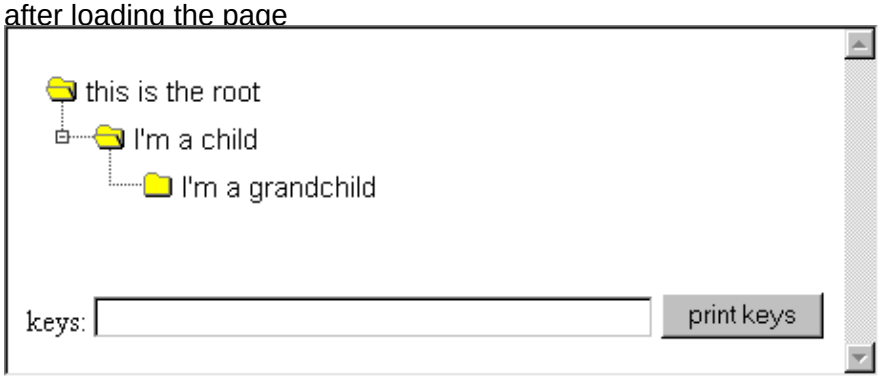
array String.split (separator)

The javascript function split operates on a String value, takes a separator as an argument and returns an Array. This gives you an easy way to convert the result string from the getKeys function to an Array.

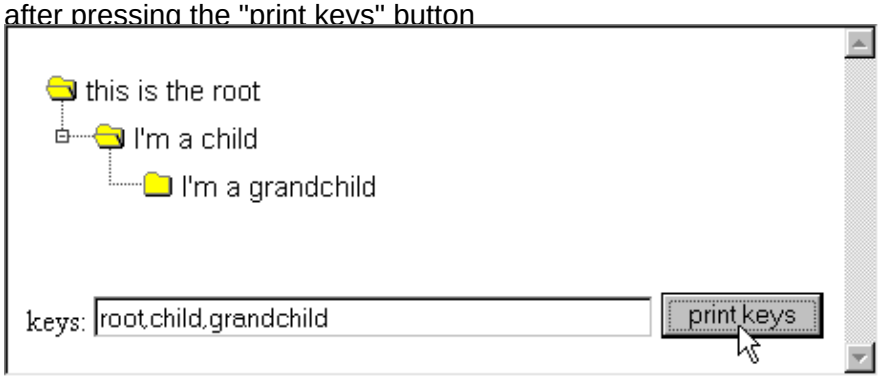
example

```
<html>
<body>
<applet CODE="diputree.class" ARCHIVE="diputree.jar" NAME="diputree"
WIDTH="300" HEIGHT="100" MAYSCRIPT>
<param name="entries"
value="
^root|this is the root|-start-|c|o
^child|I'm a child|root|c|o
^grandchild|I'm a grandchild|child
">
</applet>
<form name="main">
<p>keys: <input type="text" name="keys" size="40"> <input TYPE="BUTTON"
VALUE="print keys"
onClick="main.keys.value = diputree.getKeys('entries',',');"> </p>
</form>
</body>
</html>
```

after loading the page



after pressing the "print keys" button



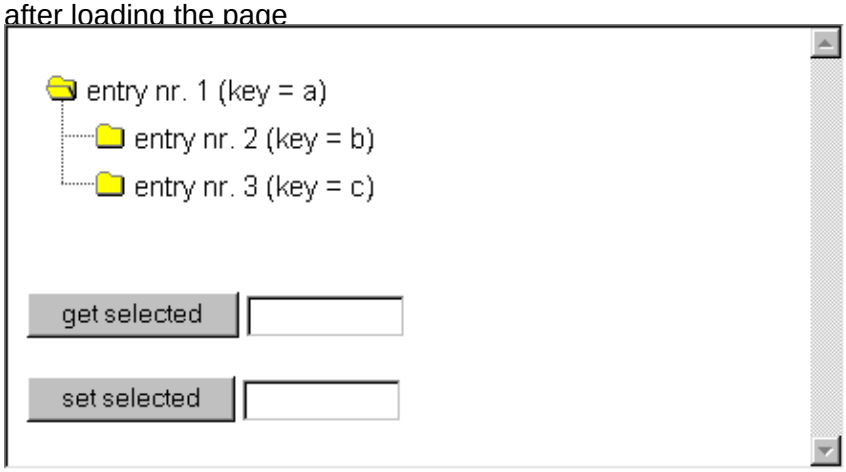
getting and setting the selected entry

With the getSelected() method you can query the applet for the selected tab. With the setSelected (key) you can programmatically set the selected tab.

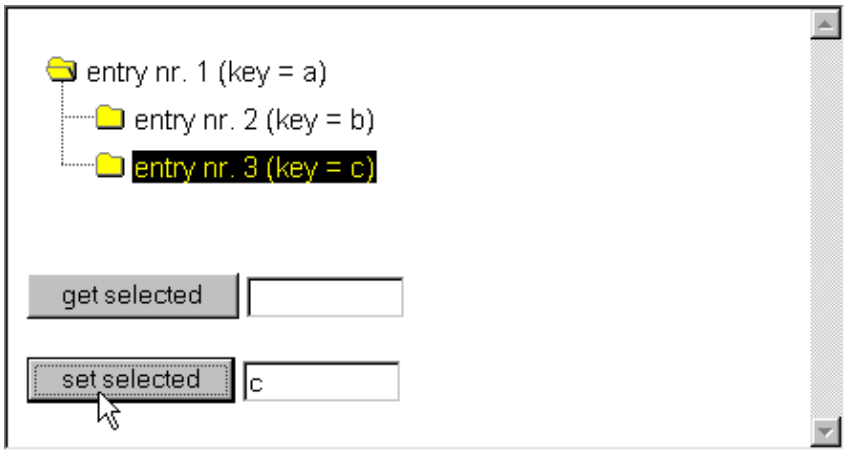
Example

```
<html>
<body>
<applet CODE="diputree.class" ARCHIVE="diputree.jar" NAME="diputree"
WIDTH="300" HEIGHT="100" MAYSCRIPT>
<param name="entries"
value="
^a|entry nr. 1 (key = a)
^b|entry nr. 2 (key = b)|a
^c|entry nr. 3 (key = c)|a
">
</applet>
 <form name="main">
<p><input TYPE="BUTTON" VALUE="get selected"
onClick="main.getselected.value = diputree.getSelected();">
<input type="text" name="getselected" size="10"></p>
 <p><input TYPE="BUTTON" VALUE="set selected"
onClick="diputree.setSelected(main.setselected.value);">
<input type="text" name="setselected" size="10"></p>
</form>
</body>
</html>
```

