

com.fr.stable.fun.FunctionDefContainer

-
-
-
-
-
-
-
-

AbstractFunctionFunctionDefContainerAbstractFunctionFunctionDefContainer

AbstractFunctionFunctionDefContainer

☐ 导出/编辑结果时, 保留公式 ☒ 填报/分析时, 保留公式用于计算

检查合法性

函数类型:	函数名:	变量:	公式说明:
常用函数	Func1	数据项 变量 - [16] \$\$page_number \$\$totalPage_number \$fine_username \$fine_role \$fine_position NULL NOFILTER reportName formletName servletURL serverSchema serverName serverPort serverURL	DemoFunc1 的用法介绍 换行说明!
数学和三角函数	Func2		
文本函数			
日期和时间函数			
逻辑函数			
数组函数			
报表函数			
其它函数			
层次坐标函数			
全部函数			
自定义函数			
插件函数			
Demo			

函数类型:	函数名:	变量:	公式说明:
常用函数	Func3	数据项 变量 - [16] \$\$page_number \$\$totalPage_number \$fine_username \$fine_role \$fine_position NULL NOFILTER reportName formletName servletURL serverSchema serverName serverPort serverURL	DemoFunc3 的用法介绍!
数学和三角函数			
文本函数			
日期和时间函数			
逻辑函数			
数组函数			
报表函数			
其它函数			
层次坐标函数			
全部函数			
自定义函数			
插件函数			
Demo			

FunctionDefContainer.java

```
package com.fr.stable.fun;

import com.fr.stable.fun.mark.Mutable;
import com.fr.stable.script.FunctionDef;

/**
 *
 */
public interface FunctionDefContainer extends Mutable {

    int CURRENT_LEVEL = 1;

    String MARK_STRING = "FunctionGroup";

    /**
     *
     * @return
     */
    FunctionDef[] getFunctionDefs();

    /**
     *
     * @return
     */
    String getGroupName();
}
```

FunctionDef.java

```
package com.fr.stable.script;

import com.fr.plugin.injectable.SpecialLevel;
import com.fr.stable.StringUtils;
import com.fr.stable.fun.FunctionDefineProvider;
import com.fr.stable.xml.XMLPrintWriter;
import com.fr.stable.xml.XMLable;
import com.fr.stable.xml.XMLableReader;

/**
 *
 */
//// TODO: 2017/12/14 holder
public class FunctionDef implements FunctionDefineProvider, XMLable {

    public static final String XML_TAG = SpecialLevel.FunctionDef.getTagName();

    public static final int CURRENT_LEVEL = 1;

    private String name = StringUtils.EMPTY;
    private String description = StringUtils.EMPTY;

    private String className;

    /**
     *
     */
    public FunctionDef() {
        this(StringUtils.EMPTY, StringUtils.EMPTY);
    }

    /**
     * @param name
     * @param className
     */
    public FunctionDef(String name, String className) {
        this(name, StringUtils.EMPTY, className);
    }
```

```

    }

    /**
     * @param name
     * @param description
     * @param className
     */
    public FunctionDef(String name, String description, String className) {
        this.setName(name);
        this.setDescription(description);

        this.setClassName(className);
    }

    public int currentAPILevel() {
        return CURRENT_LEVEL;
    }

    /**
     * @return
     */
    public String getName() {
        return this.name;
    }

    /**
     *
     */
    public void setName(String name) {
        this.name = name;
    }

    /**
     * @return
     */
    public String getDescription() {
        return description;
    }

    /**
     *
     */
    public void setDescription(String description) {
        this.description = description;
    }

    /**
     * @return
     */
    public String getClassName() {
        return className;
    }

    /**
     *
     */
    public void setClassName(String className) {
        this.className = className;
    }

    public void readXML(XMLStreamReader reader) {
    }

    public void writeXML(XMLPrintWriter writer) {
    }

    @Override
    public boolean equals(Object obj) {
        return obj instanceof FunctionDef && name != null && name.equals(((FunctionDef) obj).name);
    }

