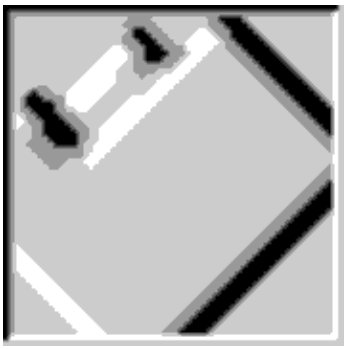


# diputab 2.2

# 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 of the dipu applets, you have total control over the applet’s representation and

functionality.

Functionality

- dipu applets is scriptable (in real time )
- tabs can be positioned at the top or the bottom of a page
- a ‘site wide’ consistent look an feel
- user definable tabs and icons
- configurable space between text, images and tabs
- configurable colors for text and background
- background images are now supported
- pop up tips are supported
- status bar information messages
- automatic expansion of tabs
- frame destination can be set for every individual link
- multiple frame destinations (unlimited)
- ‘opened’ and ‘closed’ icons
- fonts can be set for every individual entry
- standard tabs are compiled in and do not require any additional downloads
- configurable margins
- configurable indentation
- 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 diputab for more compatibility
  ( java's own scrollbar implementation is buggy )

## dipu applets have an extremely small footprint

Even with all this functionality it is only 12KB in jar format or 25Kb in class format,
NO ADDITIONAL DOWNLOADS ARE REQUIRED!

With a 33.6Kb modem and Netscape 4+ or ie4+ it only takes 3.5 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="diputab.class" ARCHIVE="diputab.jar" WIDTH=150
HEIGHT=250>

<PARAM name=recordseparator value="^">

<PARAM name=fieldseparator value="|">

<PARAM name=options value="
^backgroundcolor|C0C0C0
^foregroundcolor|C0C0C0
. . .
^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="diputab.class" ARCHIVE="diputab.jar" WIDTH=150 HEIGHT=250>
  <PARAM name="entries" value="url:http://www.mysite.com/mydirectory/entries.txt">

</APPLET>

- The external file (entries.txt)
  ^tab1|Hi, I'm tab one
  ^tab2|Hi, I'm tab two
  ^tab3|Hi, I'm tab three

## 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' 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 | .

Fields 2,3,4 and 5 are external keys who point to rows in other database tables. Field 2 is an external key for the links parameter, fields 3 and 4 are external keys for the images parameter and field 5 is an external key for the fonts parameter.

The entries table defines the structure of your tab.

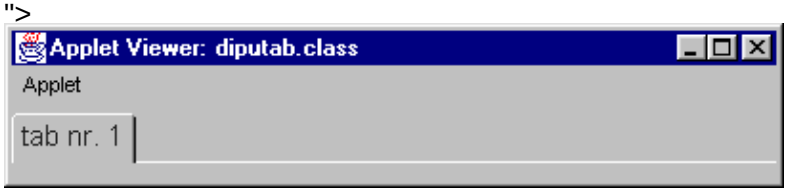
| **Record definition** | | | | |
|---|---|---|---|---|
| **field** | **Name** | **Type** | **Default** | **Range** |
| 0 | key | string | no default, obligatory field | |
| 1 | text | string | no default, obligatory field | |
| 2 | list of linksprofiles | space separated string(s) | | keys in the links table |
| 3 | name of the icon in the images parameter when tab is selected | string | | keys in the images table |
| 4 | name of the icon in the images parameter when tab is not selected | string | | keys in the images table |
| 5 | font profile | string | "defaultfont" | keys in the fonts table |
| 6 | tip/expansion (when tab is not selected) | string | | "auto" will enable auto expansion of containers |
| 7 | tip (when tab is selected) | string | | |
| 8 | status message (will be displayed on the browser's status bar) | string | | |

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

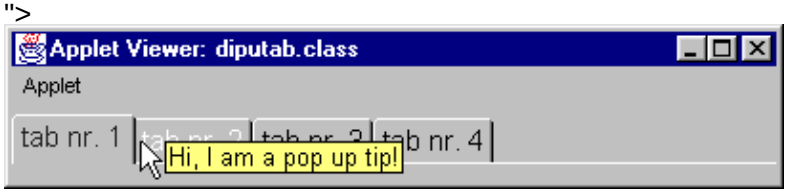
- field 1 is the text that will be displayed in your tab.
- field 2 defines the external link keys that should be activated when the user clicks on the entry. Multiple link keys are supported and should be separated with a space character. So clicking on an entry can trigger multiple links to different frames.
- field 3 contains the icon that will be displayed when the tab is not selected, field 4 when the tab is selected.
- field 5 defines the external font key used for the displayed text.
- field 6 and 7 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. field 6 will be displayed if the tab is not selected, field 7 when tab is selected.
  If field 6 contains the string "auto" then automatic expansion of tabs is activated.
- field 8 defines the status message that should be displayed if the pointing device is moved above this entry.

## examples

```
<PARAM name="entries" value="
^a|tab nr. 1
">
```




```
<PARAM name=entries value="
^a|tab nr. 1
^b|tab nr. 2|||||Hi, I am a pop up tip!
^c|tab nr. 3
^d|tab nr. 4
">
```



# 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. 2 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="
^a|home|dipuhome
^b|products|dipuproducts
^c|support|dipusupport
">
 <PARAM name=links value="
^dipuhome|http://www.dipu.com|dipuframe
^dipuproducts|http://www.dipu.com/products
^dipusupport|http://www.dipu.com/support
">
```

# 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. 3 and 4 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

## tab images

tableft, tabmiddle, tabright, tabline, selectedtableft, selectedtabmiddle and selectedtabright

You can define your own tabs. They are split up in 3 groups:

- non-selected tab (tableft, tabmiddle, tabright)
- baseline (tabline)
- selected tab (selectedtableft, selectedtabmiddle, selectedtabright)

### Tab composition

Combining the different parts of the tab together composes the tab.

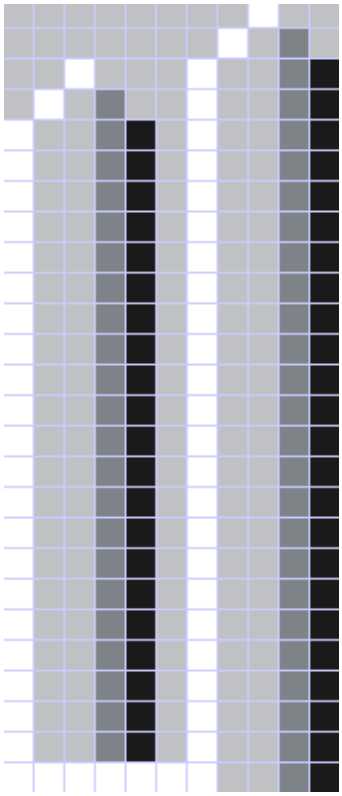
The left and right parts of a tab (either selected or not selected) are copied 'as is' from the image file.

tabmiddle, tabline and selectedtabmiddle are tiled as needed., e.g. when there are no more tabs to draw then tabline will be tiled until the whole applet area is filled.

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.

