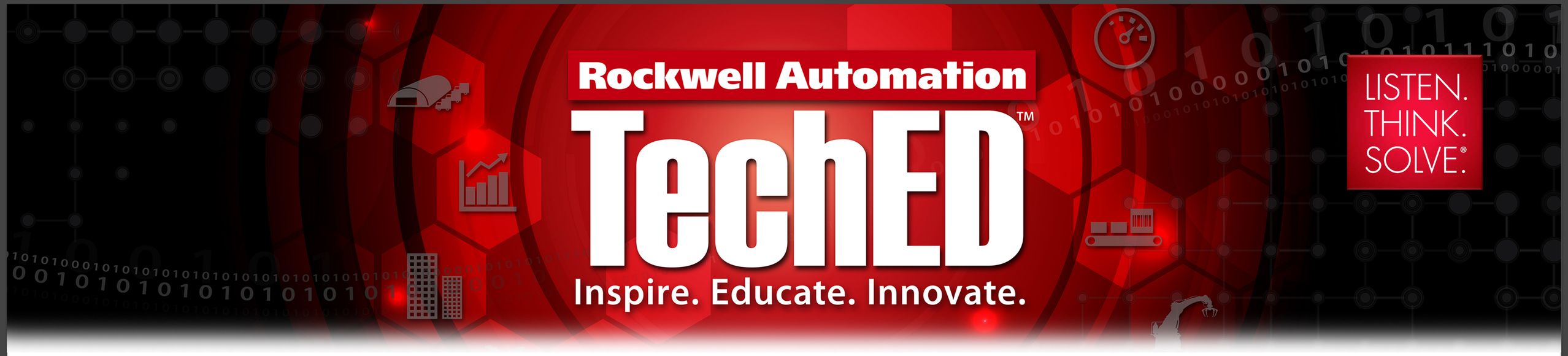


FactoryTalk ProductionCentre培训

应用开发 – 用户界面编程

ISPB (信息软件部门)
2016年9月18日

PUBLIC INFORMATION

Allen-Bradley • Rockwell Software

Rockwell Automation

课程说明

- **课程名字：**

  FTPC应用开发培训 – 用户应用编程

- **课程描述：**

  本课程介绍FactoryTalk ProductionCentre的用户应用编程。

- **目标受众：**

  业务用户，项目经理，开发人员

- **培训目标：**完成该课程之后，学员将能够
  - 了解FTPC的用户应用编程
  - 了解Pnuts编程语言

课程说明

- **课堂纪律：**
  - 关闭手机铃声，设置为静音模式
  - 举手提问、发言

内容

Rockwell Automation

- FTPC编程环境
- FTPC编程语言
- FTPC界面编程
- FTPC编程调试
- QA

FTPC编程环境

- **环境说明**

  本章介绍如何搭建基于FTPC的开发环境

- 集成开发软件
  - Java： 1.7 update 25
  - Eclipse：Eclipse IDE for Java EE Developers
    - MARS Release
- FTPC SDK
  - FTPC 9.4 MR1 SDK for JBoss
- FTPC JBoss
  - FTPC 9.4 MR1
- 数据库
  - Microsoft SQL Server 2008

FTPC编程环境

- **EDI配置说明**
  - 创建Eclipse项目
  - Debug配置
    - Main Class：

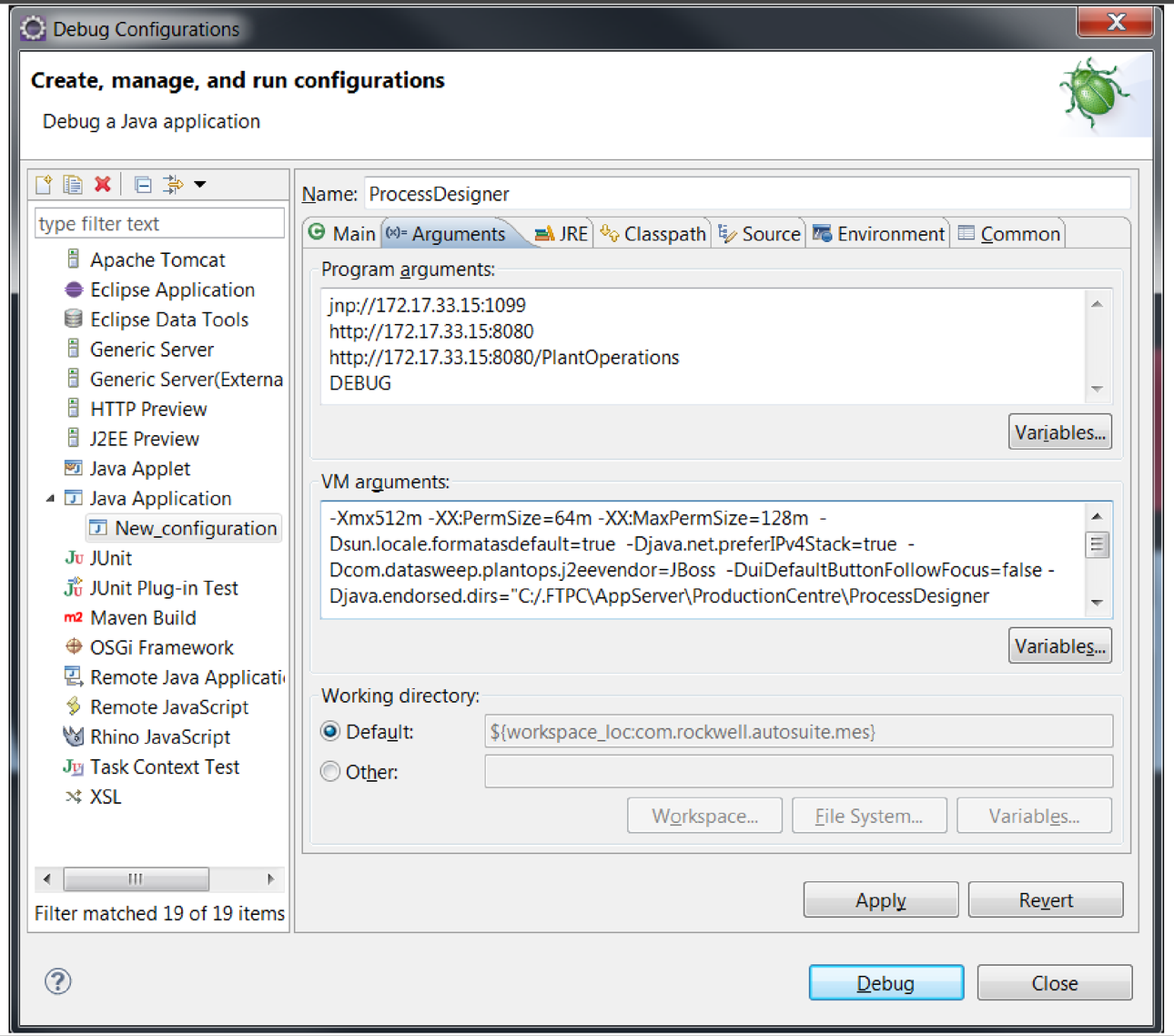
      com.datasweep.compatibility.buildtime.Buildtime

    - Arguments：
      - 设置Program Arguments

        指向开发、测试服务器

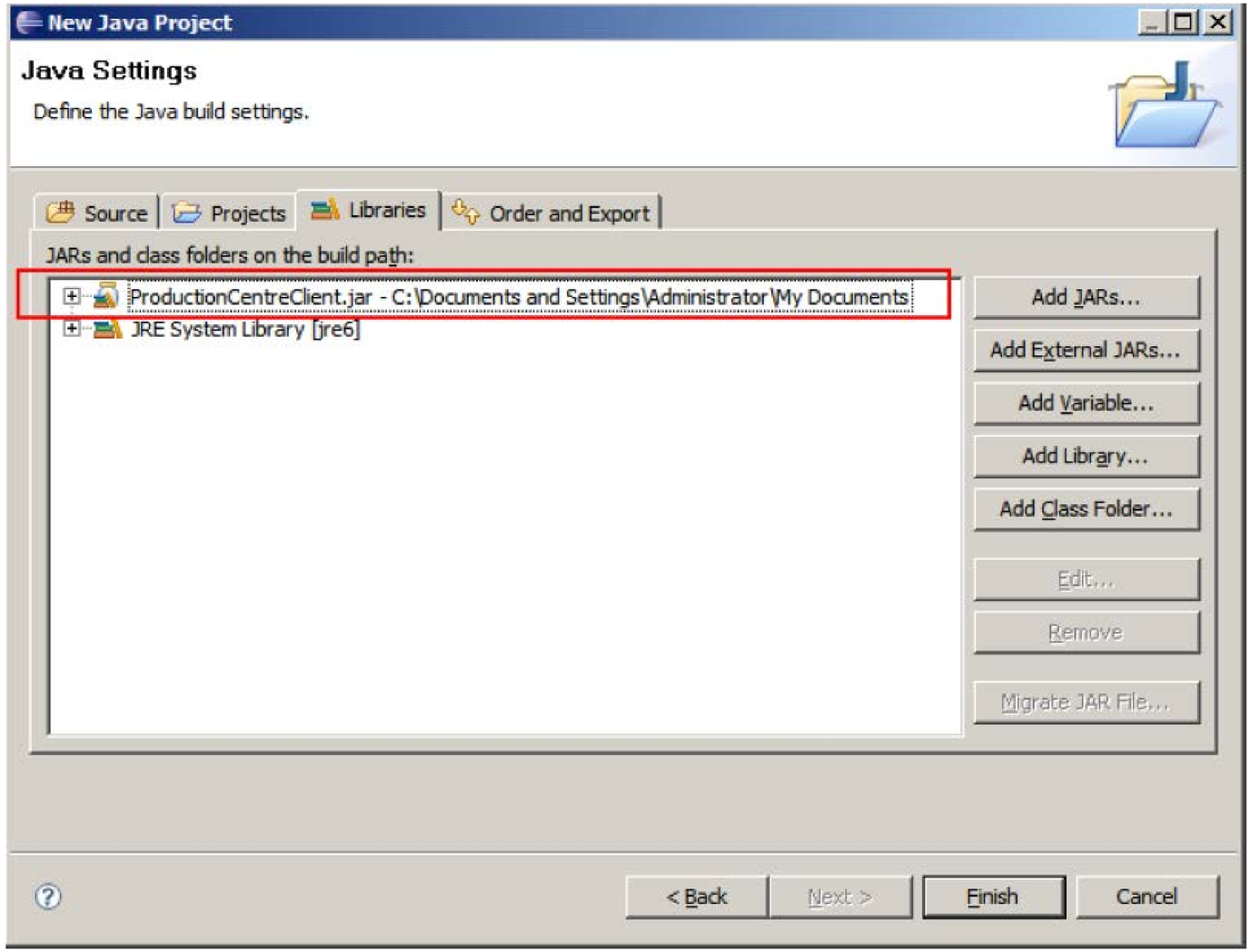
      - 设置VM Arguments
    - JRE：

      选择JavaSE-1.7（jdk1.7.0_xx)



