DiVinE

0.8.0

Generated by Doxygen 1.5.6

Fri Mar 6 09:14:49 2009

Contents

Chapter 1

Class Index

1.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 2

Class Documentation

2.1 DvePropertyEditor Class Reference

Panel used to show and change Dve Property, contains only one textarea.

Inherits divine::client::MainWindow::NodePanel.

Public Member Functions

- **DvePropertyEditor** (Client c, Property p)
- Property **property** ()
- boolean **update** (Node e)

Protected Attributes

- PlainTextViewer **m_text**

2.1.1 Detailed Description

Panel used to show and change Dve Property, contains only one textarea.

Author:

xforejt

The documentation for this class was generated from the following file:

- DvePropertyEditor.java

2.2 DwiServer Class Reference

Server part of the project, listens on the specified port and creates new instances of DwiSocket.

Public Member Functions

- String **dataPath** (String f)
- boolean **shouldStop** ()
- void **start** (int port, String data) throws Exception
- MapNode< User.Passwd > **users** ()

Static Public Member Functions

- static void **main** (String[ ] args) throws Exception
- static DwiServer **the** () throws Exception

Protected Member Functions

- String **dataFile** (String name) throws Exception

Protected Attributes

- TreeMap< DwiSocket, Thread > **m_activeSockets**
- String **m_dataPath**
- int **m_port**
- MapNode< User.Passwd > **m_users**

Static Protected Attributes

- static DwiServer **s_server**

2.2.1 Detailed Description

Server part of the project, listens on the specified port and creates new instances of DwiSocket.

Author:

xforejt, mornfall

The documentation for this class was generated from the following file:

- DwiServer.java

2.3 ExtensionFilter Class Reference

Filter that accepts regular files with given extension.

Public Member Functions

- boolean **accept** (File f)
- ExtensionFilter (String[ ] extensions)

  Creates a new instance of FilenameFilterOwner.

- String **getDescription** ()

2.3.1 Detailed Description

Filter that accepts regular files with given extension.

It also accepts all directories.

Author:

xforejt

2.3.2 Constructor & Destructor Documentation

2.3.2.1 ExtensionFilter (String[ ] *extensions*)

Creates a new instance of FilenameFilterOwner.

Parameters:

extensions Array of allowed extensions

The documentation for this class was generated from the following file:

- ExtensionFilter.java

2.4 Global Class Reference

Provides often used static methods;.

Static Public Member Functions

- static String **createMD5** (String key)

2.4.1 Detailed Description

Provides often used static methods;.

Author:

xforejt

The documentation for this class was generated from the following file:

- Global.java

2.5 JobLogViewer Class Reference

Derived from code by xforejt.

Inherits divine::client::MainWindow::NodePanel.

Public Member Functions

- JobLogViewer (Client c, Job j)

  Creates new form RecentLogPanel.

- boolean **update** (Node n)

2.5.1 Detailed Description

Derived from code by xforejt.

The documentation for this class was generated from the following file:

- JobLogViewer.java

2.6 Property Class Reference

Properties are (usually) LTL formulae concerning some model.

Inherits divine::common::Node, and divine::common::Wired::Interface.

Public Types

- enum **Type** {
  PBS, **Interactive**, **DVE**, **PML**,
  LTL, **StateSpace**, **PropertyProcess**, **Safety** }

Public Member Functions

- String **canonicalName** ()
- Object **clone** ()
- Object **fromWire** (String s) throws Exception
- Model **model** ()
- void **setText** (String t)
- void **setType** (Type type)
- String **text** ()
- String **toWire** () throws Exception
- Type **type** ()
- String **typeString** ()
- boolean **typeValid** ()

Static Public Member Functions

- static Type **typeFromString** (String s)
- static String **typeToString** (Type t)

Protected Attributes

- String **m_text**
- Type **m_type** = Type.LTL

2.6.1 Detailed Description

Properties are (usually) LTL formulae concerning some model.

The documentation for this class was generated from the following file:

- Property.java

2.7 PropertyEditor Class Reference

Panel used to display and change the content of the property.

Inherits divine::client::MainWindow::NodePanel.

Public Member Functions

- Property **property** ()
- **PropertyEditor** (Client c, Property p)
- void **setEnabled** (boolean e)
- boolean **update** (Node elem)

Protected Member Functions

- void **messageDialog** (String t, String c, int type)

Protected Attributes

- Client **m_client**
- boolean **m_inUpdate**

2.7.1 Detailed Description

Panel used to display and change the content of the property.