```
<param name="images"
value="
^tableft|images/tabtrans.gif|0|0|2|100
^tabmiddle|images/tabtrans.gif|2|0|1|100
^tabright|images/tabtrans.gif|3|0|2|100
^tabline|images/tabtrans.gif|5|0|1|100
^selectedtableft|images/tabtrans.gif|6|0|2|100
^selectedtabmiddle|images/tabtrans.gif|8|0|1|100
^selectedtabright|images/tabtrans.gif|9|0|2|100
"
```

Here is a blown up view of the tabtrans.gif image



If you make your tabs opaque then the background will not be visible but overlaid with the tabs.

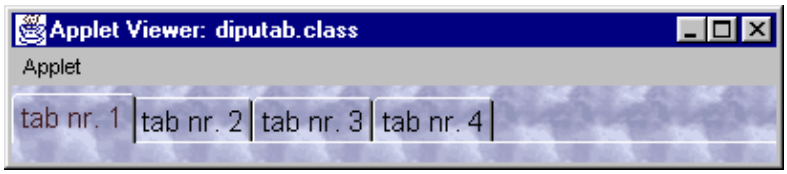
But if you use transparency in your tabs (e.g. for the background) then the background will be visible. This is especially important if you want to use an image as background.

Failing to make your tabs transparent will block out your background image totally.

## 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.

```
<PARAM name=images value="
^background|images/clouds.jpg
^tableft|images/tabtrans.gif|0|0|2|100
^tabmiddle|images/tabtrans.gif|2|0|1|100
^tabright|images/tabtrans.gif|3|0|2|100
^tabline|images/tabtrans.gif|5|0|1|100
^selectedtableft|images/tabtrans.gif|6|0|2|100
^selectedtabmiddle|images/tabtrans.gif|8|0|1|100
^selectedtabright|images/tabtrans.gif|9|0|2|100
">
```




 the background image

Notice that the default tabs are not ‘transparent’. The java language doesn’t allow creating transparent off-screen images. In tabcomposition we have demonstrated how you can use external tabs with transparent backgrounds.

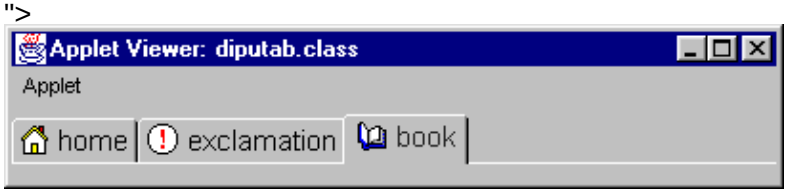
## icons

You can define an icon for every tab. An icon when the tab is not selected and an icon when the tab is selected.

## examples

```
<PARAM name=entries value="
^a|home||home|home
^b|exclamation||excl|excl
^c|book||bookclosed|bookopen
">
<PARAM name=images value="
^home|images/home.gif
^excl|images/excl.gif
^bookopen|images/bbooko.gif
```

^bookclosed|images/bbookc.gif
">



# 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. 5 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

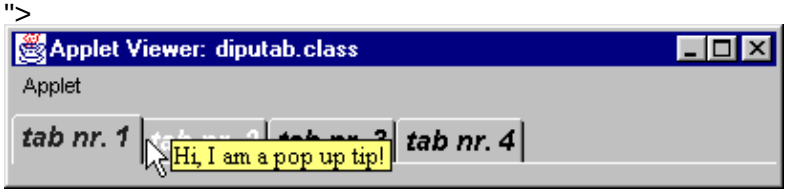
defaultfont and tipdefaultfont are built in font profiles. These 2 profiles can be used without the need

to define them first in the fonts parameter.
You can override these built-in fontprofiles.

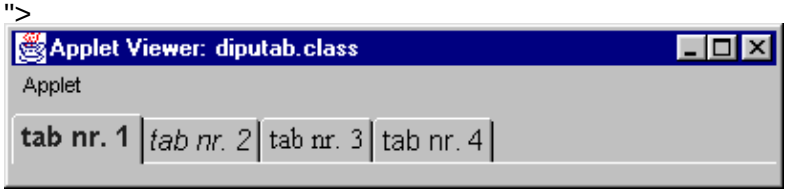
## examples

(assumptions: fontprofile = defaultfont, tipfontprofile = tipdefaultfont )

```
<PARAM name=fonts value="
^defaultfont|SansSerif|3|15
^tipdefaultfont|Serif|0|13
">
```