FTPC编程编程

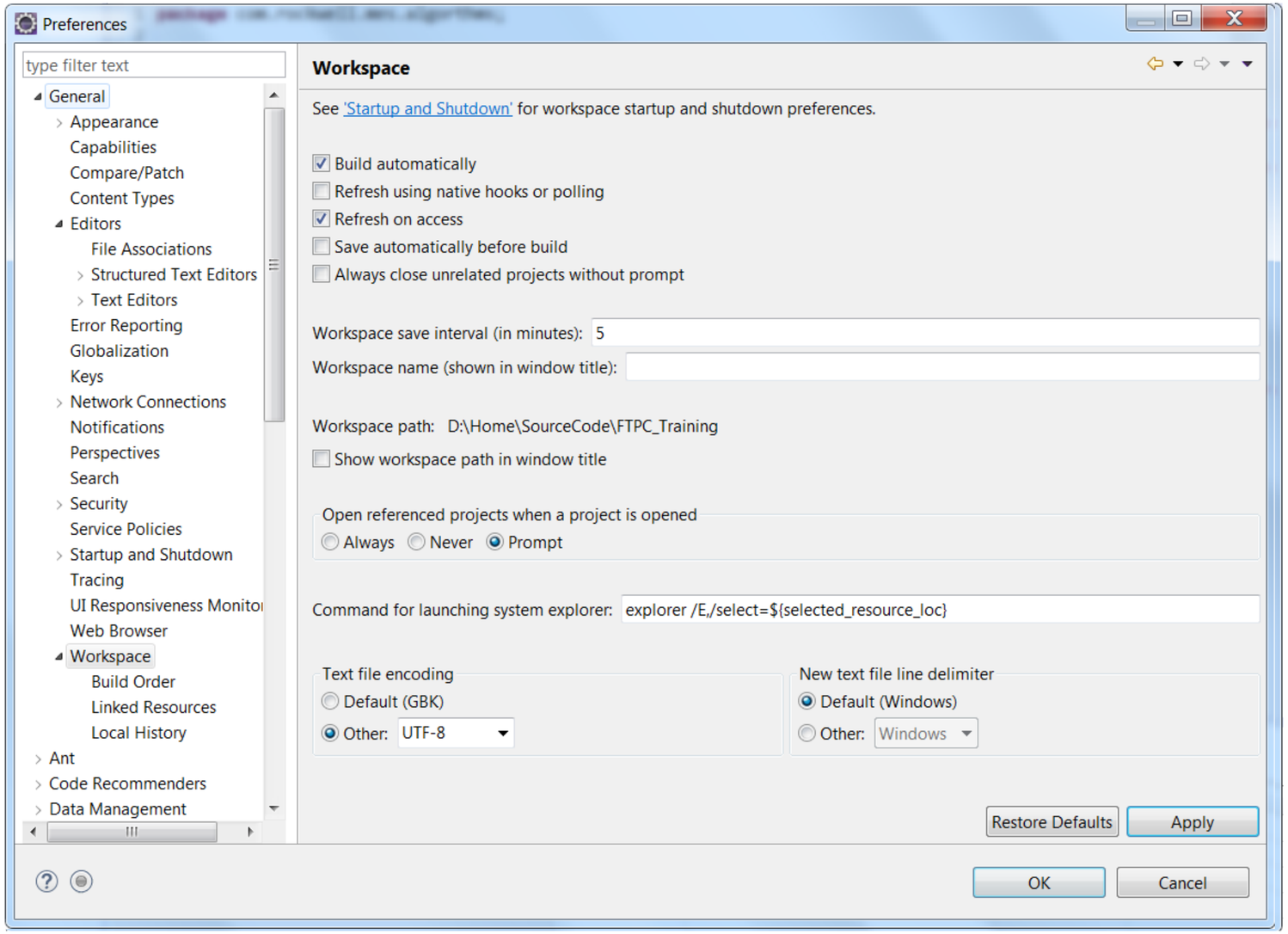
- **EDI配置说明**
  - 添加SDK



FTPC编程环境

Rockwell Automation

- **代码格式**
  - RACodeStandard.xml
  - RACodeTemplates.xml



FTPC编程环境

- **培训用环境**

应用服务器	用户	密码
http://xxx.xxx.xxx:8080/PlantOperations		

内容

Rockwell Automation

- FTPC编程环境
- FTPC编程语言
- FTPC界面编程
- FTPC编程调试
- QA

FTPC编程语言

- **Pnuts编程说明**

  Process Designer提供面向对象和基于事件驱动的脚本编程语言。用户可以使用Process Designer提供的API或Java API来构成应用程序。

  - 基于表达式（Expression Based）
  - 简单数据（Primitive Data）
  - 函数（Functions）

- **Java编程：**

  属于FTPC高级编程课程，本课程不涉及该方面的介绍。

- **.Net编程：**

  属于FTPC高级编程课程，本课程不涉及该方面的介绍。

- **Web编程：**

  属于FTPC高级编程课程，本课程不涉及该方面的介绍。

FTPC编程语言

Rockwell Automation

- Pnuts**基础知识（1）**

预留关键字

if	else	while	for	foreach	switch	case	default	break
null	continue	return	true	false	instanceof	class	function	macro
catch								

注释

```
// text
/* text */
```

特殊字符

Character	Meaning
\'	'
\n	LF
\t	TAB
\\	\
\0	NULL
\f	^L
\b	^H

支持Unicode转义

'\u0041' ==> 'A'

FTPC编程语言

Rockwell Automation

- Pnuts**基础知识（2）**

- 内置函数

import (<className>)	throw(<exception or string>)	catch(<exceptionClass>, <function>)	class(<class_name>)
import()	defined(<symbol>)	hex(Object integerOrByteArray)	println(Object objects, ...)
sleep(int msec)	CreateObject(String ProgId)	print(Object objects, ...)	quit()

```
import ("com.datasweep.compatibility.ui.*")
import ("java.io.*")

function exceptionHandler(e)
{
    println(e.getMessage()) // Output is "ThereAreProblems"
}
catch(RuntimeException, exceptionHandler)

throw("ThereAreProblems")


 //a evaluates to 310e
 a = hex(12558)


 println("UnitSaved")
```

```
import("com.datasweep.compatibility.ui.Color")
ctrlStartButton.setBackColor(Color::RED)

// declare the most generic exception class first
catch(class java.lang.Exception, catchGeneric)
// declare more specific exception classes after
catch(class java.lang.NumberFormatException, catchConvert)
catch(class java.lang.ArithmeticException, catchCalc)


 ok = defined("X") //ok evaluates to false

 X = 100
 ok = defined("X") //ok evaluates to true

 X = null
 ok = defined("X") //ok evaluates to true
```

说明：
define（ ）函数不支持本地变量，不能在函数中调用

FTPC编程语言

- PNuts**基础知识（3）**
- 附加模块（打包在pnuts-modules.jar)
  - pnuts.io: 定义I/O 函数。其不支持HTML模式。
  - pnuts.lib: 定义公共类库行数。
  - pnuts.math: 定义支持在java.lang.Math类中定义的基本数字操作符函数。
  - pnuts.regex: 提供正则表达式函数。
  - pnuts.text: 定义文本I/O操作的函数。
  - pnuts.util: 定义其他使用标准Java API的函数。
  - pnuts.xml: 提供XML文档支持函数。
- 扩展模块
  - vbscript:
  - win32com.jacob:

FTPC编程语言

- Process Designer API（1）

函数方法

```
vecAllUnits = getAllUnits()
```

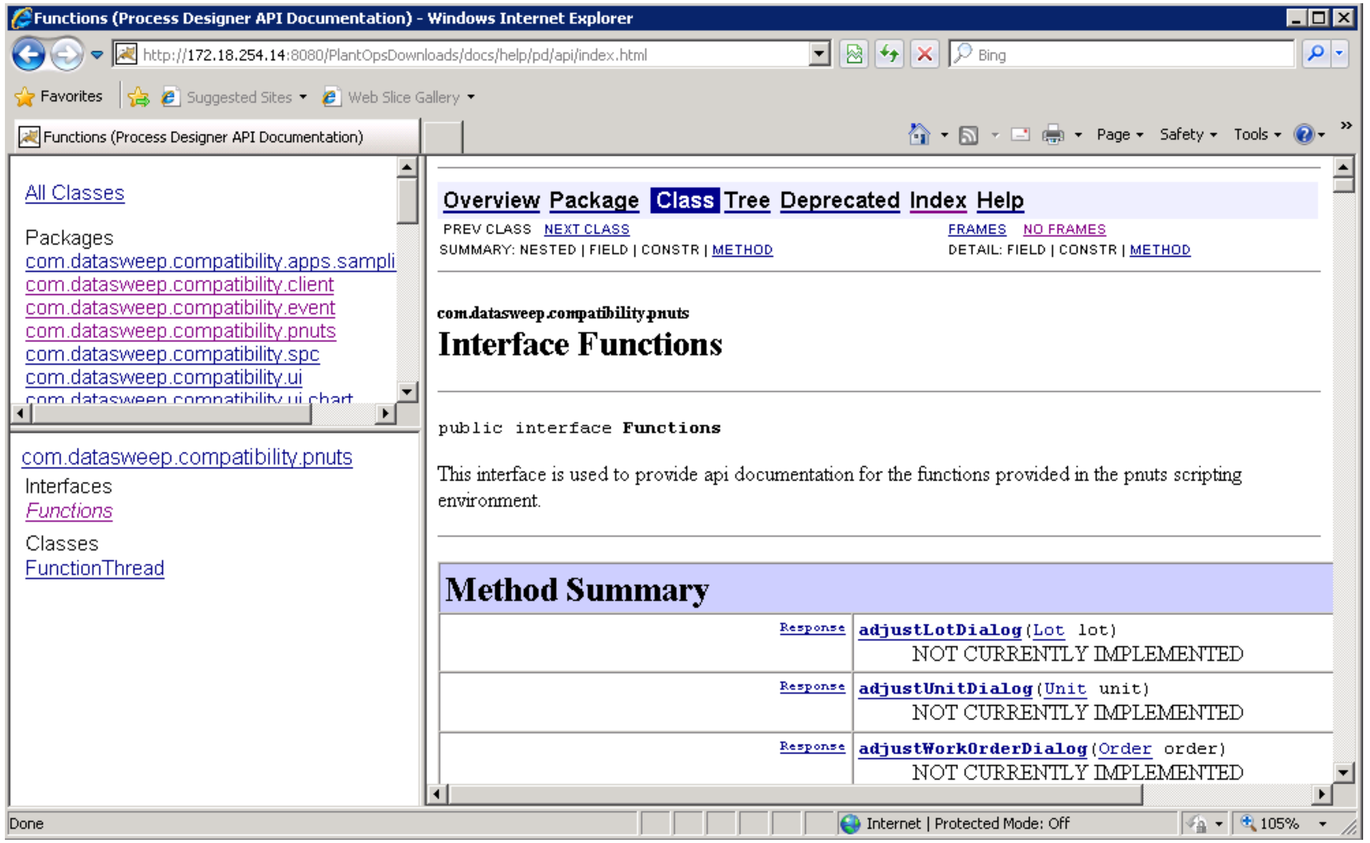
对象方法

```
unit.finish("Finished on Route A")
```

函数定义

function <*identifier*> (<*argument*>, ...) { <*expression-block*> }
function <*identifier*> (<*argument* '[' ']'>) { <*expression-block*> }

```
function topLevel() {
      dialogError("You have run the Top Level function")
      function secondLevel(){
            dialogError("You have run the second Level function")
            function thirdLevel(){
                  dialogError("You have run the third Level function")
            }
      }
}
```



FTPC编程语言

- Process Designer API（2）

子程序（Subroutine）

importSubroutine(...)

createGlobalFunctions(...)

```
subroutine(String name)
subroutine(String name, Object arg1)
subroutine(String name, Object arg1, Object arg2)
subroutine(String name, Object arg1, Object arg2, Object arg3)
subroutine(String name, Object arg1, Object arg2, Object arg3, Object arg4)
```

声明变量

<identifier> = <expression>

<expression> [<expression>] = <expression>

```
newBoolean = true
newChar = 'A'
newString = "abc123"
newInt = -1
newDouble = 4.2345e+3
newNull = null
```

```
arrArray1 = createArray(2)
arrArray1 = ["Cat",2]
arrArray1[0] = "Dog"
```

<class> :: <identifier>

使用对象字段（静态字段）

```
import("com.datasweep.compatibility.ui.Color")
ctrlStartButton.setBackColor(Color::RED)
```

FTPC编程语言

- Process Designer API（2）

子程序（Subroutine）

importSubroutine(...)　　createGlobalFunctions(...)

```
subroutine(String name)
subroutine(String name, Object arg1)
subroutine(String name, Object arg1, Object arg2)
subroutine(String name, Object arg1, Object arg2, Object arg3)
subroutine(String name, Object arg1, Object arg2, Object arg3, Object arg4)
```

声明变量

<identifier> = <expression>

<expression> [<expression>] = <expression>

```
newBoolean = true
newChar = 'A'
newString = "abc123"
newInt = -1
newDouble = 4.2345e+3
newNull = null
```

<class> :: <identifier>

```
arrArray1 = createArray(2)
arrArray1 = ["Cat",2]
arrArray1[0] = "Dog"
```

```
arrArray1 = createArray(1)
arrArray1[0] = 1.1

arrArray2 = createArray(2)
arrArray2[0] = "Dog"
arrArray2[1] = 9

arrMyArray = arrArray1 + arrArray2

length = arrArray1.length
```

```
arrArray1 = createArray(4)
arrArray1 = ["Cat",2,1.1,"Dog"]
animalType = arrArray1[0]
moreValues = arrArray1[1..]
lastValues = arrArray1[2..3]

 arrArray1 = createArray(2)
 arrArray1[0] = 1.1
 arrArray1[1] = "Dog"

 arrArray2 = createArray(2)
 arrArray2[0] = "0"
 arrArray2[1] = "1"

 arrNewArray = createArray(2)
 arrNewArray[0] = arrArray1
 arrNewArray[1] = arrArray2
```

使用对象字段（静态字段）

```
import("com.datasweep.compatibility.ui.Color")
ctrlStartButton.setBackColor(Color::RED)
```

FTPC编程语言

- **数据类型（1）**
  - Process Designer支持的数据类型

Type	Description	Example
char	A character represents a java.lang.Character object. A char must be enclosed by single quotes. A unicode escape sequence can be used in a Character Literal.	newChar = 'A' newChar = '\u0041'
String	A string is a sequence of characters enclosed by double quotes. A unicode escape sequence can be used as part of a String Literal.	newString = "abc" newString = "123" newString = "\u0041B"
int	Integers are any whole number and can be positive, zero, or negative. An integer is not restricted within *long* precision. The class of the number object is assigned automatically depending on the value: • When the integer is between Integer.MAX_VALUE and Integer.MIN_VALUE, the class is java.lang.Integer. • When the integer is between Long.MAX_VALUE and Integer.MAX_VALUE or Long.MIN_VALUE and Integer.MIN_VALUE, the class is java.lang.Long. • Otherwise, the number is a java.math.BigInteger object.	newInt = -54 newInt = 0 newInt = 54
double	A floating-point number is any number represented with a decimal point numeric. The number can be written either with a decimal point or in scientific notation--an "e" (uppercase or lowercase) to represent "times ten to the power of". There are two types of floating-point numbers: Float and Double.	newDouble = -1.2345 newDouble = 4.2345e+3 newDouble = 3.1
Boolean	Contains either true or false.	newBoolean = true newBoolean = false
null	Contains no valid data.	newNull = null

Hexadecimal Integer	Allows you to express a number as a hexadecimal rather than an int or long. Use the following characters:
	"#" (["0"-"9","a"-"f","A"-"F"]) + "0" ("x"|"x") (["0"-"9","a"-"f","A"-"F"])+ (...)
	When the number is between Byte.MAX_VALUE and Byte.MIN_VALUE, "#ff" makes a *byte* object while "0xff" makes an *int* object.
		x = #ffffff00
		//x evaluates to 268435200 - an int
		
		x = #ffffff0000
		//x evaluates to 68719411200 - a long
		
		x = 0xff
		//x evaluates to 255 - an int
		
		x = #ff
		//x evaluates to -1 - a byte

Multi-precision Decimal Number	Multi-precision decimal numbers are represented by java.math.BigDecimal objects. The value type can be indicated using the symbol B.
	x = 123.0e-4 //returns 0.0123
		c = 1.2B*2
		//c evaluates to 2.4 and is a java.math.BigDecimal object
		c = 123.0B
		//c evaluates to 123.0 and is a java.math.BigDecimal object
		c = 123.0
		//c evaluates to 123.0 and is a double

FTPC编程语言

Rockwell Automation

- **数据类型（2）**

数据类型转换

- toString()
- toHexString()
- toOctalString()
- toCharArray()
- toBinaryString()
- doubleToLongBits()
- floatToIntBits()
- stringToInt()
- stringToDouble()
- stringToBoolean()
- stringToBigDecimal()

<primitive> (<expression>)

```
int("1") ==> 1
int('1') ==> 49

char(65) ==> 'A'
```

(<type>) <variablename>

```
a = 1.2290875
b = (int)a
```

< String > + < String >

```
strLetter = "ABC"
strNumber = "123"
strCombo = strLetter + strNumber
```

数字转换

	BigDecimal	**Double**	**Integer**	**Character**	**Byte**
BigDecimal	BigDecimal	BigDecimal	BigDecimal	BigDecimal	BigDecimal
Double	BigDecimal	Double	Double	Double	Double
Integer	BigDecimal	Double	*see Note*	*see Note*	*see Note*
Character	BigDecimal	Double	*see Note*	Integer	Integer
Byte	BigDecimal	Double	*see Note*	Integer	Integer

Note:

- When the integer is between Integer.MAX_VALUE and Integer.MIN_VALUE, the class is java.lang.Integer.
- When the integer is between Long.MAX_VALUE and Integer.MAX_VALUE or Long.MIN_VALUE and Integer.MIN_VALUE, the class is java.lang.Long.
- Otherwise, the number is a java.math.BigInteger object.