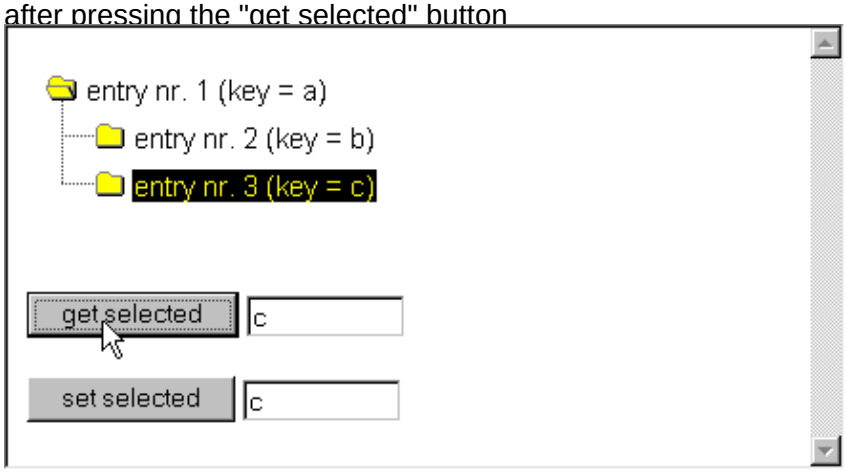
after loading the page



after filling in c in the text field
and pressing the "set selected" button



after pressing the "get selected" button



getting and setting individual fields in a record

With the getField(tablename, key, column index) method you can fetch any field value from a record. With the setField (tablename, key, column index, value) method you can set any field value in a record

Now every property of the applet is customizable.

exception

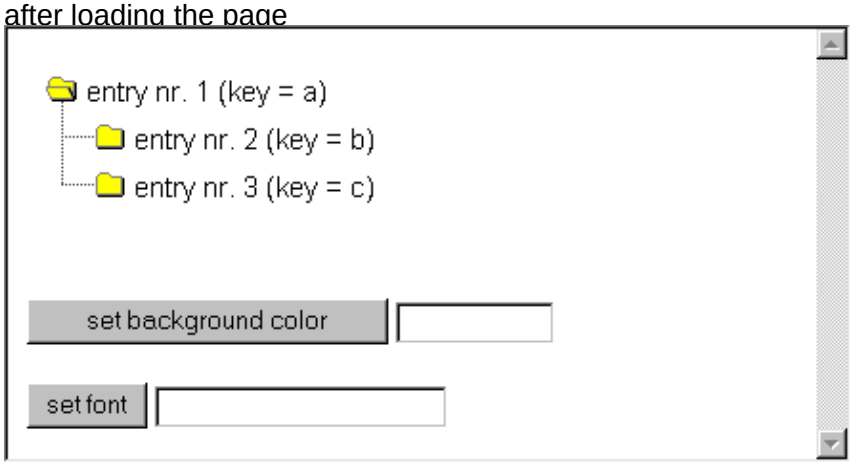
You can not edit individual fields in the images table. Just use addRecords(tablename,records) to add new images to the applet.

examples

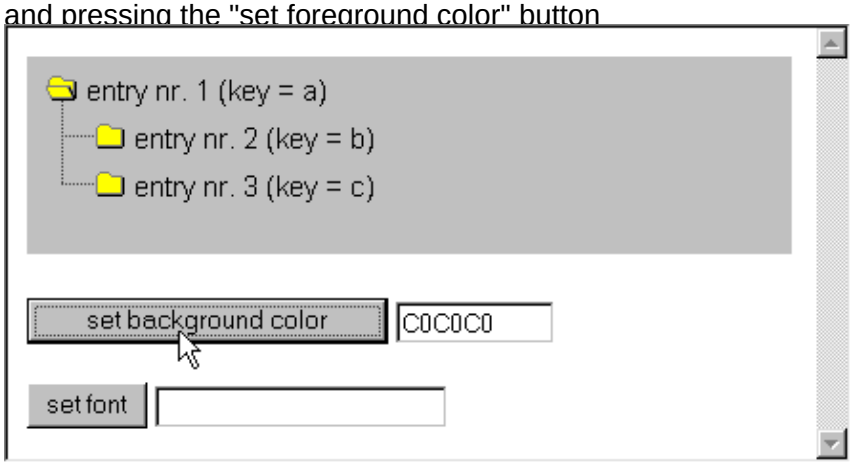
```
<html>
<body>
<applet CODE="diputree.class" ARCHIVE="diputree.jar" NAME="diputree"
```

```
WIDTH="300" HEIGHT="100" MAYSCRIPT>
<param name="entries" value="
^a|entry nr. 1
^b|entry nr. 2|a
^c|entry nr. 3|a
">
</applet>
<form name="main">
<input TYPE="BUTTON" VALUE="set background color"
onClick="diputree.setField('options','backgroundcolor',1,main.bgcolor.value);">
<input type="text" name="bgcolor" size="10">
<input TYPE="BUTTON" VALUE="set font"
onClick="diputree.setField('fonts','defaultfont',1,main.font.value);">
<input type="text" name="font" size="20"></p>
</form>
</body>
</html>
```

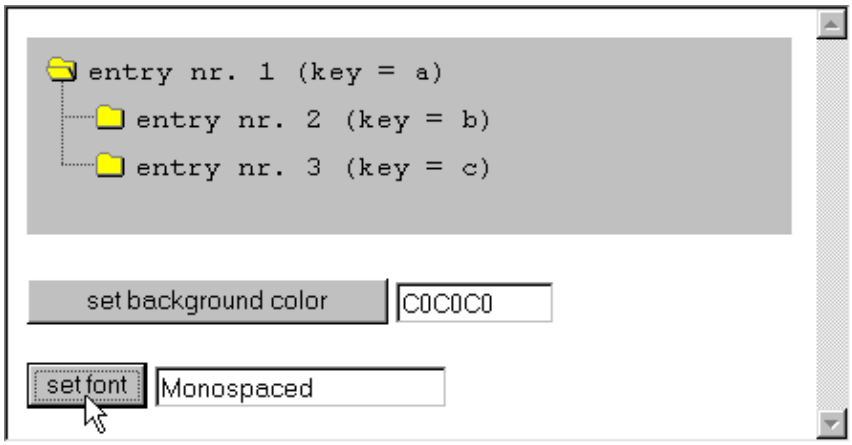
after loading the page



after filling in C0C0C0 in the text field
and pressing the "set foreground color" button



after filling in Monospaced in the text field
and pressing the "set font" button


entry nr. 1 (key = a)
entry nr. 2 (key = b)
entry nr. 3 (key = c)
set background color
C0C0C0
set font
Monospaced