```
 <PARAM name=entries value="
^a|tab nr. 1||||font1
^b|tab nr. 2||||font2
^c|tab nr. 3||||font3
^d|tab nr. 4
">
 <PARAM name=fonts value="
^font1|SansSerif|1|15
^font2|SansSerif|2|15
^font3|Serif|0|15
">
```



# 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. diputab 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, foregroundcolor, textcolor, selectedcolor, tipcolor, tipbackgroundcolor, mouseovercolor, underlinecolor, highcolor, lowcolor and shadecolor

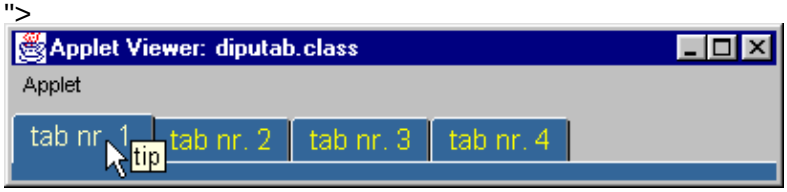
Colors are represented by a hexadecimal value. The first 2 digits are for the red component, the next 2 for the green component and the final 2 for the blue component.
The value of a color component can range from 00 to FF ( 00 = no intensity, FF = full intensity)
e.g. FFFFFF is the hexadecimal representation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

| **key** | **value** | |
|---|---|---|
| **name** | **value** | **description** |
| backgroundcolor | hexadecimal integer | the color of the background (behind the tabs) |
| foregroundcolor | hexadecimal integer | the color of the foreground (the tabs) |
| textcolor | hexadecimal integer | the color of the text |
| selectedcolor | hexadecimal integer | the 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 |
| highcolor | hexadecimal integer | the color of the line surrounding the tab |
| lowcolor | hexadecimal integer | the color of the line surrounding the tab |
| shadecolor | hexadecimal integer | the color of the line surrounding the tab |

**example**

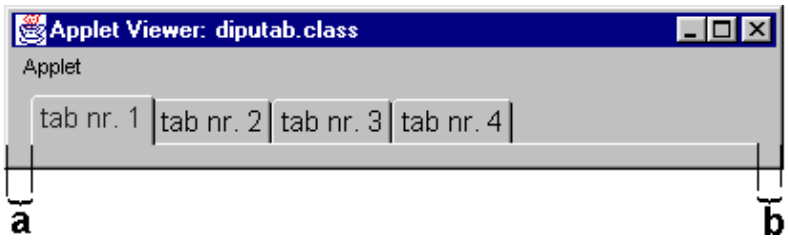
```
<param name="options" value="
^backgroundcolor|C0C0C0
^foregroundcolor|336699
^textcolor|FFFF00
^selectedcolor|FFFFFF
^tipcolor|000000
^tipbackgroundcolor|FFFFE0
^mouseovercolor|FFFFC0
^highcolor|FFFFFF
^lowcolor|000000
^shadecolor|808080
">
```




## applet margins

marginleft and marginright

| **key** | **value** | |
|---|---|---|
| **name** | **value** | **description** |
| marginleft | decimal integer | the left margin |
| marginright | decimal integer | the right margin |

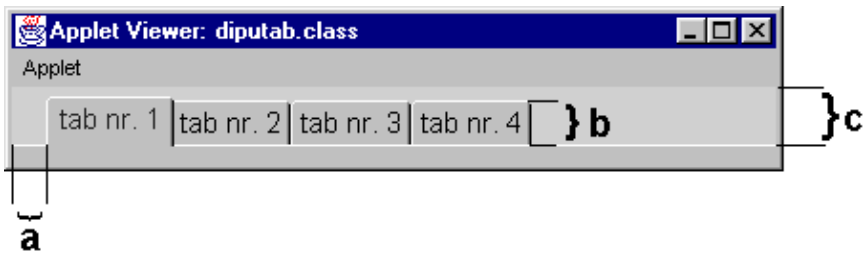



A: marginleft, B: marginright

## tab margins

baseline, tabheight and indent

| **key** | **value** | |
|---|---|---|
| **name** | **value** | **description** |
| baseline | decimal integer | the baseline of the tab |
| tabheight | decimal integer | the height of the tab |
| indent | decimal integer | indentation for the first tab |



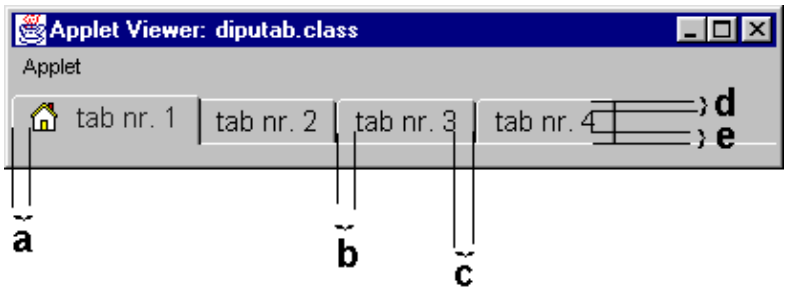

A: indent, B: tabheight , C: baseline

## text margins

textleft, textright, textup, textdown and imageleft

| **key** | **value** | |
|---|---|---|
| **name** | **value** | **description** |
| textleft | decimal integer | the space between the left side of the text and the tab (or the image if defined) |
| textright | decimal integer | the space between the right side of the text and the tab |

| | | |
|---|---|---|
| textup | decimal integer | the space between the upper side of the text and the tab |
| textdown | decimal integer | the space between the lower side of the text and the tab |
| imageleft | decimal integer | the space between the left side of the image and the tab |

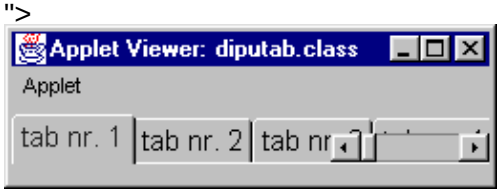



a: imageleft,b:textleft, c: textright, d: textup e: textdown

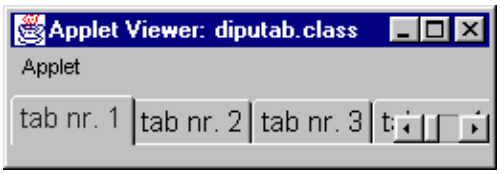
## scrollbars

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.
**keyvaluenamevaluedescription**scrollbarhorizontaldividerdecimal integerthe divider value for the horizontal scrollbar
(value can not be 0 )scrollbarverticaldivider
(only diputree)decimal integerthe divider value for the vertical scrollbar
(value can not be 0 )scrollbarwidthdecimal integerthe width of the scrollbar in pixelsscrollbuttonsizedecimal integerthe width and height of the buttonsunitdividerdecimal integerthe amount (divider value) that should be scrolled when the user wants
to scroll by 1 unit ( by clicking on the scroll button )blockdividerdecimal integerthe amount (divider value) that should be scrolled when the user wants
to scroll by 1 unit ( by clicking on the scroll button )scrollspeeddecimal integerthe amount of time between 2 consecutive scrollings in milliseconds
**Example**

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




```
<param name="options" value="
^scrollbarhorizontaldivider|5
">
```




## 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** | **value** | **description** |
| frame | String | the name of the default frame where the pages will be displayed,<br>can be overridden in the links parameter. |

- ♦ (value: string)

## fontprofile

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

## showstatus

| **key** | **value** | |
|---|---|---|
| **name** | **value** | **description** |
| showstatus | integer boolean | showing status messages can be switched on or off<br>( 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** | **value** | **description** |
| mouseover | integer boolean | highlighting due to the mouse being rolled over can be switched on or off<br>(0 = off, 1 = on) |
| underline | integer boolean | underlining due to the mouse being rolled over can be switched on or off<br>(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** | **value** | **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** | **value** | **description** |
| selected | string | the key from the entries table, that should be set as the selected record. |

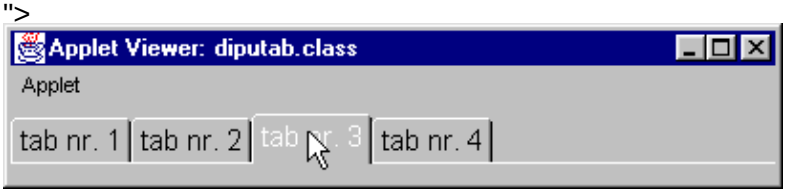
## overlap

Overlap is the amount of pixels that the selected tab will extend over the adjacent tabs. This will add a perspective effect.

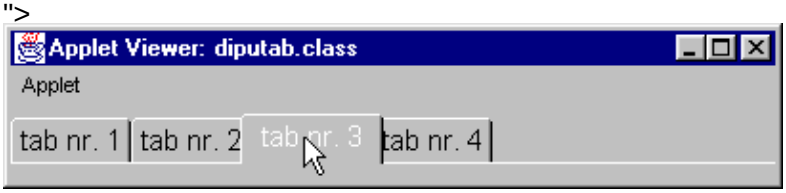
| key | value | |
|---|---|---|
| **name** | **value** | **description** |
| overlap | decimal integer | the width in pixels that the selected tab will overlap over the adjacent tabs |

### Example

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




```
<param name="options" value="
^overlap|6
">
```



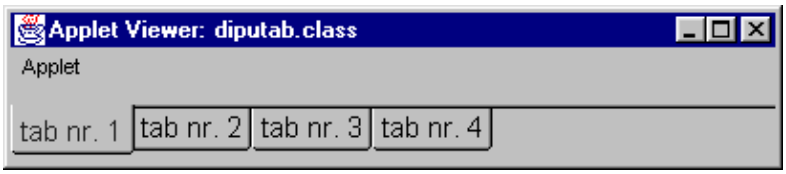

## position

The tabs can be positioned at the top or the bottom.

| key | value | |
|---|---|---|
| **name** | **value** | **description** |
| position | String | The position where the tabs will be located.<br>(values: "top" or "bottom") |

### Example