FTPC编程语言

Rockwell Automation

- **条件表达式** (1)

- if-else

```
if ( < boolean expression1 > ) {
    < expression-block1 >
} else if ( < boolean expression2 > ) {
    < expression-block2 >
...
} else {
    < expression-block >
}
```

```
newArray = [1, 2, 3]
if (newArray.length > 5){
    arraySize = "Large"
} else if (newArray.length > 4){
    arraySize = "Medium"
} else {
    arraySize = "Small"
}
```

- foreach

```
foreach < identifier > < array-expression > < expression-block >
foreach < identifier > ( < array > ) < expression-block >
foreach < identifier > ( < java.util.Enumeration > ) < expression-block >
foreach < identifier > ( < java.util.Iterator > ) < expression-block >
```

```
sum = 0
foreach i ([1, 2, 3]) {
  sum = sum + i
}
```

- while

```
while ( < boolean expression > ) < expression-block >
```

```
sum = 5
while (sum > 0){
    sum = sum - 1
}
```

FTPC编程语言

- **条件表达式** (2)

- for

for ([< *identifier* > = < *expression* > , …] ; [< *boolean expression* >] ; [< *iteration expression* > , …]) < *expression-block* >

```
j = 0
for (i = 0; i < 5; ++i){
    j = j + i
}
```

- switch

```
switch ( < expression > ) {
 case < expression > : < expression-block >
   …
 default : < expression-block >
}
```

```
month = 8
switch (month) {
    case 1:  println("January"); break
    case 2:  println("February"); break
    case 3:  println("March"); break
    case 4:  println("April"); break
    case 5:  println("May"); break
    case 6:  println("June"); break
    case 7:  println("July"); break
    case 8:  println("August"); break
    case 9:  println("September"); break
    case 10: println("October"); break
    case 11: println("November"); break
    case 12: println("December")
}
```

- break

```
for (i = 0; i < 5; i++){
    if (i == 2){
        break // exits the loop when i = 2
    }
    println(i)
}
```

FTPC编程语言

Rockwell Automation

- **运算符**
  - 数学运算
  - 逻辑运算
  - 位运算
  - 赋值运算
  - 比较运算

Computational		Logical		Bitwise		Assignment		Comparison	
Decription	Symbol	Description	Symbol	Description	Symbol	Description	Symbol	Description	Symbol
Addition	+	Logical NOT	!	Bitwise complement	~	Assignment	=	Less than	<
Subtraction	-	Logical AND	&&	Bitwise Left Shift	<<	A = A + B	+=	Greater than	>
Multiplication	*	Logical OR	||	Bitwise Right Shift	>>	A = A - B	-=	Less than or equal to	<=
Division	/			Unsigned Right Shift	>>>	A = A * B	*=	Greater than or equal to	>=
Modulus arithmetic	%			Bitwise AND	&	A = A / B	/=	Equality	==
Increment	++			Bitwise XOR	^	A = A % B	%=	Inequality	!=
Decrement	--			Bitwise OR	|	A = A << B	<<=		
						A = A >> B	>>=		
						A = A >>> B	>>>=		
						A = A & B	&=		
						A = A ^ B	^=		
						A = A | B	|=		