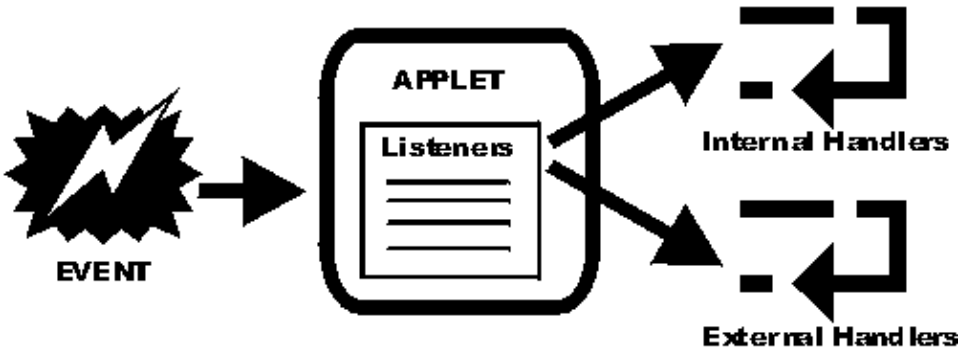
3. The events framework

introduction

The event framework permits script writers to respond to events that happen inside an applet.

the life of an event

- the Java Environment will notice that the user has performed an action (e.g. a mouse click).
- the Java Environment will create an event describing the user's action
- the Java Environment will send this action event to the applet
- The applet will check if there are listeners listening to this specific event
- If there is a listener associated with this action, then the applet will execute the handler asscociated with this listener



event listeners

An event listener is a mapping between an event and a handler to execute when this event occurs. The listeners table is a Map with events as keys and methods as values (duplicate keys are however allowed).

diputree has three types of listeners:

- the "selected" listeners.
- the "expanded" listeners.
- the "contracted" listeners

listeners are single shot by default

A listener will remove himself from the listeners map after he has responded to an event he listened for but before he has executed the handler.

The design rationale for this behavior is that most handlers are single shot and that it would be to cumbersome to force the programmer to remove every listener after it was being used.

You can re-register a listener during the execution of his handler, effectively removing the single shot behavior, or use permanent listeners.

permanent listeners

If you prefix the key of the handler in the listeners table with two plus signs "++", then the listener will not remove its self from the listeners table.

selected

selected is the listeners map that listens for mouse click events on tree entries.

expanded

expanded is the listeners map that listens for mouse click events or keyboard actions that expands containers (nodes).

contracted

contracted is the listeners map that listens for mouse click events or keyboard actions that contract container(nodes).

event handlers

precedence rules

If both internal and external event handlers are defined, then the external event handlers have precedence over internal event handlers. This rule makes it easy to provide for 'fall through' event handling, when external handlers are not available.

internal handlers

Internal handlers are predefined in diputree. There is no external intervention necessary to execute this handlers. The applet just invokes the method with the parameters defined in the internal handler record.

These type of handlers are ALWAYS possible, because the applet takes care of them. So you can always count on them to occur.

external handlers

In scripting you saw how the applet's methods could be invoked from scripting languages. Now we'll see how the applet can invoke script methods through external handlers.

Think of them as call back function from the applet back to the script.

browser support

There are some conditions that must be met before external handlers can take place.

- Only certain browsers support call backs from applets (notably netscape3+ and explorer4.01+)
- the "MAYSCRIPT" option must set in the applet definition.

The internalhandlers table

This is a 'database table' parameter. Please refer to passing database tables for a detailed explanation of database tables layout. In all the following examples we assume that the recordseparator is ^ and the fieldseparator is | .

Internal handlers can perform different tasks. The task defines the value and the type of the different fields.

adding records from an external file to a table

task = 0

Record definition				
Field	**Name**	**Type**	**Default**	**Range**
0	key	string	no default, obligatory field	
1	task	decimal integer	no default, obligatory field	should be 0 for adding records from an external file
2	table	string	no default, obligatory field	the name of the table where the new rows should be added to
3	URL	string	no default, obligatory field	the URL of the external file where the new fields can be found

- Field 0 is the (primary) key. This key will be used when referring to this record
- Field 1 is the internal task to be perfomed
- Field 2 defines the table on which to perform the task
- Field 3 defines the URL of the external file where the new fields are located. For a detailed explanation of external files refer to Using External Files in the parameter chapter.

adding a record to a table

task = 1

Record definition				
Field	**Name**	**Type**	**Default**	**Range**
0	key	string	no default, obligatory field	

1	task	decimal integer	no default, obligatory field	should be 1 for adding a record
2	table	string	no default, obligatory field	the name of the table where the new record should be added to
3 and further	record	record		see specific table

- Field 0 is the (primary) key. This key will be used when referring to this record
- Field 1 is the internal task to be perfomed
- Field 2 defines the table on which to perform the task
- Field 3 and the following Fields define the record that should be added to the table. Please refer to the specific tables for the description of the fields.

setting a field in a record

task = 2

Record definition				
Field	**Name**	**Type**	**Default**	**Range**
0	key	string	no default, obligatory field	
1	task	decimal integer	no default, obligatory field	should be 2 for setting a field in a record
2	table	string	no default, obligatory field	the name of the table where the field should be set
3	key (to set)	string	no default, obligatory field	
4	value (to set)	depends on destination field	no default, obligatory field	
5	field index	integer	default = 1	index 0 is the key !

◊ Field 0 is the (primary) key. This key will be used when referring to this record
◊ Field 1 is the internal task to be perfomed
◊ Field 2 defines the table on which to perform the task
◊ Field 3 the key of the record that should be changed

◊ Field 4 the value that should be set
◊ Field 5 the index of the field that should be changed

remove a record in a table

task = 3

Record definition				
Field	**Name**	**Type**	**Default**	**Range**
0	key	string	no default, obligatory field	
1	task	decimal integer	no default, obligatory field	should be 3 for removing records
2	table	string	no default, obligatory field	the name of the table where record should be removed from
3	key (to delete)	string	no default, obligatory field	

- Field 0 is the (primary) key. This key will be used when referring to this record
- Field 1 is the internal task to be perfomed
- Field 2 defines the table on which to perform the task
- Field 3 the key of the record that should be deleted

set the selected key

task = 4

Record definition				
Field	**Name**	**Type**	**Default**	**Range**
0	key	string	no default, obligatory field	
1	task	decimal integer	no default, obligatory field	should be 0 for loading
2	table	string	should be blank	
3	key (to be selected)	string	no default, obligatory field	

- Field 0 is the (primary) key. This key will be used when referring to this record

- Field 1 is the internal task to be perfomed
- Field 2 should be left blank
- Field 3 the key of the record that should be set as the selected one

The externalhandlers table

This is a 'database table' parameter. Please refer to passing database tables for a detailed explanation of database tables layout. In all the following examples we assume that the recordseparator is ^ and the fieldseparator is | .