```
<param name="options" value="
^position|bottom
">
```



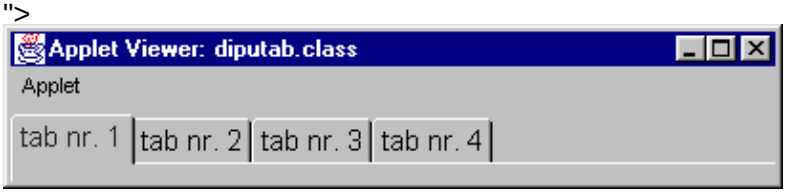

## connecting lines

connectlineleft and connectlineright

If the tab is used in conjunction with a borderline from another html page a little gap can exist between the tabs and the borderline in the other frame.
Use connectlineleft and connectlineright to connect the tab with the borderline.

| **key** | **value** | |
|---|---|---|
| **name** | **value** | **description** |
| connectlineleft | integer boolean | connect a line to the left<br>( 0 = no connection , 1 = connection) |
| connectlineleft | integer boolean | connect a line to the right<br>(0 = no connection and 1 = connection) |

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




## 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 | value | description |
|---|---|---|
| externalhandlers | boolean integer | instructs the applet to test if external handlers are supported<br>(values: 0 = don't test an consequently don't allow external handlers<br>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** | **value** | **description** |
| asyncload | boolean integer | tells the applet to enable asynchronous loading of images<br>(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’s 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/diputab.class", then the baseurl will be "http://www.mysite.com/mydirectory".

baseurl is used in the links parameter.

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

# 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

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

## adding or replacing records

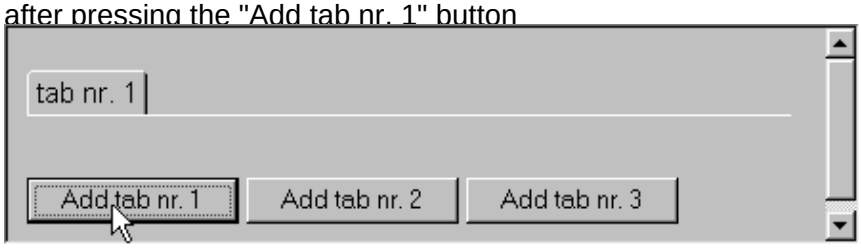
With the addRecords(tablename,records) method you can add record(s) to tha applet in the specified table. If there is already a record with the specified key, then the new record will replace the old record.

**example**

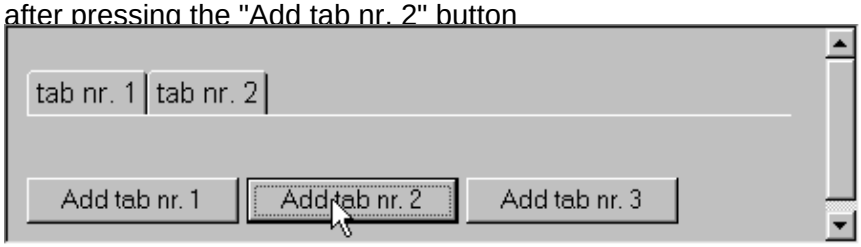
```
<html>
<body bgcolor="#C0C0C0">
<applet CODE="diputab.class" NAME="diputab" WIDTH="400" HEIGHT="40">
</applet>
<form>
<input TYPE="BUTTON" VALUE="Add tab nr. 1"
onClick="diputab.addRecords('entries','^a|tab nr. 1');">
<input TYPE="BUTTON" VALUE="Add tab nr. 2"
onClick="diputab.addRecords('entries','^b|tab nr. 2');">
<input TYPE="BUTTON" VALUE="Add tab nr. 3"
onClick="diputab.addRecords('entries','^c|tab nr. 3');">
</form>
</body>
</html>
```

after loading the page

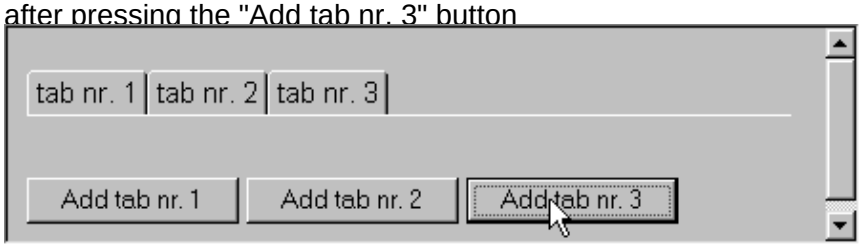

after pressing the "Add tab nr. 1" button



after pressing the "Add tab nr. 2" button




after pressing the "Add tab nr. 3" button



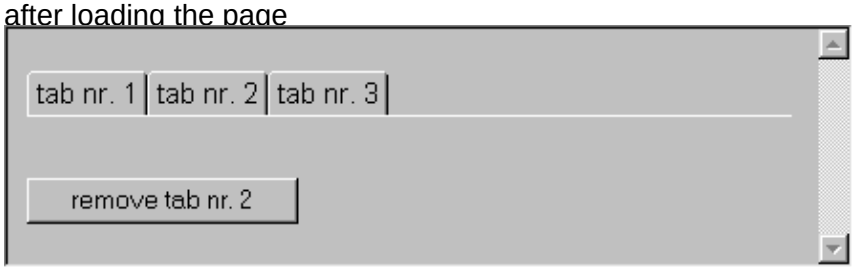

## removing records

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

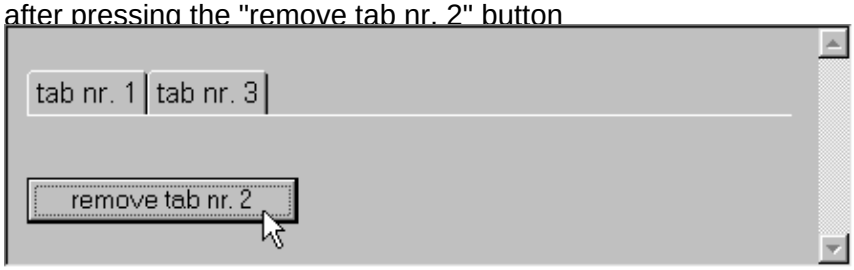
Example

```
<html>
<body bgcolor="#C0C0C0">
<applet CODE="diputab.class" NAME="diputab" WIDTH="400" HEIGHT="40">
<param name="entries" value="
^a|tab nr. 1
^b|tab nr. 2
^c|tab nr. 3
">
</applet>

<form>
<input TYPE="BUTTON" VALUE="remove tab nr. 2"
onClick="diputab.removeRecords('entries','^b');"></p>
</form>

</body>
</html>
```

after loading the page



after pressing the "remove tab nr. 2" 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.

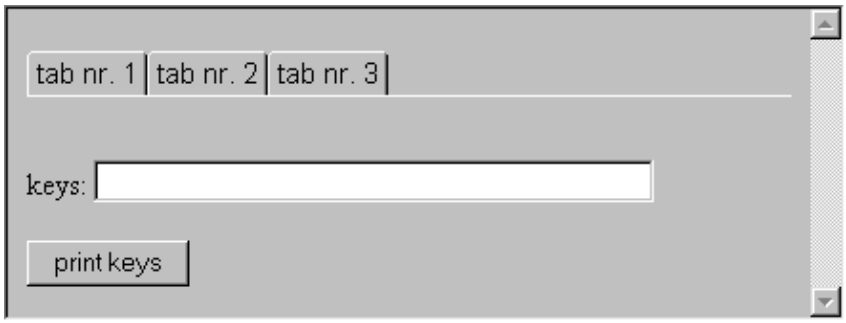
**array String.split (separator)**

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.

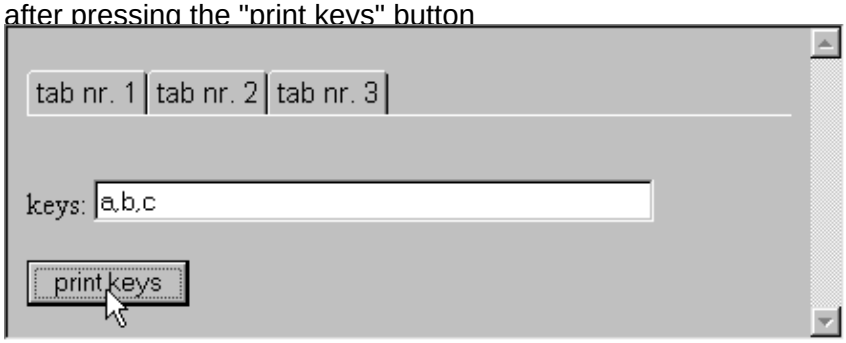
**example**