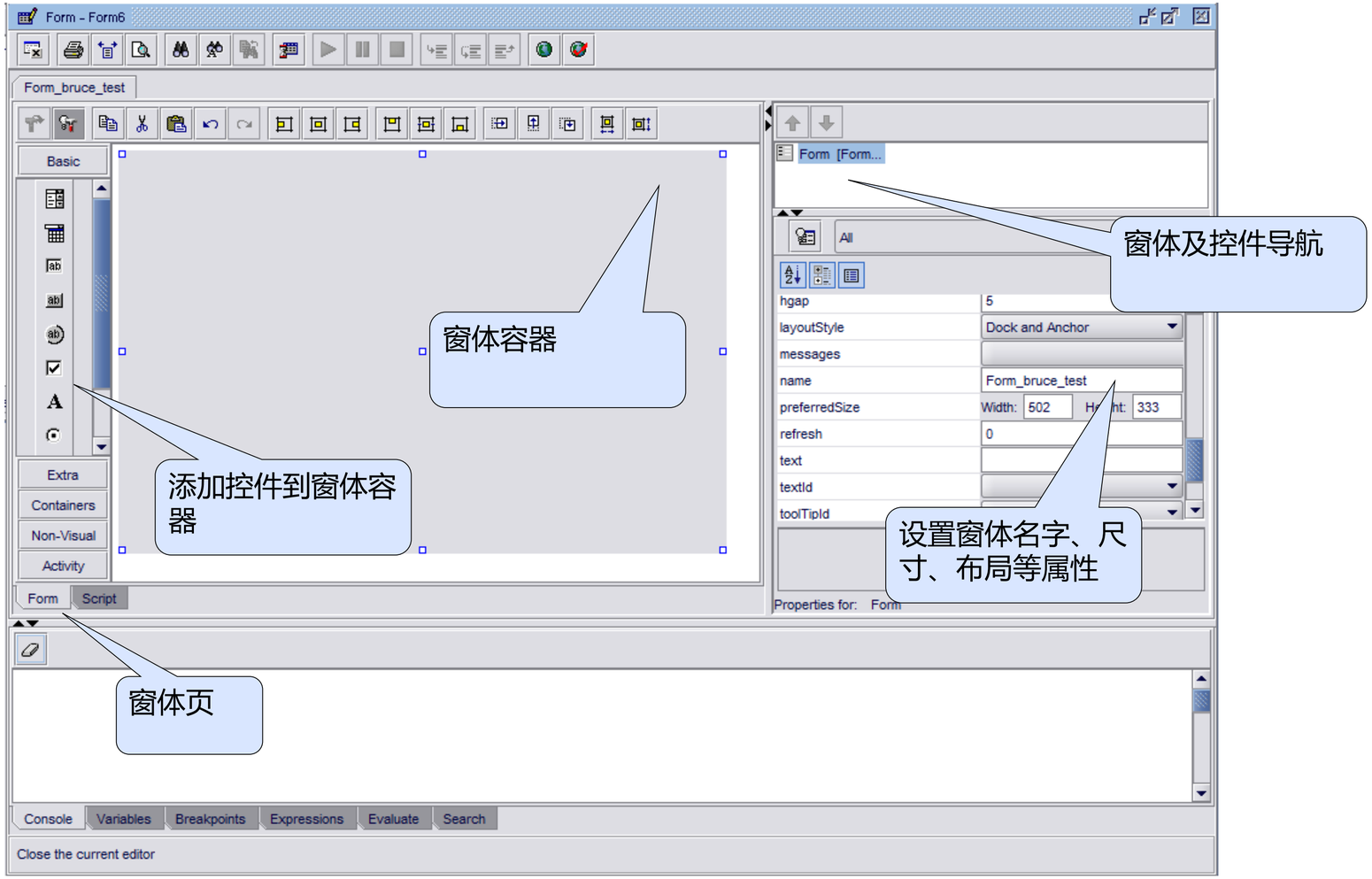
内容

Rockwell Automation

- FTPC编程环境
- FTPC编程语言
- **FTPC界面编程**
- FTPC编程调试
- QA

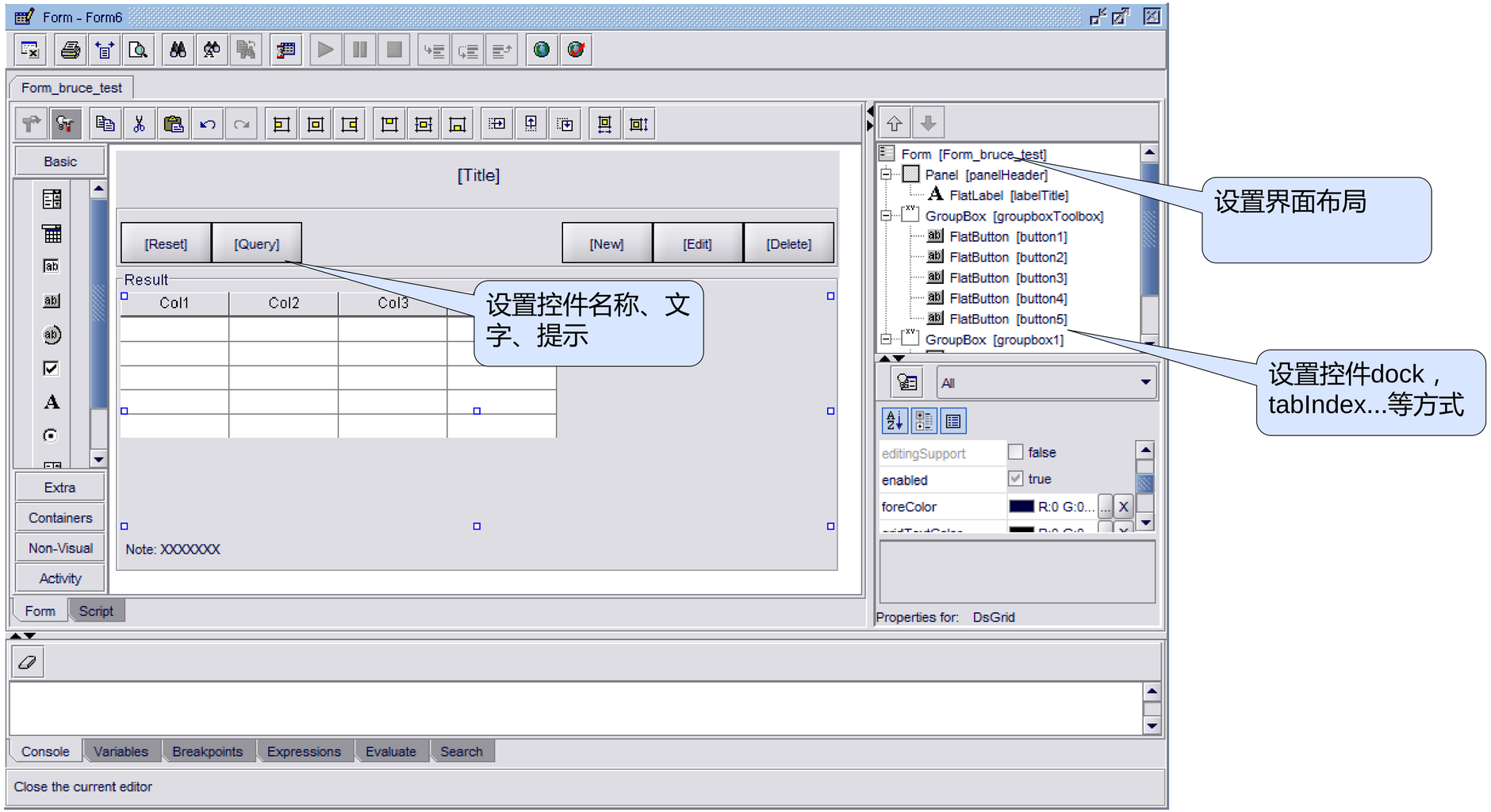
FTPC界面编程

- **创建**Form



FTPC界面编程

- **添加Form控件**



FTPC界面编程

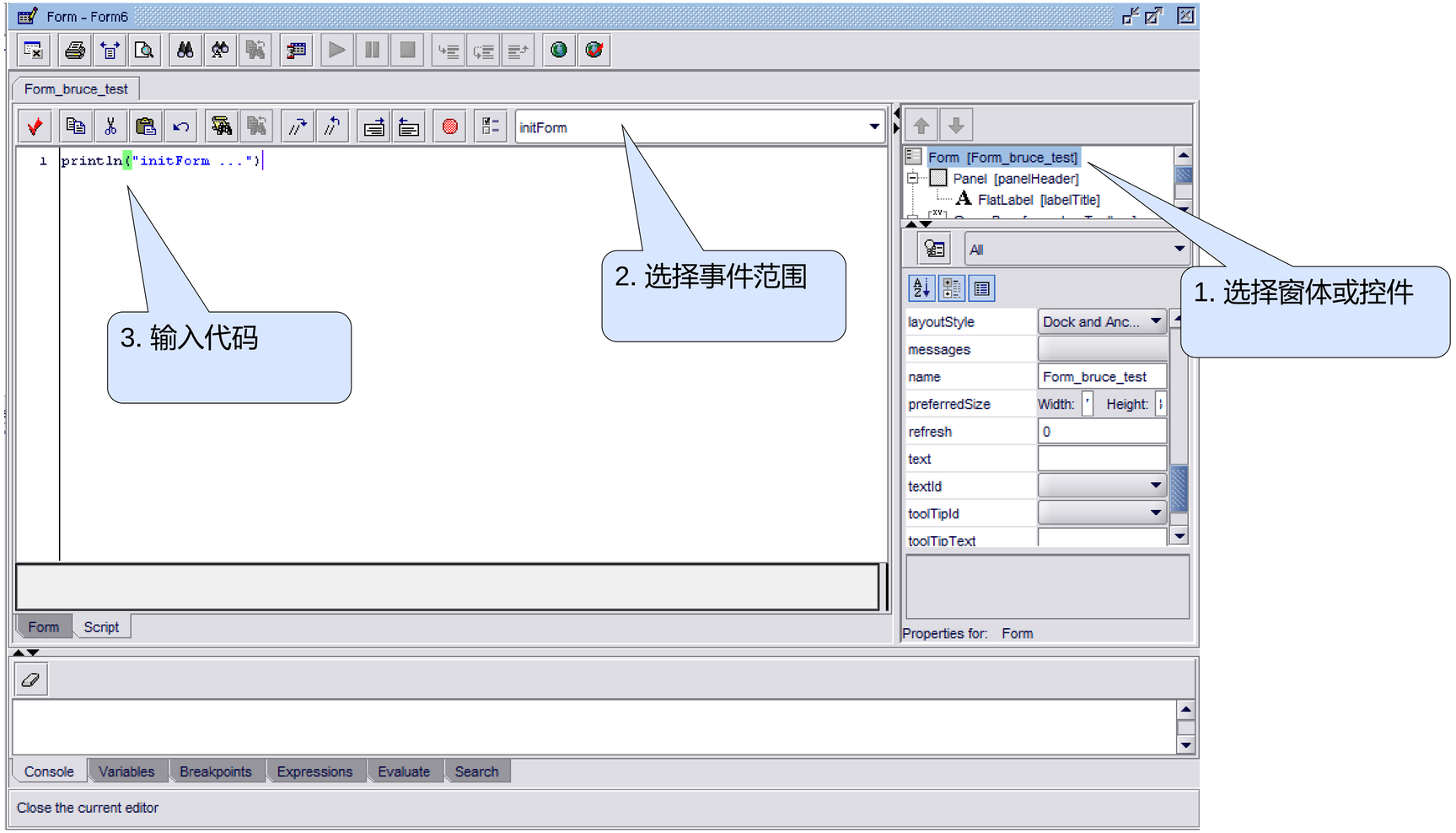
Rockwell Automation

- **添加界面事件代码（1）**

Event	Description
activate	Runs the script when the form gains focus (brought to the front).
background	Runs the script in a second thread.
click	Runs the script when the user clicks the form.
closeForm	Runs the script when the user closes the form.
deactivate	Runs the script when the form loses focus.
F2-9, F11-F12	Runs the script when the user presses the corresponding function key.
formPropertyChanged	Runs the script when a form property has changed.
functions	Contains common functions that can be used by other scripts in the form. This allows you to share scripts between controls on the form.
globalPropertyChanged	Runs the script when a global property has changed.
initForm	Runs the script when the form is first opened.
subroutines*	If you associate subroutines with a form, then the subroutines will appear on the event drop-down. You can then view the subroutine or search for the subroutine within the associated form. You must call the subroutine() method to run it.
user1-5	Allows you to break up scripts into smaller elements to simplify coding and debugging. You can call these scripts from other scripts in the form.
userChanged	Runs the script when the user has changed.
* Each subroutine name appears as a separate item in the list of events.	