Record definition				
Field	**Name**	**Type**	**Default**	**Range**
0	key	string	no default, obligatory field	
1	script	string	no default, obligatory field	script code

- Field 0 is the (primary) key. This key will be used when referring to this record
- Field 1 is the script code to be executed

script code & context

External handlers contain script code. The scope of the code being executed is the same as the context of the applet.

So just imagine that the applet is replaced by the script code, to determine the context of your script code.

The selected table

This is a 'database table' parameter. Please refer to passing database_tables for a detailed explanation of database tables layout. In all the following examples we assume that the recordseparator is ^ and the fieldseparator is | .

Record definition				
Field	**Name**	**Type**	**Default**	**Range**
0	key	string	no default, obligatory field	
1	handler	string	no default, obligatory field	the key of an internal and/or external handler

- Field 0 is the (primary) key. This key will be used when referring to this record
- Field 1 is the script code to be executed

listening for and responding to selected events

A selected event will occur when the user selects an entry to be the selected one. It is possible that the entry is already the selected one.

Due to the precedence rule the externalhandlers table shall be searched before the interhandlers table is searched. If a match is found in the externalhandlers table, then the internalhandlers table will not be searched.

Multiple records are possible, but order of execution is undefined.

The expanded table

This is a 'database table' parameter. Please refer to passing database tables for a detailed explanation of database tables layout. In all the following examples we assume that the recordseparator is ^ and the fieldseparator is | .

Record definition				
Field	**Name**	**Type**	**Default**	**Range**
0	key	string	no default, obligatory field	
1	handler	string	no default, obligatory field	the key of an internal and/or external handler

- Field 0 is the (primary) key. This key will be used when referring to this record
- Field 1 is the script code to be executed

listening for and responding to expanded events

An expanded event will occur when the user expands a container (node).

Due to the precedence rule the externalhandlers table shall be searched before the interhandlers table is searched. If a match is found in the externalhandlers table, then the internalhandlers table will not be searched.

Multiple records are possible, but order of execution is undefined.

The contracted table

This is a 'database table' parameter. Please refer to passing database_tables for a detailed explanation of database tables layout. In all the following examples we assume that the recordseparator is ^ and the fieldseparator is | .

Record definition				
Field	**Name**	**Type**	**Default**	**Range**
0	key	string	no default, obligatory field	
1	handler	string	no default, obligatory field	the key of an internal and/or external handler

- Field 0 is the (primary) key. This key will be used when referring to this record
- Field 1 is the script code to be executed

listening for and responding to contracted events

A contracted event will occur when the user contracts a container (node).

Due to the precedence rule the externalhandlers table shall be searched before the interhandlers table is searched. If a match is found in the externalhandlers table, then the internalhandlers table will not be searched.

Multiple records are possible, but order of execution is undefined.

examples

Internal Handlers

loading from an external file

- The applet parameters
  <param name="selected" value="
  ^a|load
  ">
  <param name="internalhandlers"
  value="
  ^load|0|entries|examples/documentation/events/entries.txt
  ">
  <param name="entries" value="
  ^a|node a
  ">
- The entries.txt file
  ^b|node b
  ^c|node c

adding a record

<param name="selected" value="
^a|adding
">

<param name="internalhandlers" value="
^adding|1|entries|b|node b was added !
">

<param name="entries" value="
^a|node a
">

setting a field in a record

<param name="selected" value="
^a|change-text
">
<param name="internalhandlers" value="
^change-text|2|entries|a|I changed !!!
">
<param name="entries" value="
^a|node a
">

remove a record

```
<param name="selected" value="
^a|remove
">
<param name="internalhandlers" value="
^remove|3|entries|b
">
<param name="entries" value="
^a|node a
^b|node b
">
```

set selected (repeated)

```
<param name="selected" value="
^a|++select
">
<param name="internalhandlers" value="
^select|4||c
">
<param name="entries" value="
^a|node a
^b|node b
^c|node c
">
```

External Handlers

using pop-up messages

```
<param name="options" value="
^externalhandlers|1
">

<param name="selected" value="
^a|++popa
^b|++popb
">

<param name="externalhandlers"
value="
^popa|alert ('you clicked node nr. 1')
^popb|alert ('you clicked node nr. 2')
">

<param name="entries" value="
^a|node nr. 1
^b|node nr. 2
```

">

4. The configurator

introduction

Until now you had to go straight to metal and edit the html code line by line.

No more!

The configurator is a graphical user interface (GUI) tool to configure the applet and its parameters. It takes the burden out of configuring the dipu applets.

Your projects will be less error prone, due to the fact that

- all option names are hard wired in (and can't for this reason be misspelled)
- special choosers assist you in choosing a value (e.g. the color chooser)
- external key reference integrity (you can choose your key from the already defined keys)

And last but not least, you don't need any html experience to make your applet work !

requirements

Although dipuree will work on any java browser (jdk1.02), the configurator requires java 2.0

(jdk1.2+).
You can freely download the java runtime environment (jre) from the sun's jre website http://java.sun.com/products/jdk/1.2/jre/ or from sun's website http://java.sun.com .

preparing your project

The configurator configures your web page completely, but it does not copy the required .class and .jar files to the destination. So before you can test your web page you need to copy those file to the required directory.

These files are:

- diputree.class
- diputree.jar

Just copy those two files to the directory where you will be using your applet.

diputree.jar

diputree.jar is just the diputree.class file but compressed.

For compatibilty reasons it is best to copy both those files to the destination directory.
If the browser supports archive (jar) files, then the jar file will be used instead of the uncompressed class file, saving you a lot of download time over slow links.

starting the configurator

- from the command line type "java configurator.jar"
- from a windowing environment, just double-click on the "configurator.jar" file.
  (this can be operating system dependent and requires system support)

Both methods require a java2 compatible environment to be present and correctly configured.