```
<html>
<body bgcolor="#C0C0C0">
<applet CODE="diputab.class" NAME="diputab" WIDTH="400" HEIGHT="40">
<param name="entries" value="
^a|tab nr. 1
^b|tab nr. 2
^c|tab nr. 3
">
</applet>

<form name="main">
<BR>keys: <input type="text" name="keys" size="40">
<BR><input TYPE="BUTTON" VALUE="print keys" onClick="main.keys.value =
diputab.getKeys('entries',',');">
</form>
</body>
</html>
```

after loading the page




after pressing the "print keys" button



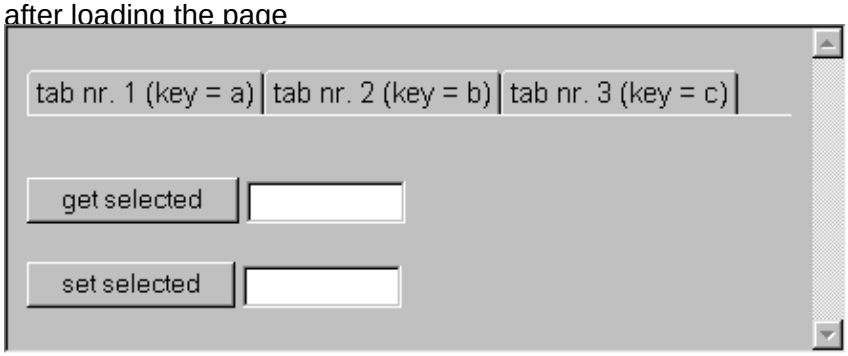

## getting and setting the selected tab

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

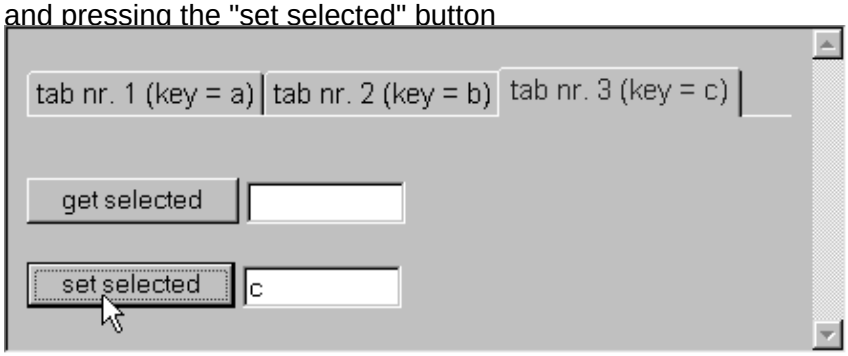
Example

```
<html>
<body bgcolor="#C0C0C0">
<applet CODE="diputab.class" NAME="diputab" WIDTH="400" HEIGHT="40">
<param name="entries"
value="
^a|tab nr. 1 (key = a)
^b|tab nr. 2 (key = b)
^c|tab nr. 3 (key = c)
">
</applet>
 <form name="main">
<p><input TYPE="BUTTON" VALUE="get selected"
onClick="main.getselected.value = diputab.getSelected();">
<input type="text" name="getselected" size="10"></p>
 <p><input TYPE="BUTTON" VALUE="set selected"
onClick="diputab.setSelected(main.setselected.value);">
<input type="text" name="setselected" size="10"></p>
</form>
</body>
</html>
```

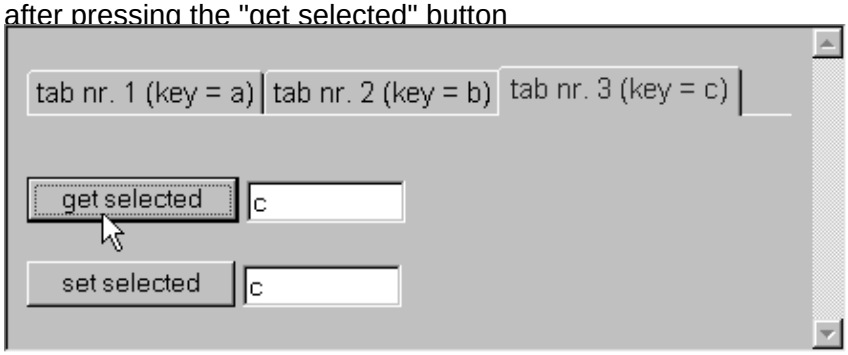
after loading the page




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**

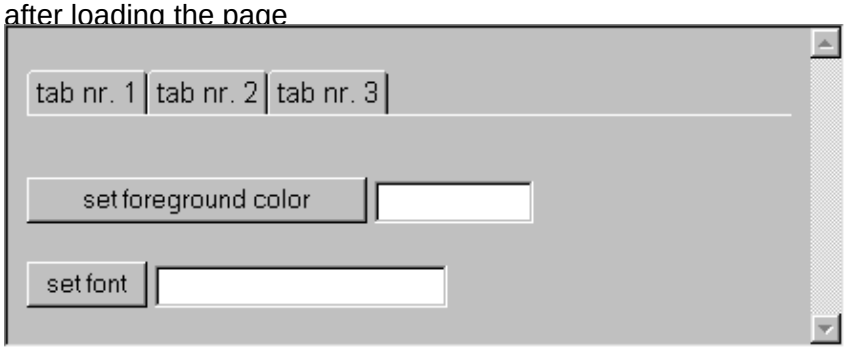
You can not edit individual fields in the images table. Just use addRecords(tablename,records) to

add new images to the applet.

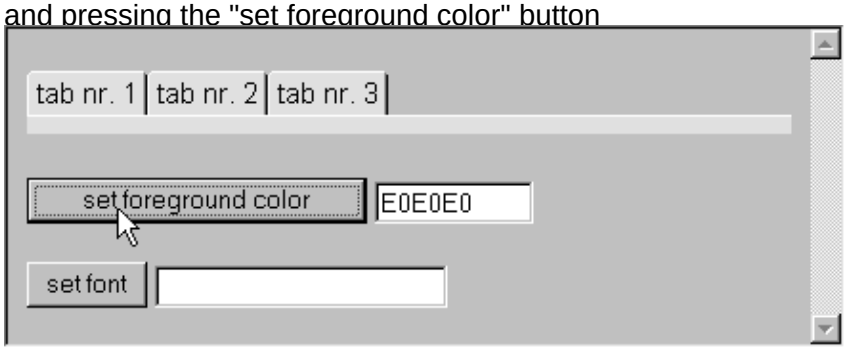
**examples**