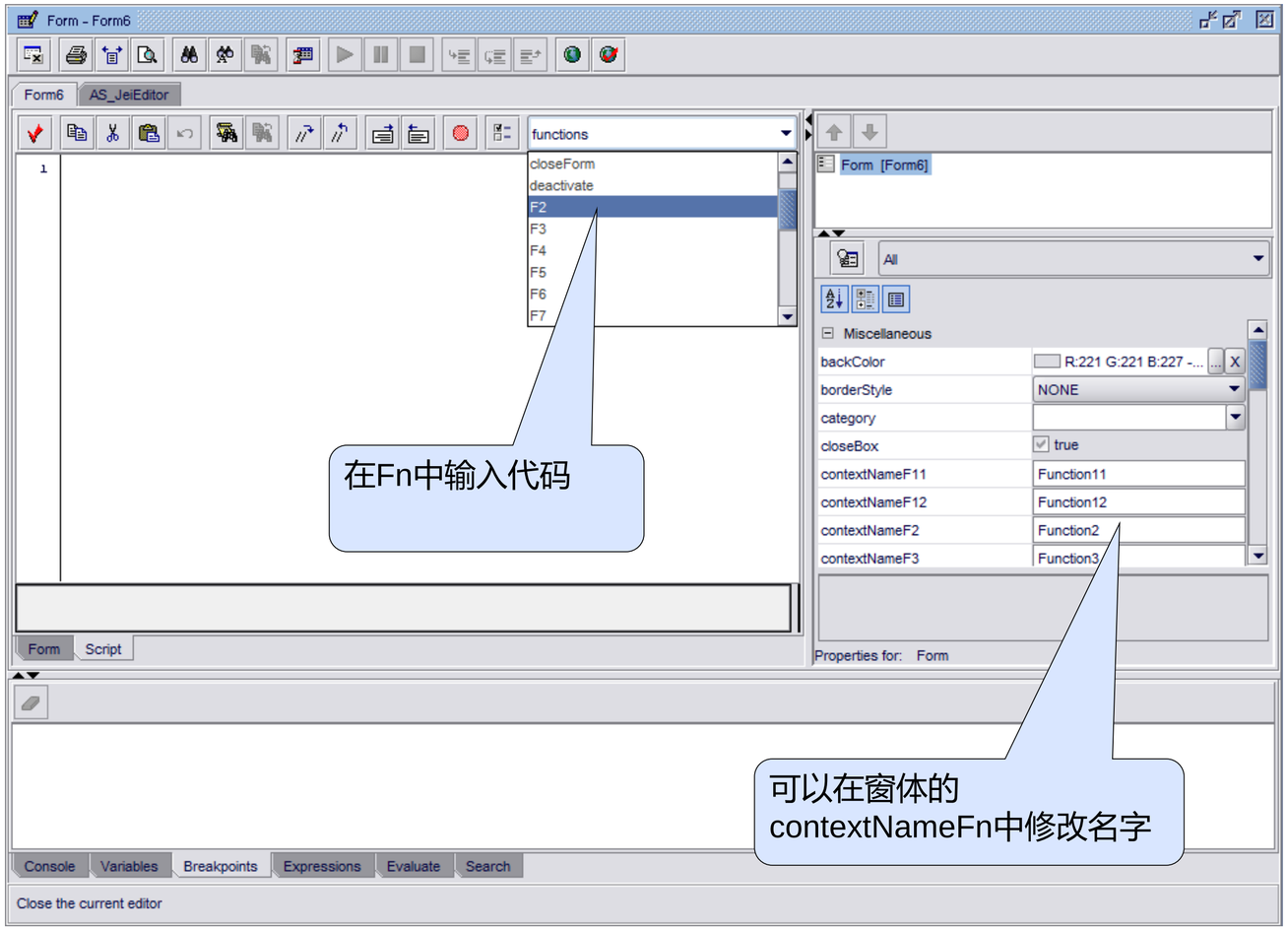
FTPC界面编程

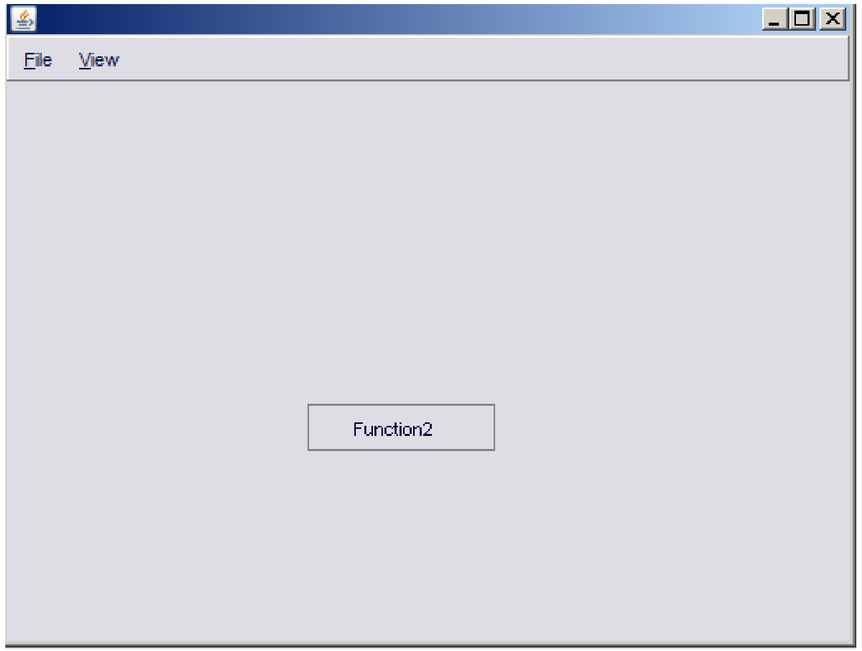
- **添加界面事件代码（2）**



FTPC界面编程

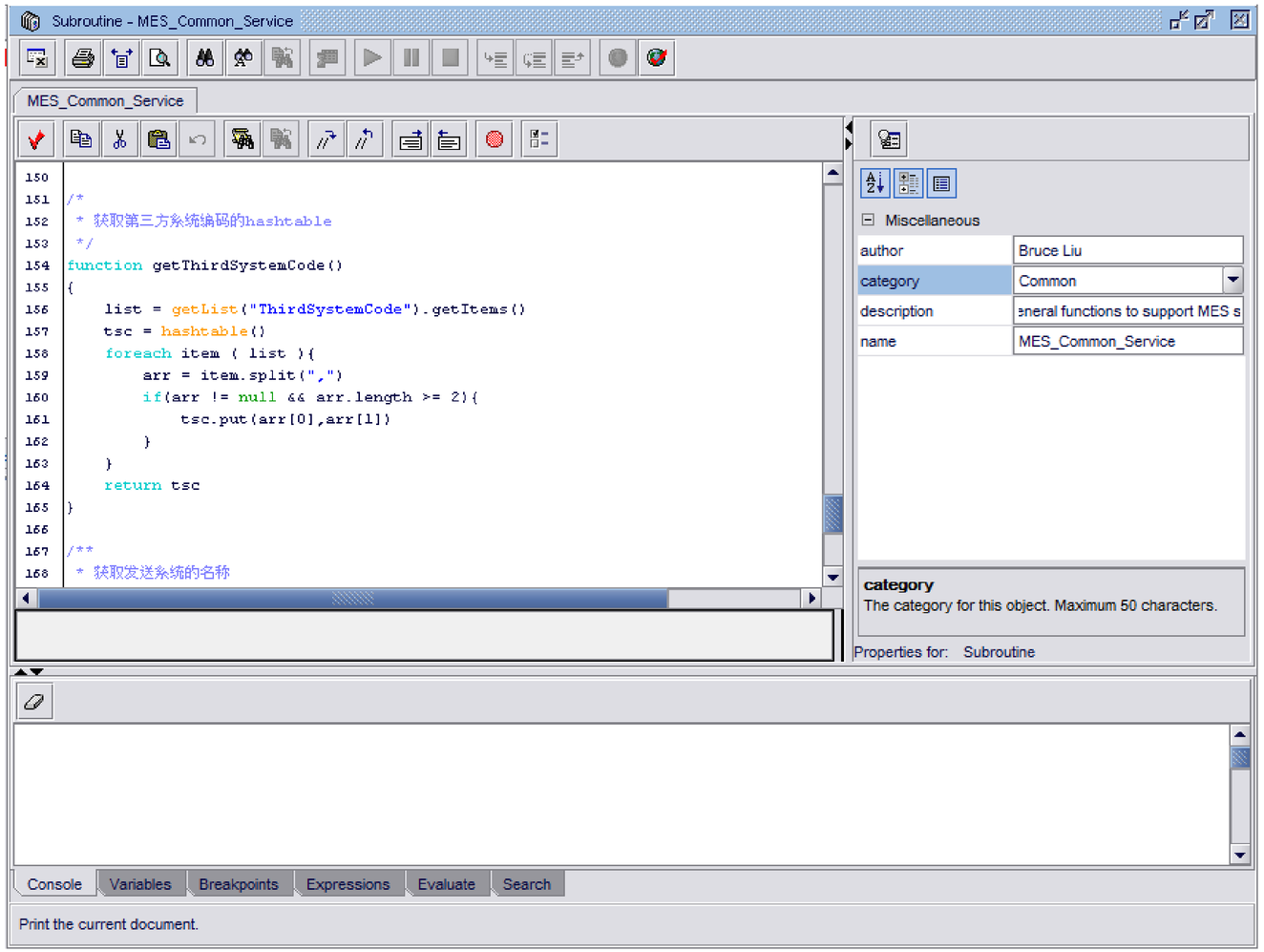
- **创建快捷菜单**





FTPC界面编程

- **创建子程序** Subroutine

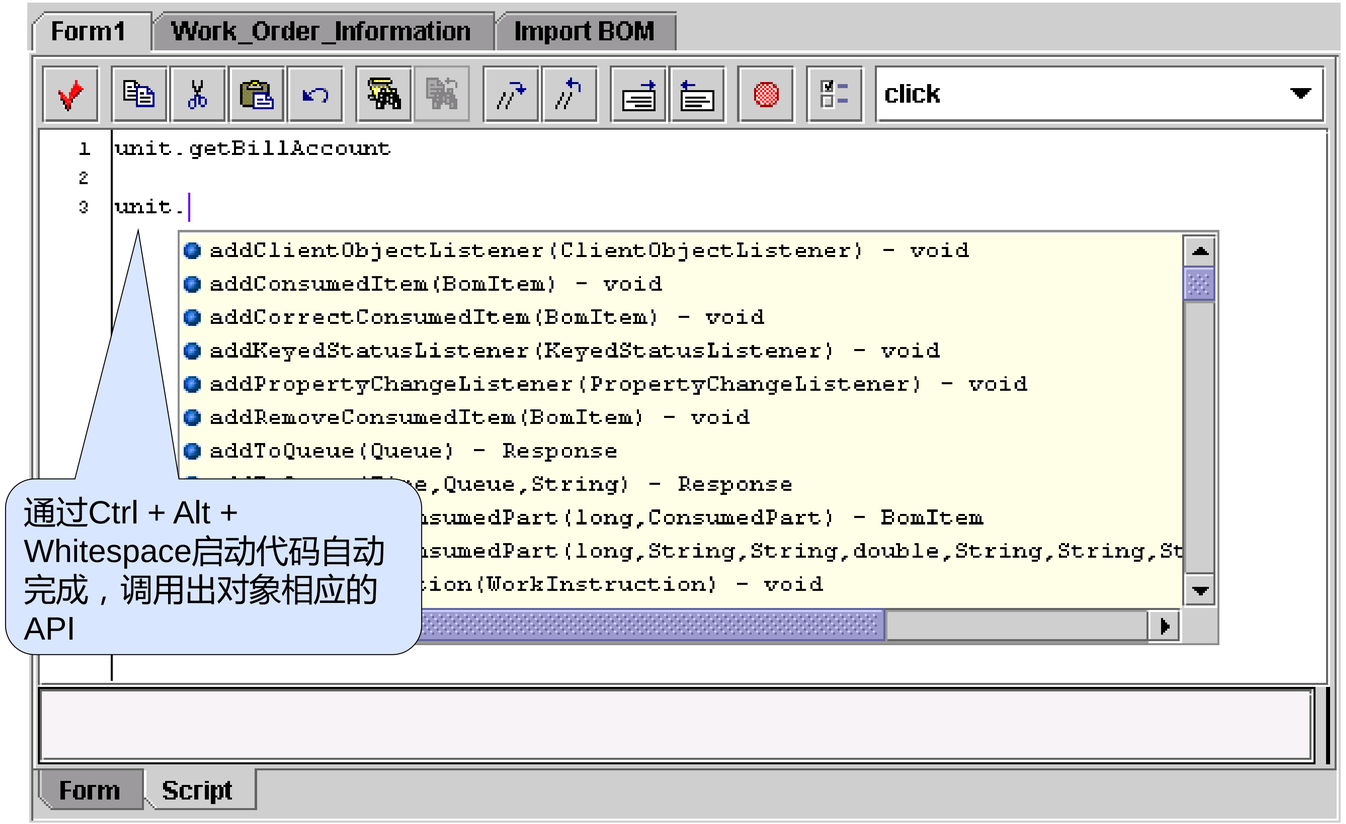


importSubroutine(...)

```
subroutine(String name)
subroutine(String name, Object arg1)
subroutine(String name, Object arg1, Object arg2)
subroutine(String name, Object arg1, Object arg2, Object arg3)
subroutine(String name, Object arg1, Object arg2, Object arg3, Object arg4)
```