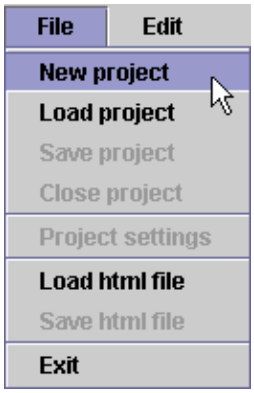
Projects

The configurator always uses a project. If you don't explicitly create or load a project, then the configurator will create a default project.

Projects contain all the information concerning the location of the files, the browsers to use when previewing and other meta-data.

create a new Project

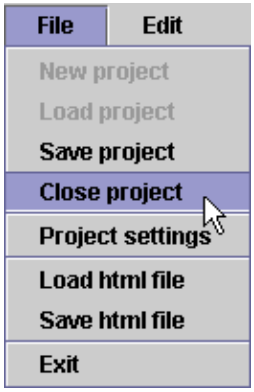
You can create a new project when the configurator is just started or when you closed the current project.



1. In the "File" Menu select "New Project" or press  in the tool bar.

close a Project

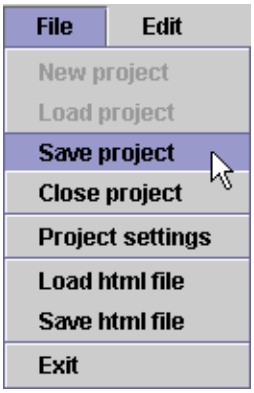
You can close the current project. This will return the configurator to the initial state, allowing you to create or open a new project.



1. In the "File" Menu select "Close Project".

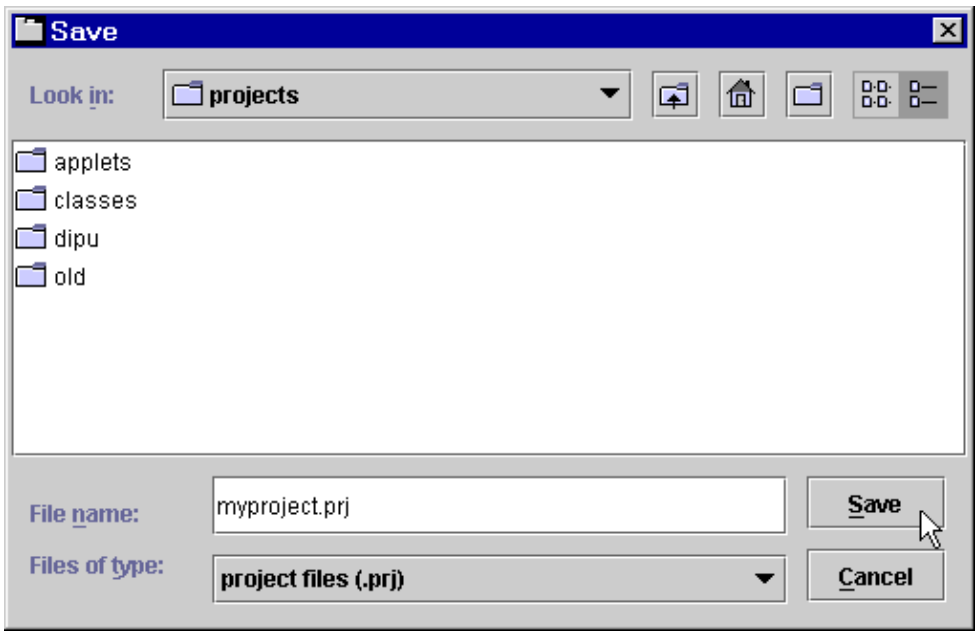
save a Project

You can save the current project to disk. The preferred extension is .prj but you can use any extension you want.



1. In the "File" Menu select "Save Project" or press in the tool bar.

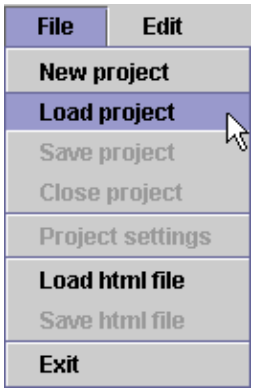
2. Give the project a name and select save



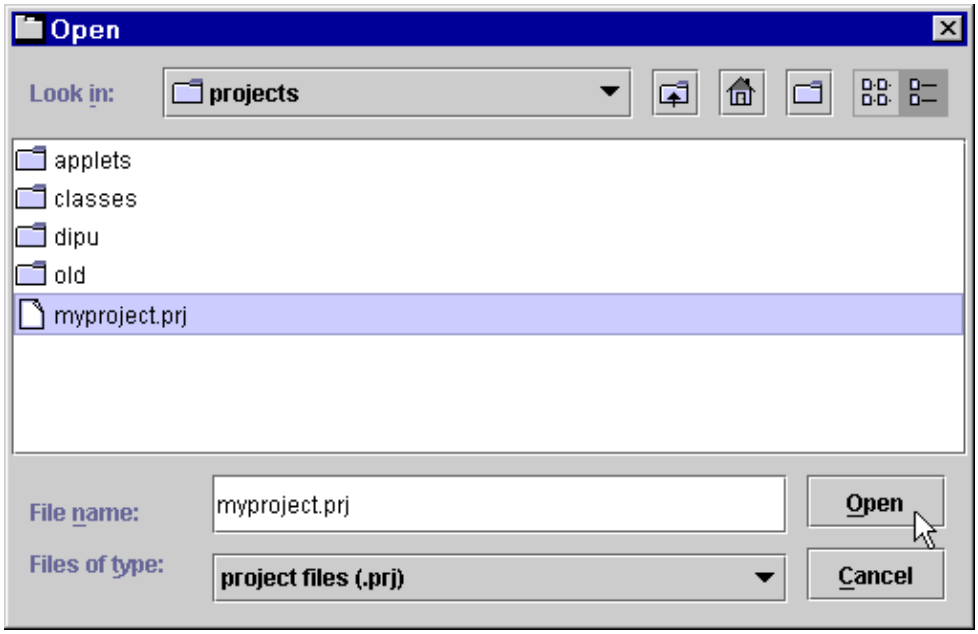
load a Project

You can load a previously saved project when the configurator is just started or when you closed the current project.