```
<html>
<body bgcolor="#C0C0C0">
<applet CODE="diputab.class" NAME="diputab" WIDTH="400" HEIGHT="40">
<param name="entries" value="
^a|tab nr. 1
^b|tab nr. 2
^c|tab nr. 3
">
</applet>
 <form name="main">
<p><input TYPE="BUTTON" VALUE="set foreground color"
onClick="diputab.setField('options','foregroundcolor',1,main.fgcolor.value);">
<input type="text" name="fgcolor" size="10"></p>
 <p><input TYPE="BUTTON" VALUE="set font"
onClick="diputab.setField('fonts','defaultfont',1,main.font.value);">
<input type="text" name="font" size="20"></p>
</form>
</body>
</html>
```

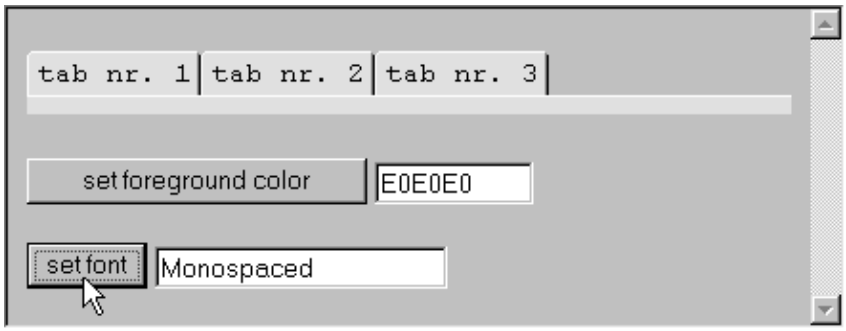
after loading the page




after filling in E0E0E0 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


tab nr. 1
tab nr. 2
tab nr. 3
set foreground color
E0E0E0
set font
Monospaced

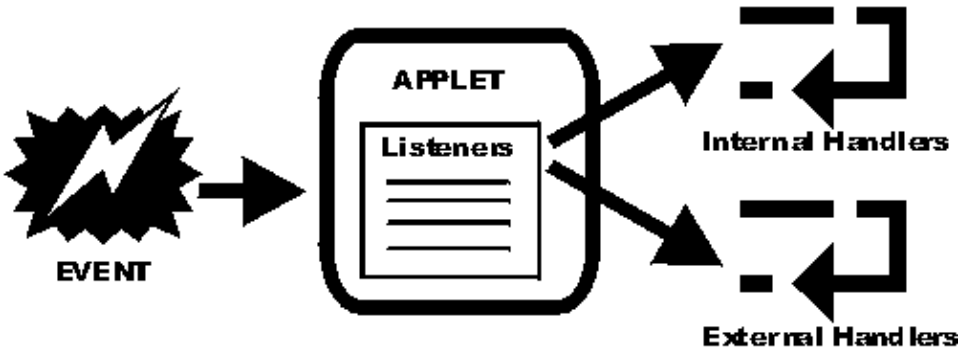
# 3. The events framework

## 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).

diputab has only one type of listeners, the "selected" 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 tabs.

## 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 diputab. 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.

# 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|tab nr. 1
  ">
- The entries.txt file
  ^b|tab nr. 2
  ^c|tab nr. 3

### adding a record

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

 <param name="internalhandlers" value="
^adding|1|entries|b|tab nr.2 was added !
">

 <param name="entries" value="
^a|tab nr. 1
">
```

### 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|tab nr. 1
">
```

**remove 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|tab nr. 1
">
```

**set selected (repeated)**

```
<param name="selected" value="
^a|++select
">
<param name="internalhandlers" value="
^select|4||c
">
<param name="entries" value="
^a|tab nr. 1
^b|tab nr. 2
^c|tab nr. 3
">
```

## 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 tab nr. 1')
^popb|alert ('you clicked tab nr. 2')
">

<param name="entries" value="
^a|tab nr. 1
^b|tab 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 diputab 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:

- diputab.class
- diputab.jar

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

## diputab.jar

diputab.jar is just the diputab.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.

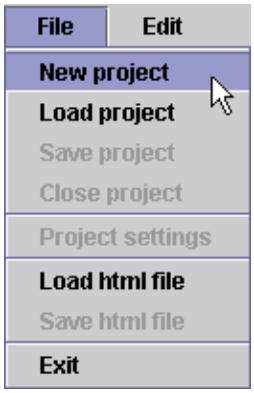
# Projects

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

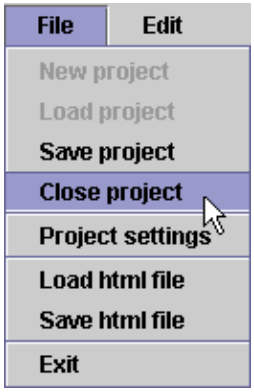
You can create a new project when the configurator is just started or when you closed the current project.




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

## close a Project

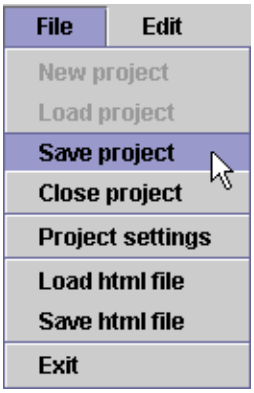
You can close the current project. This will return the configurator to the initial state, allowing you to create or open a new project.




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

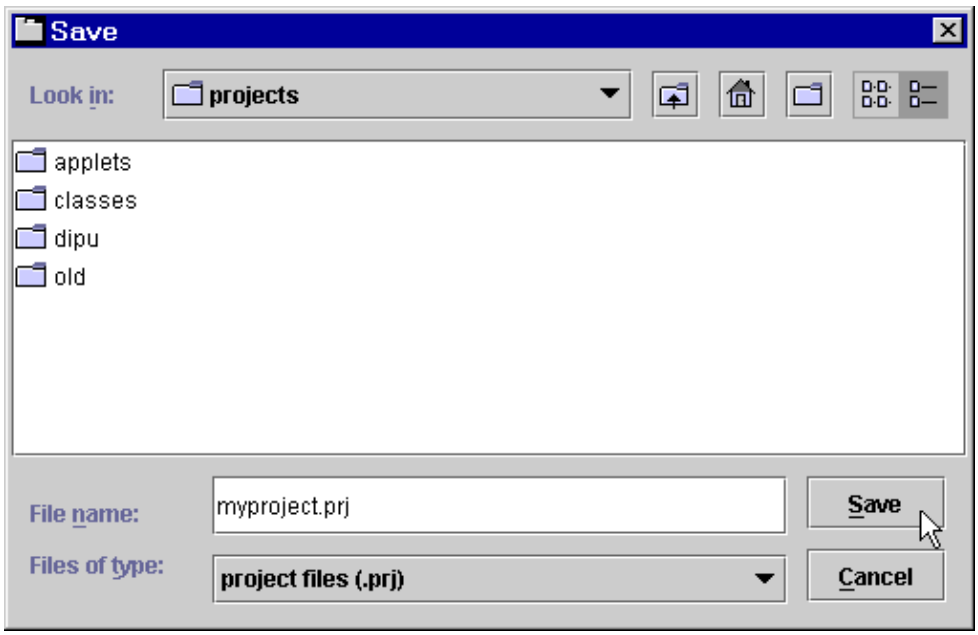
## save a Project

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.