FTPC界面编程

- **成员帮助** Member Help



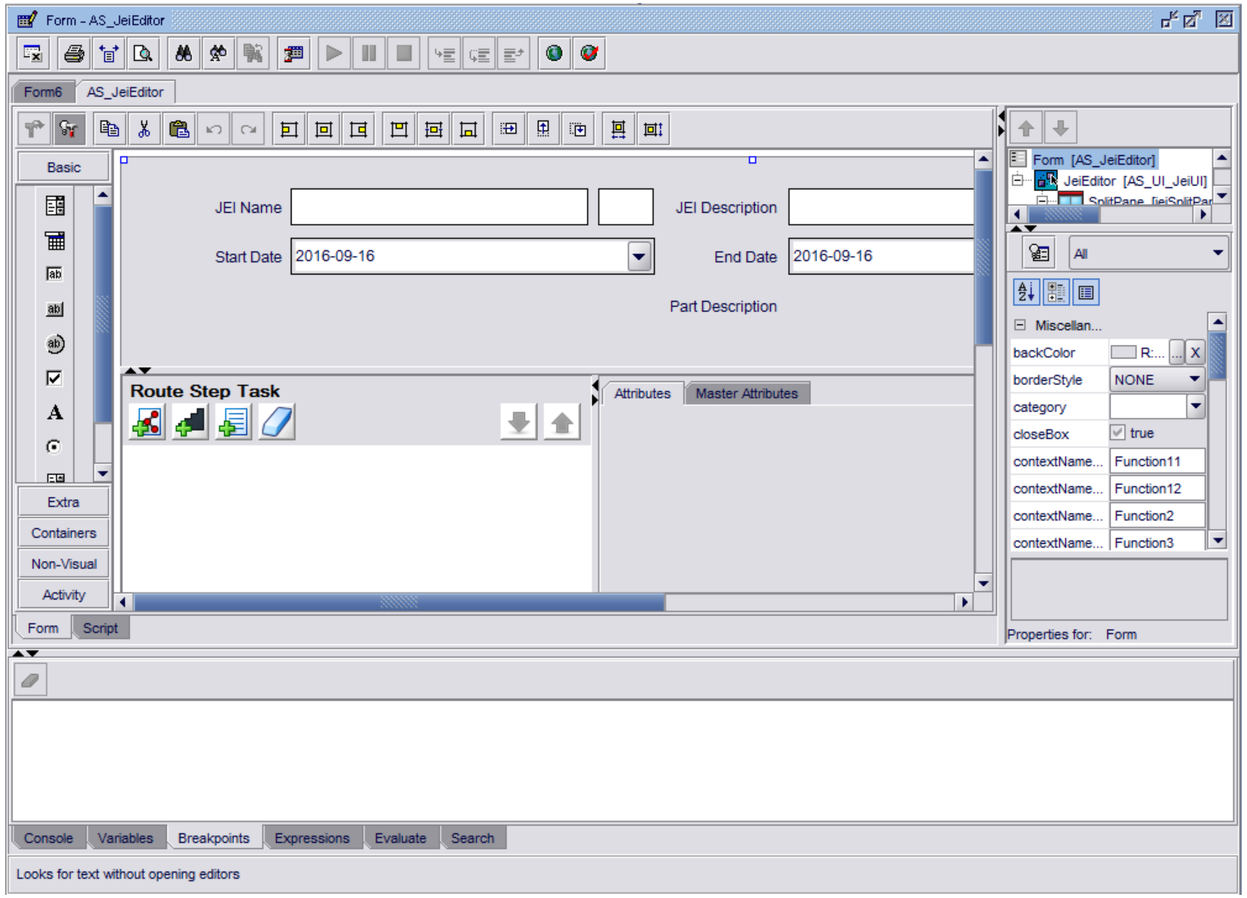
内容

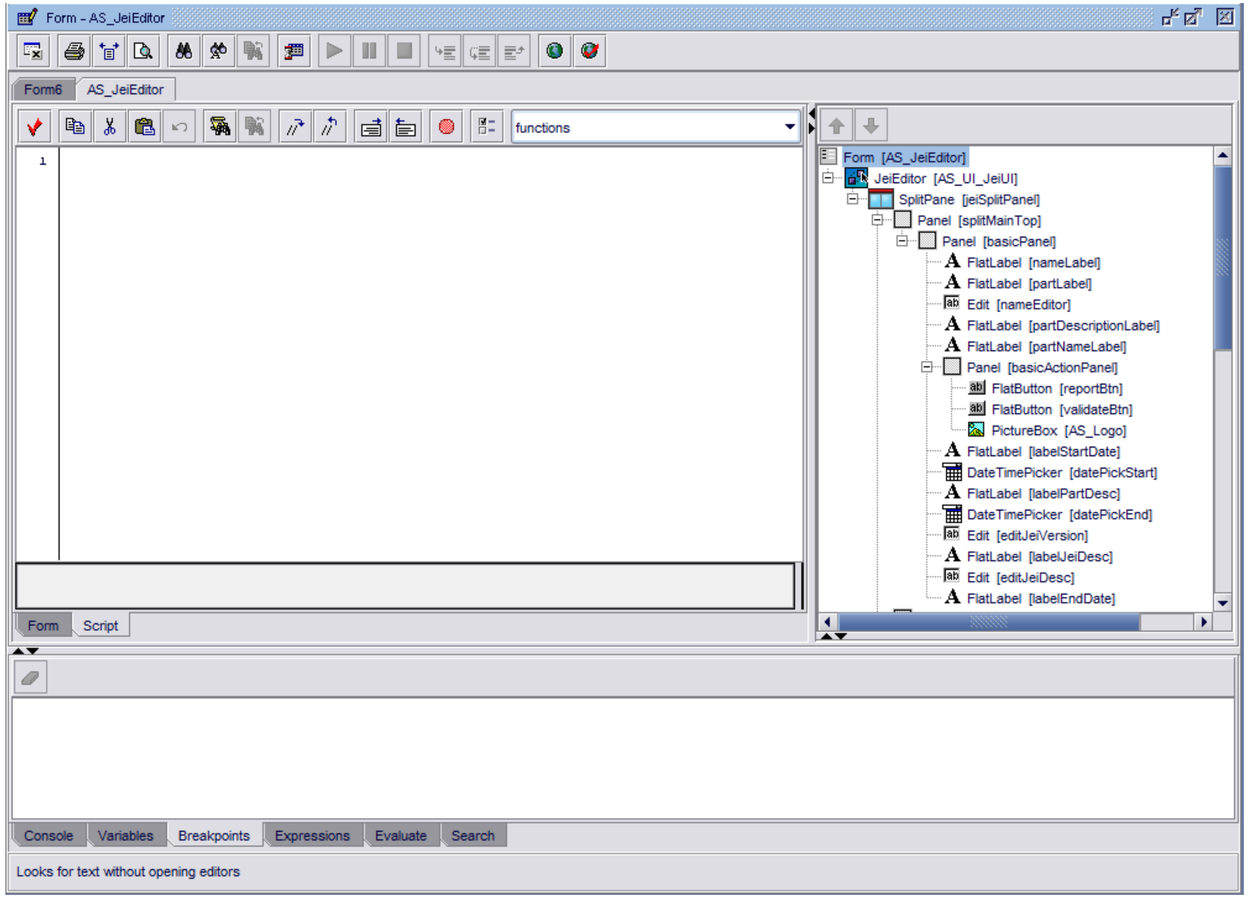
Rockwell Automation

- FTPC编程环境
- FTPC编程语言
- FTPC界面编程
- **FTPC编程调试**
- QA

FTPC编程调试

- Process Designer**调试器**





FTPC编程调试

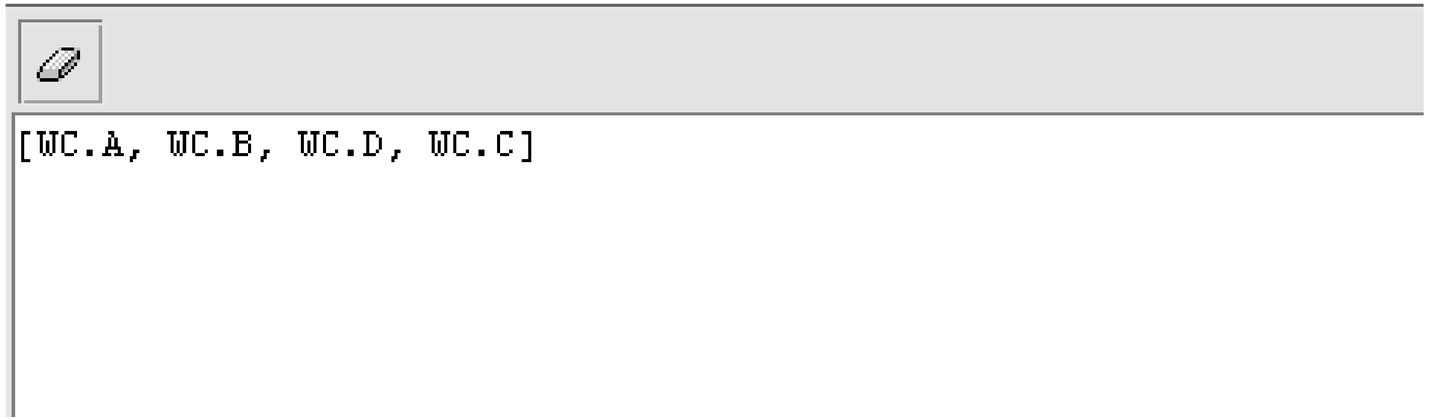
- Process Designer**调试器**

Debugger Toolbar Commands	
	Continue Execution: Continue running the form or event sheet from the current line.
	Break Execution: Pauses the form or event sheet but does not close the form's window.
	Abort Execution: Stop the form or event sheet and close the form's window.
	Toggle Break Point: Toggle on or off a break point at the current line the cursor is on. You can also add or remove break points by clicking in the grey margin where you want them.
	Step Into: Execute the current line and display at the next executable line of code. If the current line calls a function or subroutine, the next executable line will be the first line in that function/subroutine. If you use the Step Into button while in a function/subroutine, then the script debugger will execute the current line and display at the next executable line of code in that function/subroutine.
	Step Over: Execute the current line, and then continue to the next executable line of code. If the line of code is a function call or importSubroutine() statement, then the Script Debugger will execute the function/subroutine *without stepping into it* and display at the next executable line of code after the function/subroutine call. If the function/subroutine contains a breakpoint, then the normal behavior is overridden, and the Script Debugger will step into the function/subroutine and display at that breakpoint only.
	Step Up: Step out of the current function or subroutine and to the next line after the one that called the function or subroutine.