1. In the "File" Menu select "Load Project" or press in the tool bar.

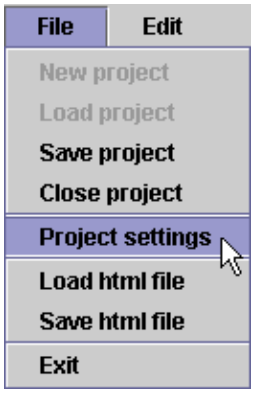


2. Select the project and press load.



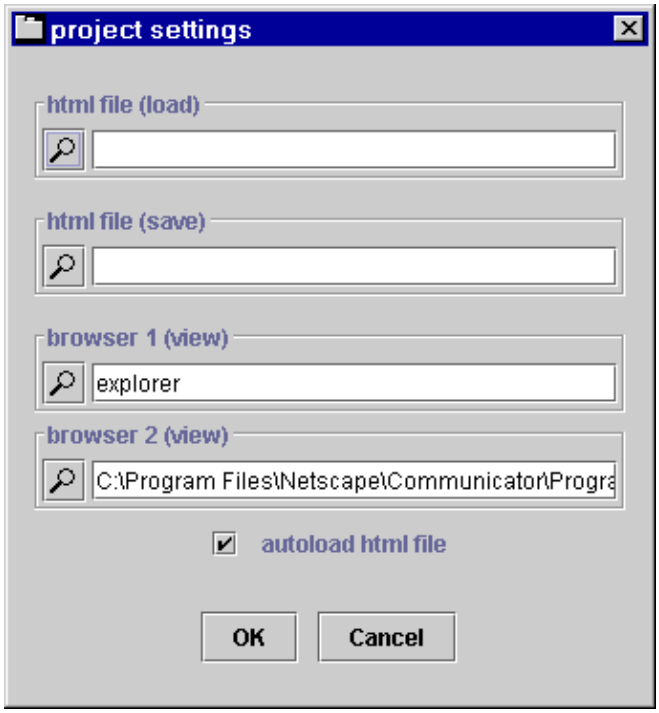
project Settings

You can edit the current project settings. This allows you to tailor the project to your needs.



1. In the "File" Menu select "Project Settings".

2. Edit the fields you want to edit. Press "ok" or "cancel".



key	**value**	
value	**description**	
html file (load)	string	the html file where all tables will be loaded from
html file (save)	string	the html file where all tables will be written to
browser 1	string	a browser used to preview the applet
browser 2	string	a browser used to preview the applet
autoload html file	boolean integer	defines automatical loading of the html file when the project is opened (unchecked = don't autoload , checked = autoload)

Html files

The html files are the actual 'containers' of the settings.

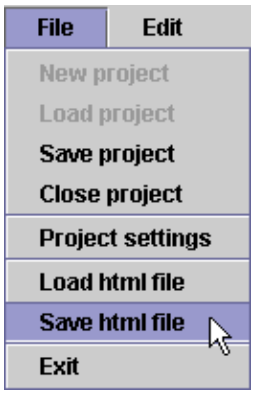
How the configurator works with html files

- it will create an 'in memory' default html page when a new project is created (you MUST use the "Save html file" to make these settings persistent)
- it can load a html file and extract all the settings from it
- it can save the settings to a html file

Saving a project does not ensure that your html file is saved. You always need to save your html file!

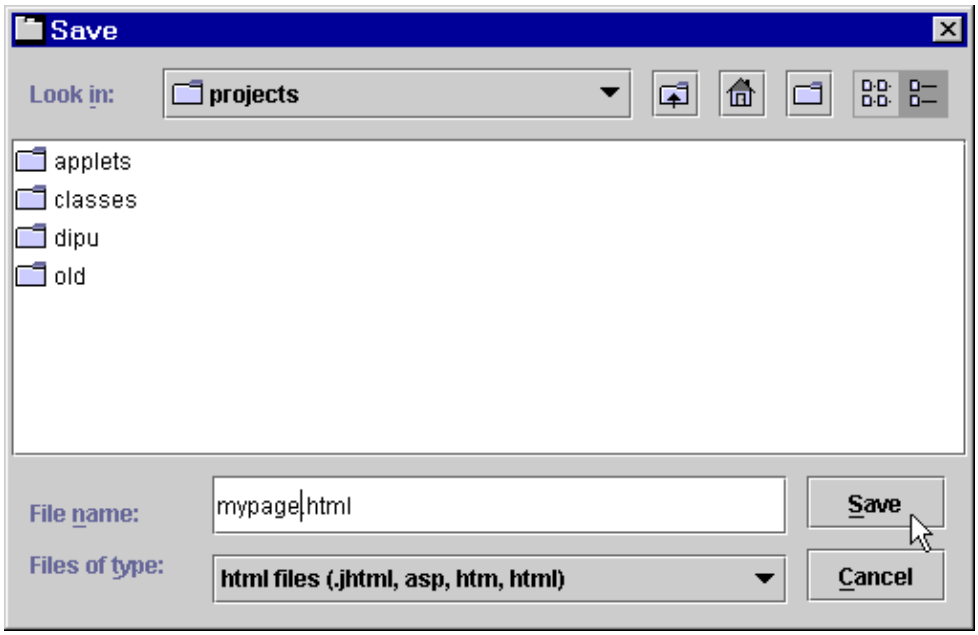
save a html file

This will apply the new settings to the html file. Use "Save html file" to make your changes persistent.



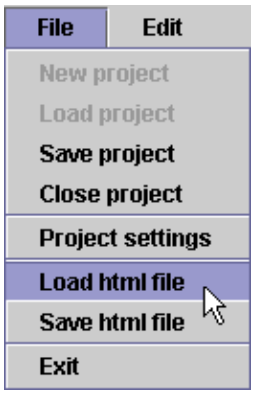
1. In the "File" Menu select "Save html file" or press  in the tool bar.