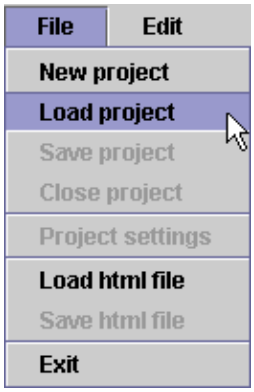
2. Give the project a name and select save



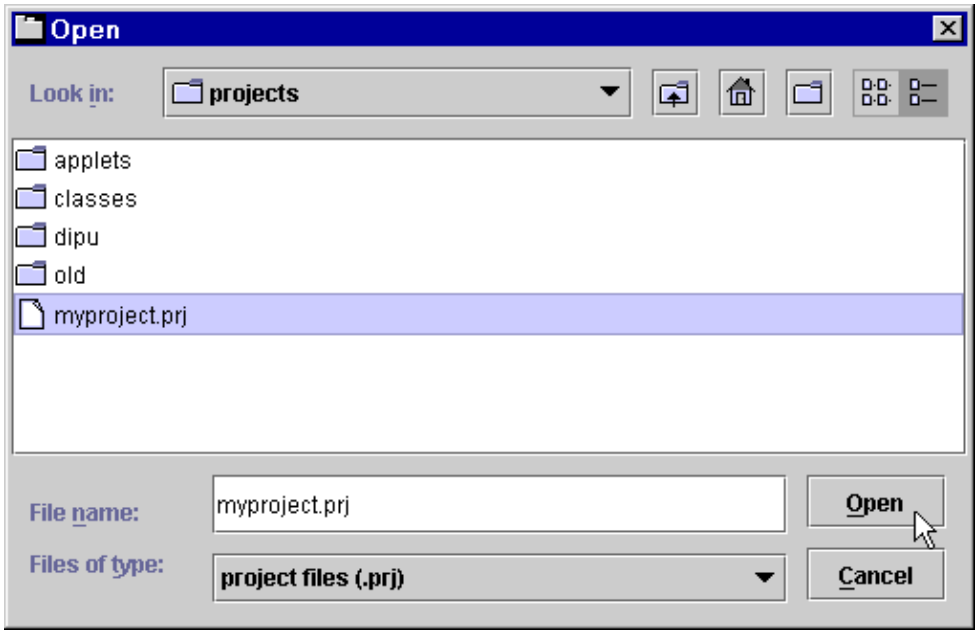

## load a Project

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.

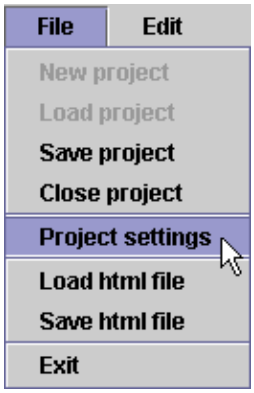



2. Select the project and press load.



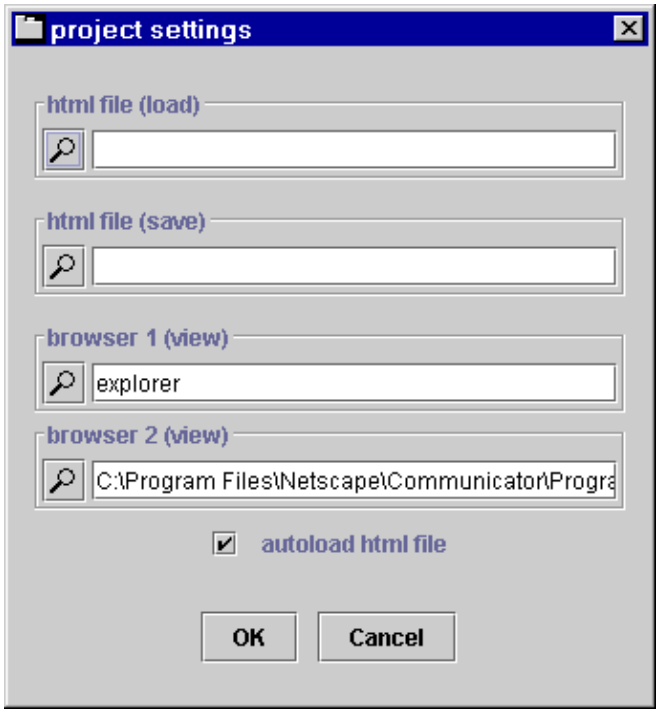

## project Settings

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<br>(unchecked = don't autoload , checked = autoload ) |

# Html files

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

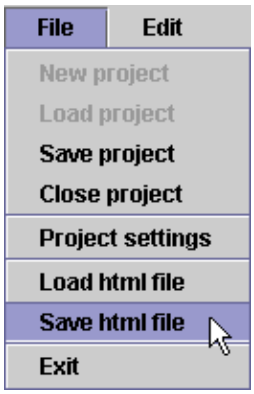
How the configurator works with html files

- 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!