```

```

    }

    @Override
    public int hashCode() {
        return name == null ? 0 : name.hashCode();
    }

    public Object clone() throws CloneNotSupportedException {
        return super.clone();
    }
}

```

Function.java

```

//
// Source code recreated from a .class file by IntelliJ IDEA
// (powered by Fernflower decompiler)
//

package com.fr.stable.script;

import com.fr.stable.UtilEvalError;
import java.io.Serializable;
import java.util.Locale;

public interface Function extends Serializable {
    Function.Type MATH = new Function.Type();
    Function.Type TEXT = new Function.Type();
    Function.Type DATETIME = new Function.Type();
    Function.Type LOGIC = new Function.Type();
    Function.Type ARRAY = new Function.Type();
    Function.Type REPORT = new Function.Type();
    Function.Type HA = new Function.Type();
    Function.Type OTHER = new Function.Type();
    Function.Type DELETE = new Function.Type();

    void setName(String var1);

    CalculatorProvider getCalculatorProvider();

    void setCalculator(CalculatorProvider var1);

    /** @deprecated */
    @Deprecated
    String getCN();

    /** @deprecated */
    @Deprecated
    String getEN();

    String getDescription(Locale var1);

    Function.Type getType();

    Object evalExpression(Node[] var1) throws UtilEvalError;

    public static final class Type {
        private Type() {
        }
    }
}

```

AbstractFunction.java

```

package com.fr.script;

```

```

import com.fr.common.annotations.Open;
import com.fr.locale.InterProviderFactory;
import com.fr.stable.Primitive;
import com.fr.stable.UtilEvalError;
import com.fr.stable.exception.FormulaException;
import com.fr.stable.script.Node;

/**
 * /.
 *
 * @author richie
 * created on 2020-09-04
 */
@Open
public abstract class AbstractFunction extends CalculatorEmbeddedFunction {

    /**
     * .
     *
     * @param arguments
     * @return
     * @throws UtilEvalError
     */
    @Override
    public Object evalExpression(Node[] arguments) throws UtilEvalError {

        Object returnValue;
        if (this.getType() == HA) {
            //eval
            returnValue = tryRun(arguments);
        } else {
            Object[] args = new Object[arguments.length];
            for (int i = 0; i < arguments.length; i++) {
                args[i] = this.getCalculatorProvider().evalValue(arguments[i]);
            }
            returnValue = tryRun(args);
        }

        if (returnValue == Primitive.ERROR_VALUE || returnValue == Primitive.ERROR_NAME) {
            log(InterProviderFactory.getProvider().getLocText("Fine-Core_Base_NS_Cell_Formula") + toString());
        }

        return returnValue;
    }

    /**
     * .
     *
     * @return
     */
    @Deprecated
    public Calculator getCalculator() {
        return (Calculator) getCalculatorProvider();
    }

    private Object tryRun(Object[] args) throws UtilEvalError {

        try {
            return run(args);
        } catch (FormulaException fe) {
            throw new UtilEvalError(fe);
        }
    }

    /**
     * .
     *
     * @param args
     * @return
     * @throws FormulaException
     */
}

```

```
public abstract Object run(Object[] args) throws FormulaException;

}
```

FR	8.0		
FR	9.0		
FR	10.0		
BI	3.6		
BI	4.0		
BI	5.1		
BI	5.1.2		
BI	5.1.3		

plugin.xml

```
<extra-core>
  <!-- -->
  <FunctionGroup class="your class name"/>
  <!-- -->
  <FunctionDefineProvider class="your class name"
                        name="" description=""/>
</extra-core>
```

ExtraClassManagermountSpecificaddFunctionDefxmlclass/name/description FunctionDeffunctionsFunctionDefineProvider
DefaultNameSpaceDefaultNameSpace#getMethod
ExtraClassManagerfunctions

demo[demo-function-def-container](#)

[demo-function-fibonacci](#)

[open-JSD-7837](#)

[open-JSD-7615](#)