2. Give the html file a name and select save



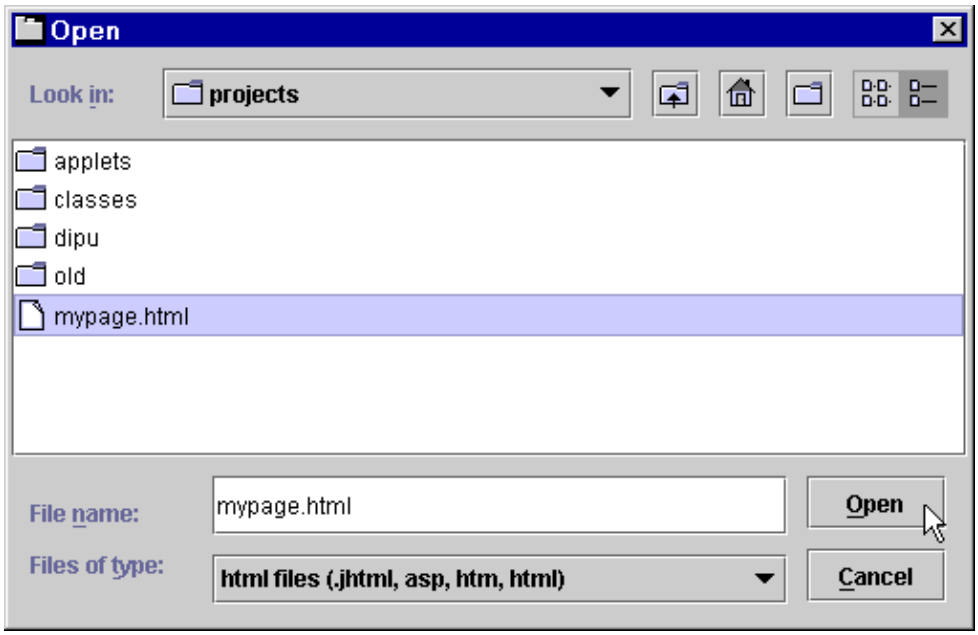
load a html file

This will extract the setting from the loaded html file.



1. In the "File" Menu select "Load html file" or press  in the tool bar..

2. Select the html file and select open


Open
Look in:
projects
applets
classes
dipu
old
mypage.html
File name:
mypage.html
Open
Files of type:
html files (.jhtml, asp, htm, html)
Cancel

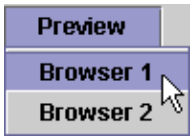
Previewing

After you applied changes you can preview the applet in a browser. The configurator allows you to define two browsers to preview your applet.

The configurator saves all changes to the html file before previewing the applet.

IMPORTANT:

The configurator does not create a temporary file! He applies the changes to the last saved file.
If you don't want this behaviour you can first save to a temporary file and then select preview. Now the configurator will use this file for previewing

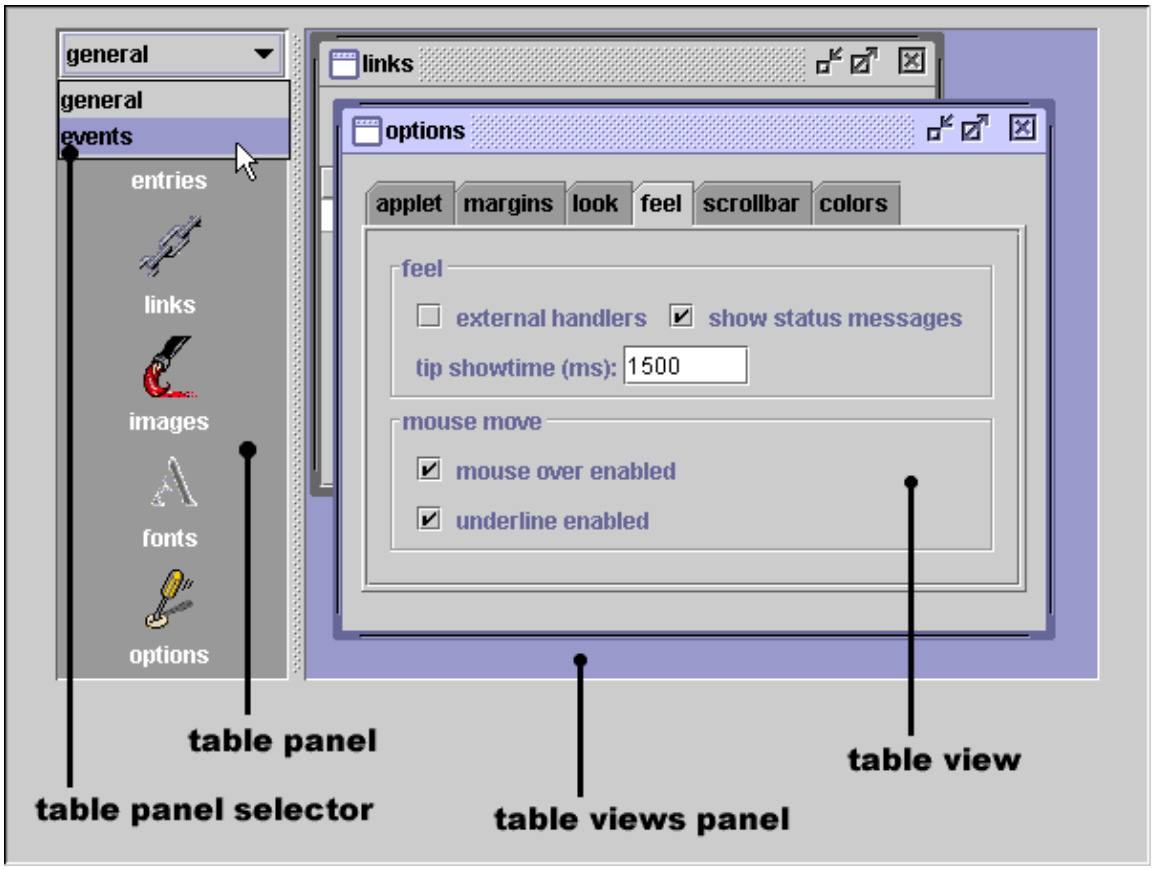


- In the "Preview" Menu select "Browser 1" or "Browser 2"
- Click in the toolbar on  for previewing in browser 1 or on  for previewing in browser 2.

Tables

Please refer to the tables chapter for a detailed description of all fields used in the tables.

the workspace



the table panel

The table panels contain all the tables you can edit. You can switch between the table panels by clicking on the table panel selector.

the table views panel

The table views panel contains the views of the table. Every table view has its own window.

These table views allow you to edit the fields contained in the tables.

selecting a table

You can select a table

- by clicking on it in the tables menu
- by clicking on its icon in the table panel

the table menu

The "tables" menu contains all the tables available to edit.

the table panel

Select one of the following icons to open the corresponding table view in the table views panel

General tables

entries links images fonts options

Event tables

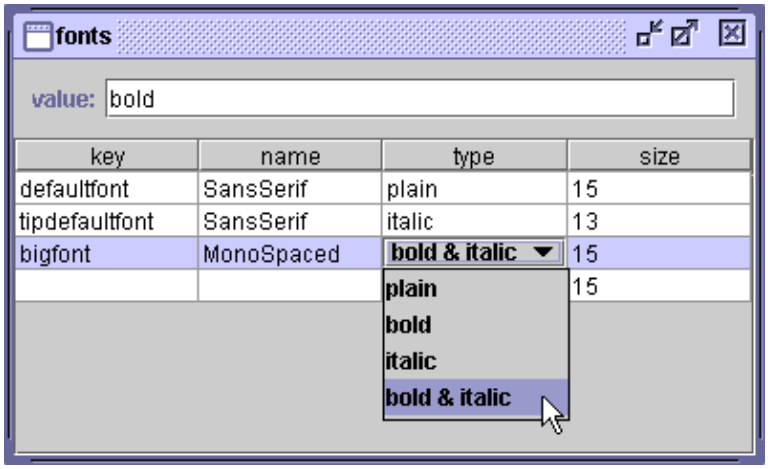
internal handlers external handlers selected expanded contracted

editing a table

in cell editing

You can click on a cell and start editing in the cell.