Author:

xforejt

The documentation for this class was generated from the following file:

- PropertyEditor.java

2.8 SpringUtilities Class Reference

A 1.4 file that provides utility methods for creating form- or grid-style layouts with SpringLayout.

Static Public Member Functions

- static void makeCompactGrid (Container parent, int rows, int cols, int initialX, int initialY, int xPad, int yPad)

  Aligns the first `rows * cols` *components of* `parent` *in a grid.*

- static void makeGrid (Container parent, int rows, int cols, int initialX, int initialY, int xPad, int yPad)

  Aligns the first `rows * cols` *components of* `parent` *in a grid.*

- static void printSizes (Component c)

  A debugging utility that prints to stdout the component's minimum, preferred, and maximum sizes.

2.8.1 Detailed Description

A 1.4 file that provides utility methods for creating form- or grid-style layouts with SpringLayout.

These utilities are used by several programs, such as SpringBox and SpringCompactGrid.

2.8.2 Member Function Documentation

2.8.2.1 static void makeCompactGrid (Container *parent*, int *rows*, int *cols*, int *initialX*, int *initialY*, int *xPad*, int *yPad*) `[static]`

Aligns the first `rows * cols` components of `parent` in a grid.

Each component in a column is as wide as the maximum preferred width of the components in that column; height is similarly determined for each row. The parent is made just big enough to fit them all.

Parameters:

rows number of rows

cols number of columns

initialX x location to start the grid at

initialY y location to start the grid at

xPad x padding between cells

yPad y padding between cells

2.8.2.2 static void makeGrid (Container *parent*, int *rows*, int *cols*, int *initialX*, int *initialY*, int *xPad*, int *yPad*) [static]

Aligns the first `rows * cols` components of `parent` in a grid.

Each component is as big as the maximum preferred width and height of the components. The parent is made just big enough to fit them all.

Parameters:

rows number of rows

cols number of columns

initialX x location to start the grid at

initialY y location to start the grid at

xPad x padding between cells

yPad y padding between cells

The documentation for this class was generated from the following file:

- SpringUtilities.java

2.9 VTextIcon Class Reference

VTextIcon is an Icon implementation which draws a short string vertically.

Public Member Functions

- int getIconHeight ()

  Returns the icon's height.

- int getIconWidth ()

  Returns the icon's width.

- void paintIcon (Component c, Graphics g, int x, int y)

  Draw the icon at the specified location.

- void propertyChange (PropertyChangeEvent e)

  Checks for changes to the font on the fComponent so that it can invalidate the layout if the size changes.

- void setLabel (String label)

  sets the label to the given string, updating the orientation as needed and invalidating the layout if the size changes

- VTextIcon (Component component, String label, int rotateHint)

  Creates a `VTextIcon` *for the specified* `component` *with the specified* `label`.

- VTextIcon (Component component, String label)

  Creates a `VTextIcon` *for the specified* `component` *with the specified* `label`.

Static Public Member Functions

- static int verifyRotation (String label, int rotateHint)

  verifyRotation

Static Public Attributes

- static final int **ROTATE_DEFAULT** = 0x00
- static final int **ROTATE_LEFT** = 0x02
- static final int **ROTATE_NONE** = 0x01
- static final int **ROTATE_RIGHT** = 0x04

Package Functions

- void **calcDimensions** ()
- void recalcDimensions ()

  Calculates the dimensions.

Package Attributes

- int **fCharHeight**
- String[ ] **fCharStrings**
- int[ ] **fCharWidths**
- Component **fComponent**
- int **fDescent**
- int **fHeight**
- String **fLabel**
- int[ ] **fPosition**
- int **fRotation**
- int **fWidth**

Static Package Attributes

- static final int **DEFAULT_CJK** = ROTATE_NONE
- static final int **DEFAULT_MUST_ROTATE** = ROTATE_LEFT
- static final int **DEFAULT_ROMAN** = ROTATE_RIGHT
- static final int **kBufferSpace** = 5
- static final int **LEGAL_MUST_ROTATE** = ROTATE_LEFT | ROTATE_-RIGHT
- static final int **LEGAL_ROMAN** = ROTATE_NONE | ROTATE_LEFT | ROTATE_RIGHT
- static final double **NINETY_DEGREES** = Math.toRadians(90.0)
- static final int **POSITION_FAR_TOP_RIGHT** = 2
- static final int **POSITION_NORMAL** = 0
- static final int **POSITION_TOP_RIGHT** = 1
- static final String **sDrawsInFarTopRight** = "\u3001\u3002"
- static final String **sDrawsInTopRight**

2.9.1 Detailed Description

VTextIcon is an Icon implementation which draws a short string vertically.