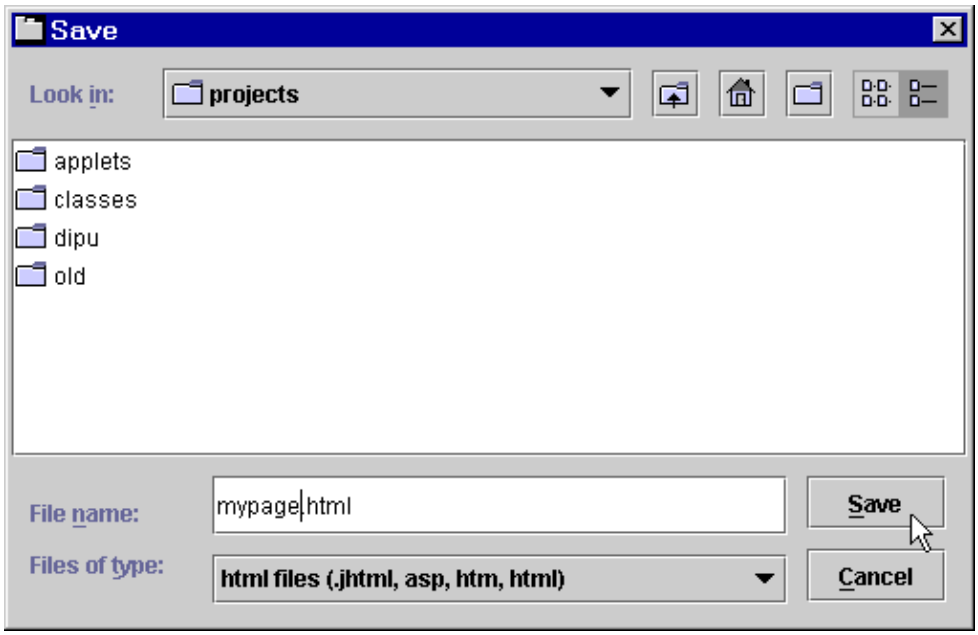
## save a html file

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



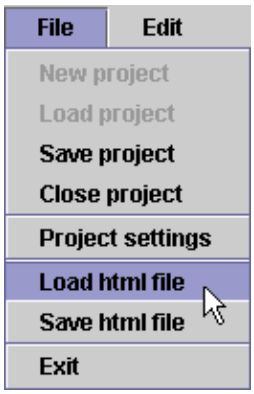

1. In the "File" Menu select "Save html file" or press  in the tool bar.

2. Give the html file a name and select save



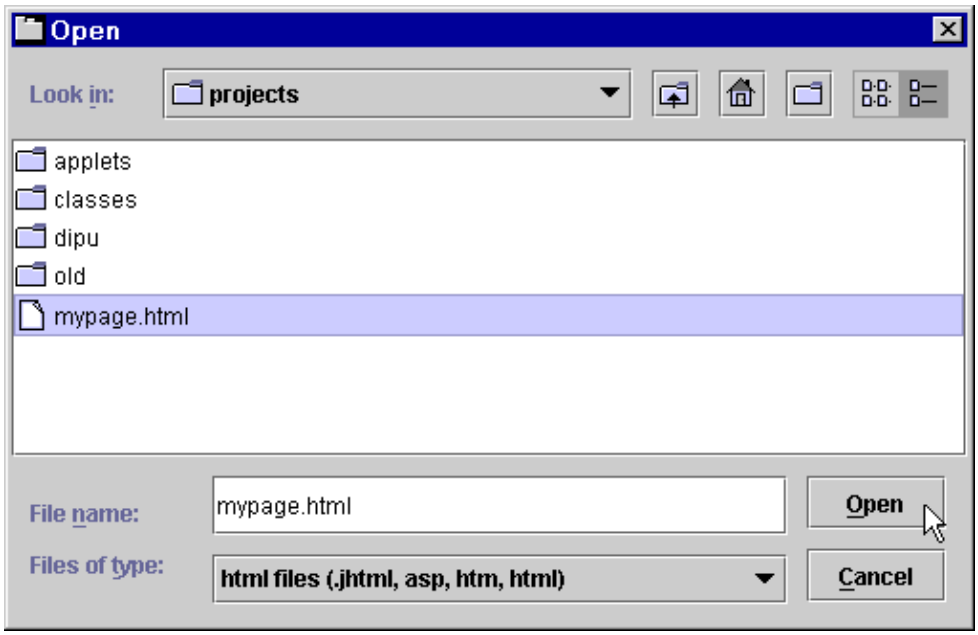

## load a html file

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

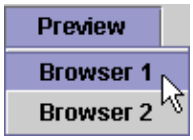
# Previewing

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

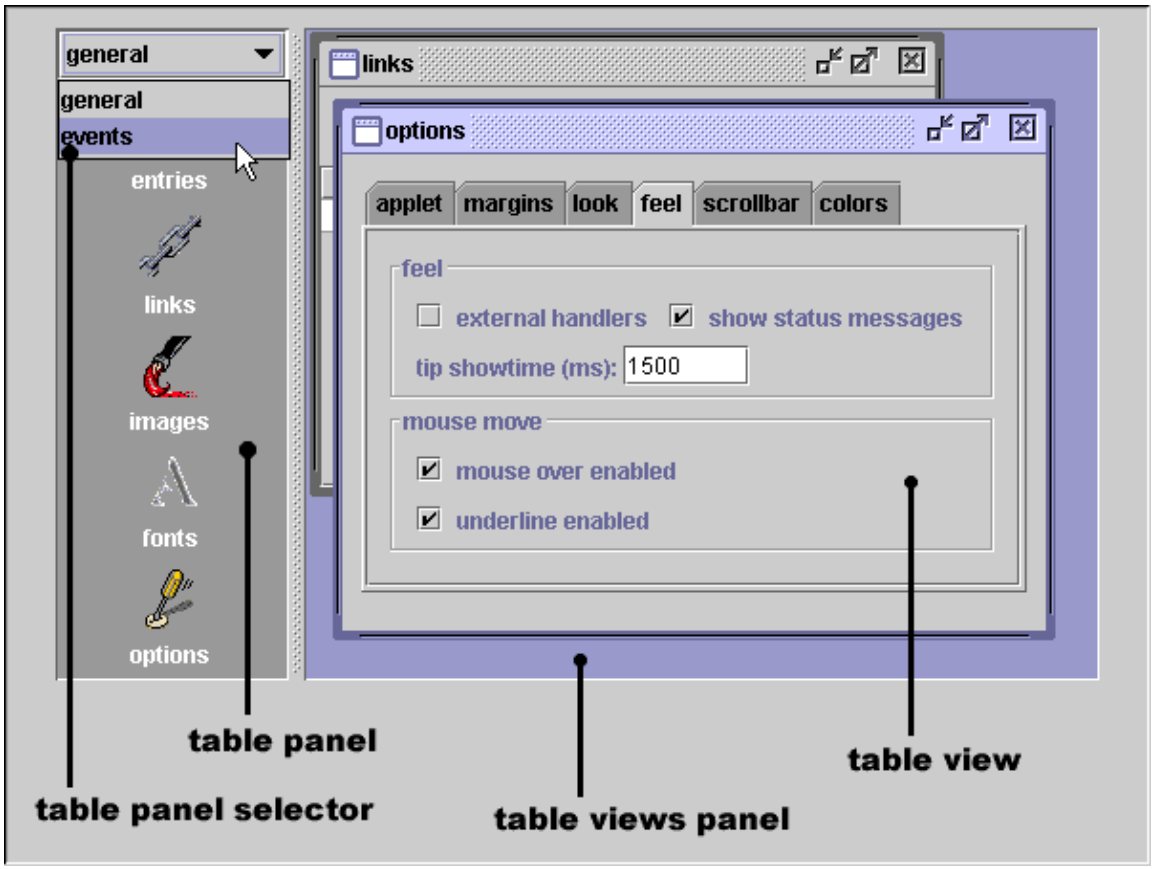



- In the "Preview" Menu select "Browser 1" or "Browser 2"
- Click in the toolbar on  for previewing in browser 1 or on  for previewing in browser 2.

# 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**

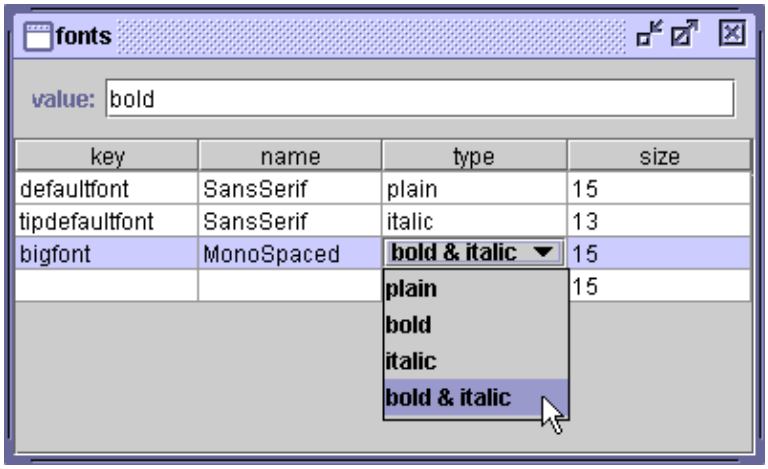
[internal handlers](#) [external handlers](#) [selected](#)

## editing a table

### in cell editing

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

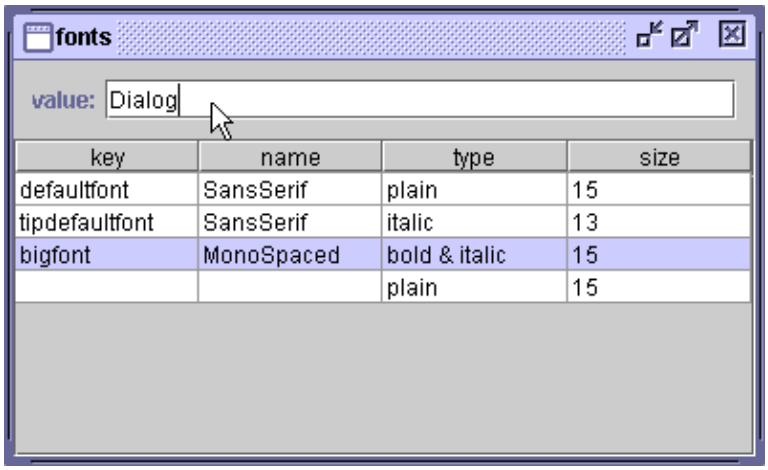
If the cell is an external key or the possible values are limited and known then a combo box is presented. This will prevent you from making typing errors and keep referential integrity among tables.




**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