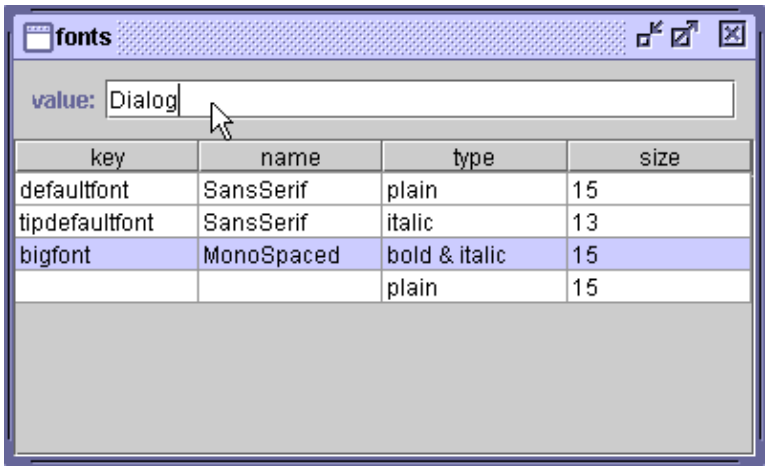
If the cell is an external key or the possible values are limited and known then a combo box is presented. This will prevent you from making typing errors and keep referential integrity among tables.



line editing

Sometimes not all the information is visible in a cell and editing can become cumbersome. Therefor is the cell value displayed in the line at the top of the view.

Changes made in this line will be made permanent after the enter key is pressed. This is also a way to override the combo box style of cell editing when using external keys.



Appendix

legal notices

license

Limited License Grant.

Dipu grants to you ("Licensee") a nonexclusive, nontransferable, worldwide, royalty-free license to use this binary version of the diputree applet (the "Software").
The Licensee has the right to use the software according to the appropriate version he obtained. When referred to a site, a site means a location where all the web pages share a single fully qualified domain name.

Restrictions

Software may not be transferred, leased, assigned, or sublicensed, in whole or in part.

IF THESE RESTRICTIONS ARE NOT ALLOWED BY YOUR RESPECTIVE LAW THEN DON'T USE THIS SOFTWARE.

CONTACT DIPU TO CHECK IF THE RESTRICTIONS CAN BE RELAXED IN YOUR CASE. THE RELAXATION CAN ONLY BE GIVEN BY WRITTEN CONSENT.

disclaimer of warranty.

The Software is provided "AS IS," without a warranty of any kind.
ALL EXPRESS OR IMPLIED REPRESENTATIONS AND WARRANTIES,
INCLUDING ANY IMPLIED WARRANTY OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT,
ARE HEREBY EXCLUDED.

IF THIS DISCLAIMER IS NOT ALLOWED BY YOUR RESPECTIVE LAW THEN DON'T USE THIS SOFTWARE.

CONTACT DIPU TO CHECK IF THE DISCLAIMER CAN BE RELAXED IN YOUR CASE. THE RELAXATION CAN ONLY BE GIVEN BY WRITTEN CONSENT.

limitation of liability.

IN NO EVENT WILL DIPU BE LIABLE FOR ANY LOST REVENUE, PROFIT OR DATA, OR FOR DIRECT, INDIRECT, SPECIAL, CONSEQUENTIAL, INCIDENTAL OR PUNITIVE DAMAGES HOWEVER CAUSED AND REGARDLESS OF THEORY OF LIABILITY ARISING OUT OF THE USE OF, INABILITY TO USE OR DOWNLOADING OF THE SOFTWARE, EVEN IF DIPU HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

IF THIS LIMITATION IS NOT ALLOWED BY YOUR RESPECTIVE LAW THEN DON'T USE THIS SOFTWARE.

CONTACT DIPU TO CHECK IF THE LIMITATION CAN BE RELAXED IN YOUR CASE. THE RELAXATION CAN ONLY BE GIVEN BY WRITTEN CONSENT.

the evaluation version.

The Licensee has the right to evaluate the Software for a period of 30 days. After this period the licensee is obliged to purchase the Software or destroy every copy of the Software the Licensee has.

the free version.

The Licensee has the right to use the software at no cost if
ALL the following conditions are met.

- (a) if it is used for NON-COMMERCIAL USE
  - every use that has no direct or indirect commercial use.
- (b) if it is used by PRIVATE PERSONS
  - the free version only applies if used by individuals (physical persons).

BOTH CONDITIONS MUST BE MET AT THE SAME TIME

The internet version.

The Licensee has the right to use the software at no cost if
ALL the following conditions are met.

- (a) for use on only 1 web page server accessible through 1 IP address (virtual or actual)
- (b) accessible from the internet and not for internal use

BOTH CONDITIONS MUST BE MET AT THE SAME TIME

The intranet/extranet version.

The Licensee has the right to use the software at no cost if
ALL the following conditions are met.

- (a) for use on only 1 web page server accessible through 1 IP address (virtual or actual)
- (b) accessible only for a clearly defined audience and not for the general internet

BOTH CONDITIONS MUST BE MET AT THE SAME TIME

The OEM version.

The Licensee has the right to use the software at no cost if
ALL the following conditions are met.

- (a) to redistribute the class and jar files for 1 clearly defined project.
- (b) internet license conditions

  BOTH CONDITIONS MUST BE MET AT THE SAME TIME

Copyright

USE OR DATA, INTERRUPTION OF BUSINESS, OR FOR INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES OF ANY KIND, ARISING FROM ANY ERROR IN THIS DOCUMENTATION.

This means that the software and documentation come with no warranty, without any limitation. No liability of any kind can be accepted and this the fullest extent. If your law doesn't allow this, then do not use this software and documentation.

dipu bvba, Remerstraat 50, B-3128 Baal, Europe

Credits

We like to thank the following people

- Rob Duncan from Sun for his html parser
- William Bull from www.artillion.com for some of the icons
- Michael Sweet for HTMLDOC