It's useful for JTabbedPanes with LEFT or RIGHT tabs but can be used in any component which supports Icons, such as JLabel or JButton

You can provide a hint to indicate whether to rotate the string to the left or right, or not at all, and it checks to make sure that the rotation is legal for the given string (for example, Chinese/Japanese/Korean scripts have special rules when drawn vertically and should never be rotated)

2.9.2 Constructor & Destructor Documentation

2.9.2.1 VTextIcon (Component *component*, String *label*)

Creates a `VTextIcon` for the specified `component` with the specified `label`.

It sets the orientation to the default for the string

See also:

verifyRotation

2.9.2.2 VTextIcon (Component *component*, String *label*, int *rotateHint*)

Creates a `VTextIcon` for the specified `component` with the specified `label`.

It sets the orientation to the provided value if it's legal for the string

See also:

verifyRotation

2.9.3 Member Function Documentation

2.9.3.1 int getIconHeight ()

Returns the icon's height.

Returns:

an int specifying the fixed height of the icon.

2.9.3.2 int getIconWidth ()

Returns the icon's width.

Returns:

an int specifying the fixed width of the icon.

2.9.3.3 void paintIcon (Component *c*, Graphics *g*, int *x*, int *y*)

Draw the icon at the specified location.

Icon implementations may use the Component argument to get properties useful for painting, e.g. the foreground or background color.

2.9.3.4 void recalcDimensions () [package]

Calculates the dimensions.

If they've changed, invalidates the component

2.9.3.5 void setLabel (String *label*)

sets the label to the given string, updating the orientation as needed and invalidating the layout if the size changes

See also:

verifyRotation

2.9.3.6 static int verifyRotation (String *label*, int *rotateHint*) [static]

verifyRotation

returns the best rotation for the string (ROTATE_NONE, ROTATE_LEFT, ROTATE_-RIGHT)

This is public static so you can use it to test a string without creating a VTextIcon

from http://www.unicode.org/unicode/reports/tr9/tr9-3.html
When setting text using the Arabic script in vertical lines, it is more common to employ a horizontal baseline that is rotated by 90 deg counterclockwise so that the characters are ordered from top to bottom. Latin text and numbers may be rotated 90 deg clockwise so that the characters are also ordered from top to bottom.

Rotation rules

- Roman can rotate left, right, or none - default right (counterclockwise)
- CJK can't rotate
- Arabic must rotate - default left (clockwise)

from the online edition of _The Unicode Standard, Version 3.0_, file ch10.pdf page 4 Ideographs are found in three blocks of the Unicode Standard... U+4E00-U+9FFF, U+3400-U+4DFF, U+F900-U+FAFF

Hiragana is U+3040-U+309F, katakana is U+30A0-U+30FF

from http://www.unicode.org/unicode/faq/writingdirections.html
East Asian scripts are frequently written in vertical lines which run from top-to-bottom and are arrange columns either from left-to-right (Mongolian) or right-to-left (other scripts). Most characters use the same shape and orientation when displayed horizontally or vertically, but many punctuation characters will change their shape when displayed vertically.

Letters and words from other scripts are generally rotated through ninety degree angles so that they, too, will read from top to bottom. That is, letters from left-to-right scripts

will be rotated clockwise and letters from right-to-left scripts counterclockwise, both through ninety degree angles.

Unlike the bidirectional case, the choice of vertical layout is usually treated as a formatting style; therefore, the Unicode Standard does not define default rendering behavior for vertical text nor provide directionality controls designed to override such behavior

2.9.4 Member Data Documentation

2.9.4.1 final String sDrawsInTopRight [static, package]

Initial value:

```
"\u3041\u3043\u3045\u3047\u3049\u3063\u3083\u3085\u3087\u308E" +
"\u30A1\u30A3\u30A5\u30A7\u30A9\u30C3\u30E3\u30E5\u30E7\u30EE\u30F5\u30F6"
```

The documentation for this class was generated from the following file:

- VTextIcon.java

Index