FTPC编程调试

- Process Designer**控制台**

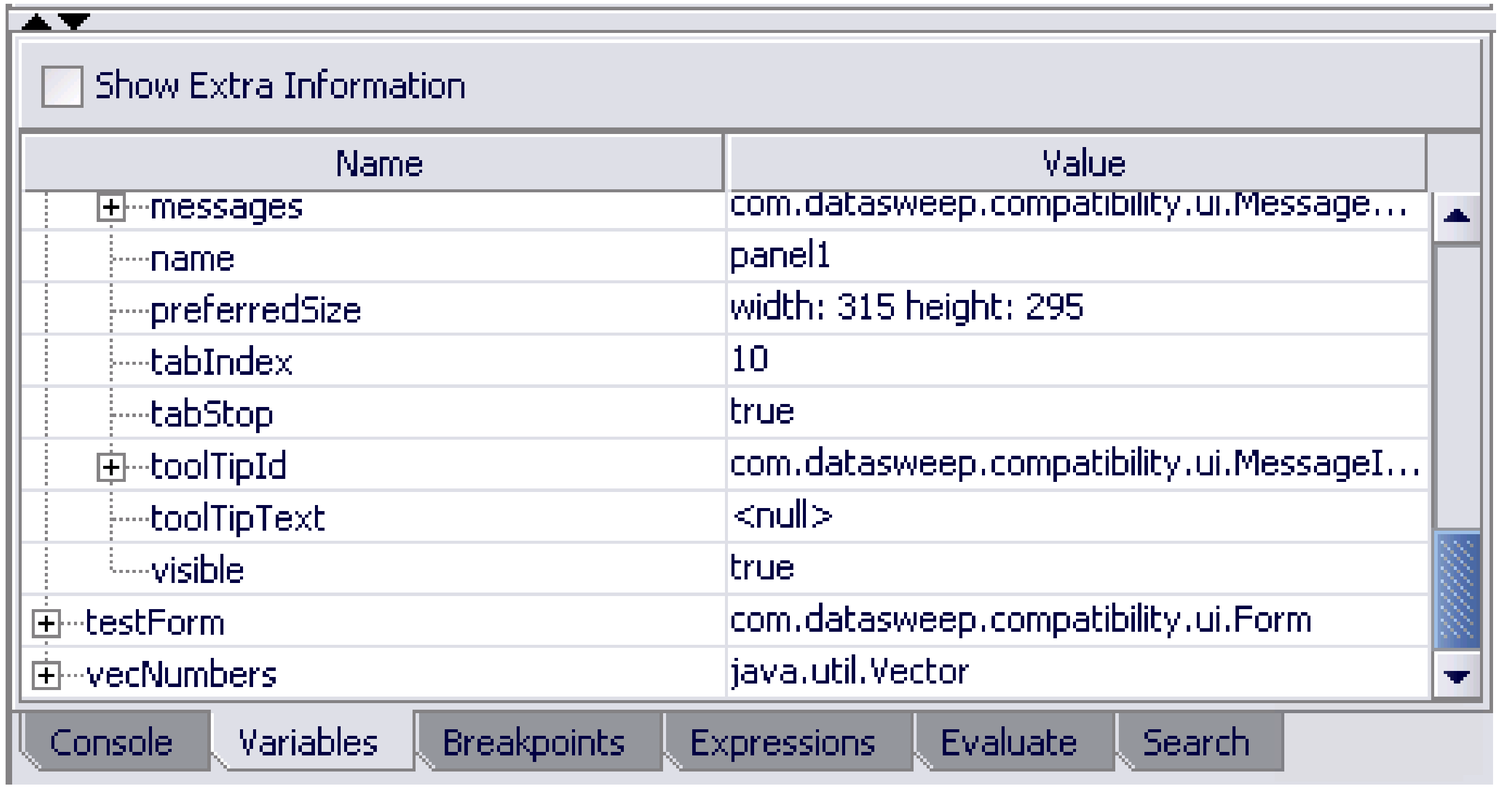
  可以通过print(...)或println(...)函数输出内容到该控制台



FTPC编程调试

- Process Designer**变量列表**

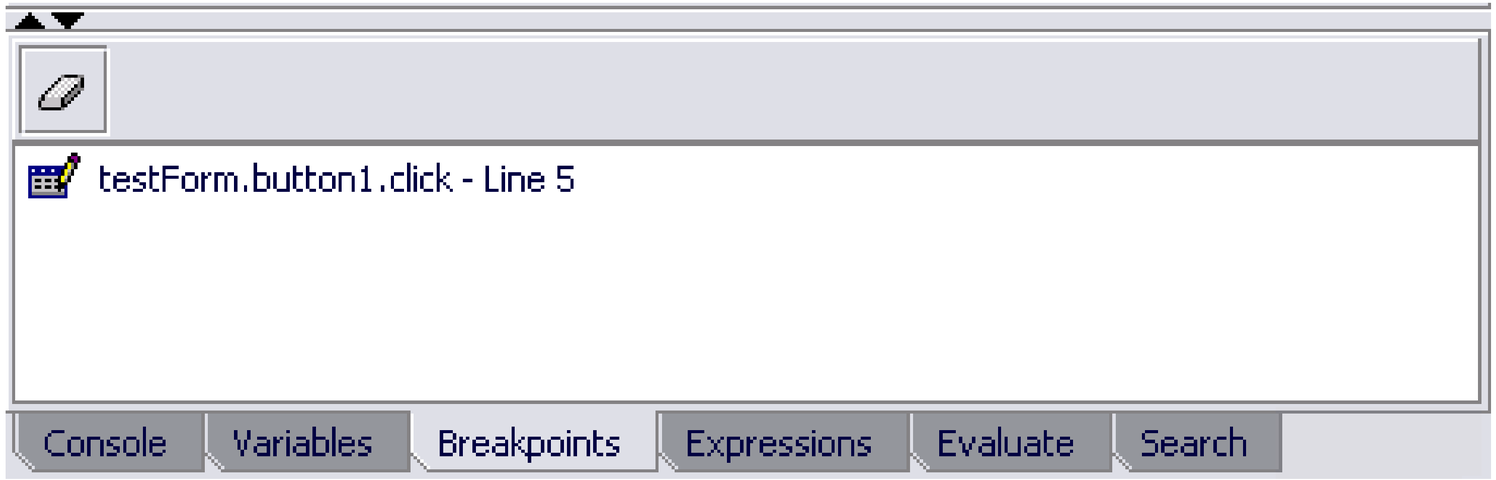
  在单步执行或执行到断点位置时，可以通过变量列表查看当前程序的变量



FTPC编程调试

- Process Designer**断点清单**

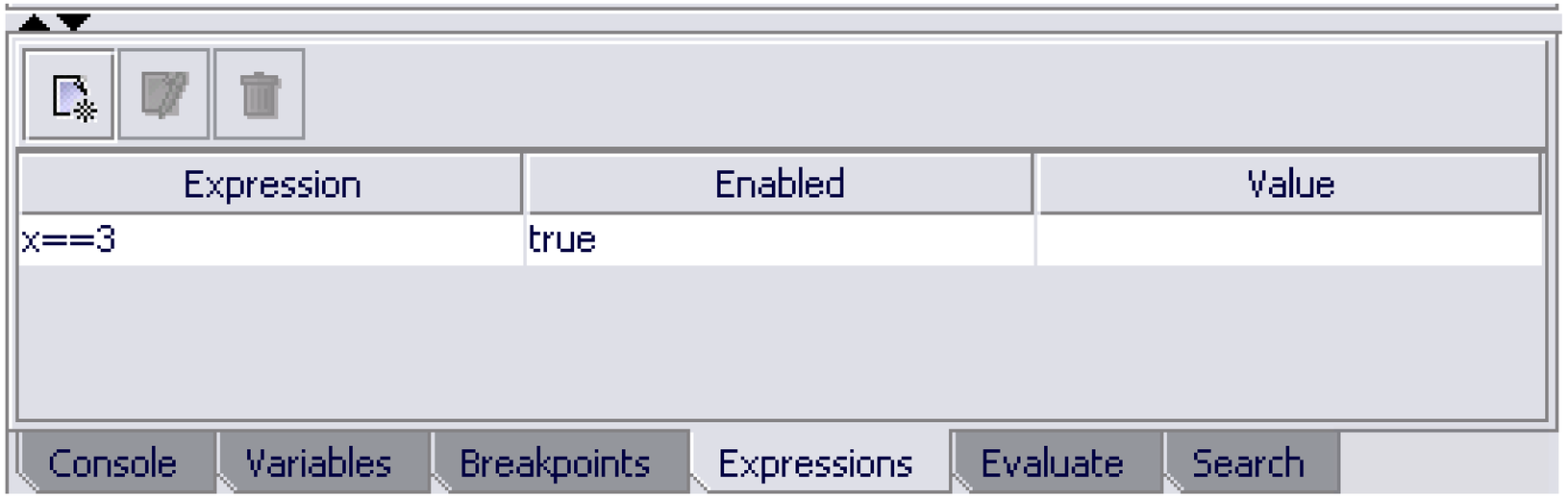
  列出当前窗体设置的所有断点位置



FTPC编程调试

- Process Designer**表达式清单（1）**

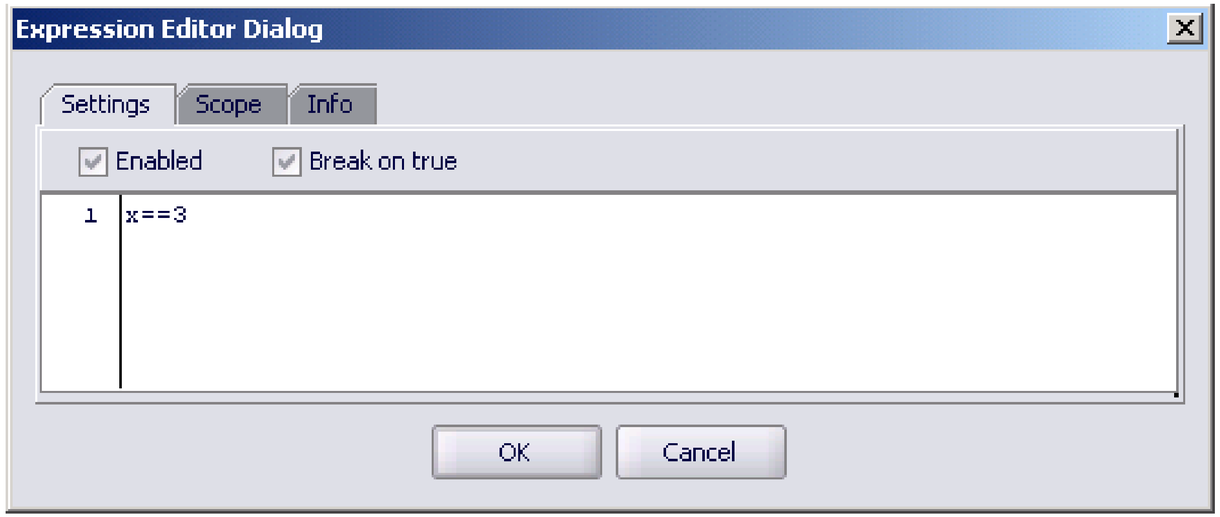
  测试模式时，显示用户输入的表达式值

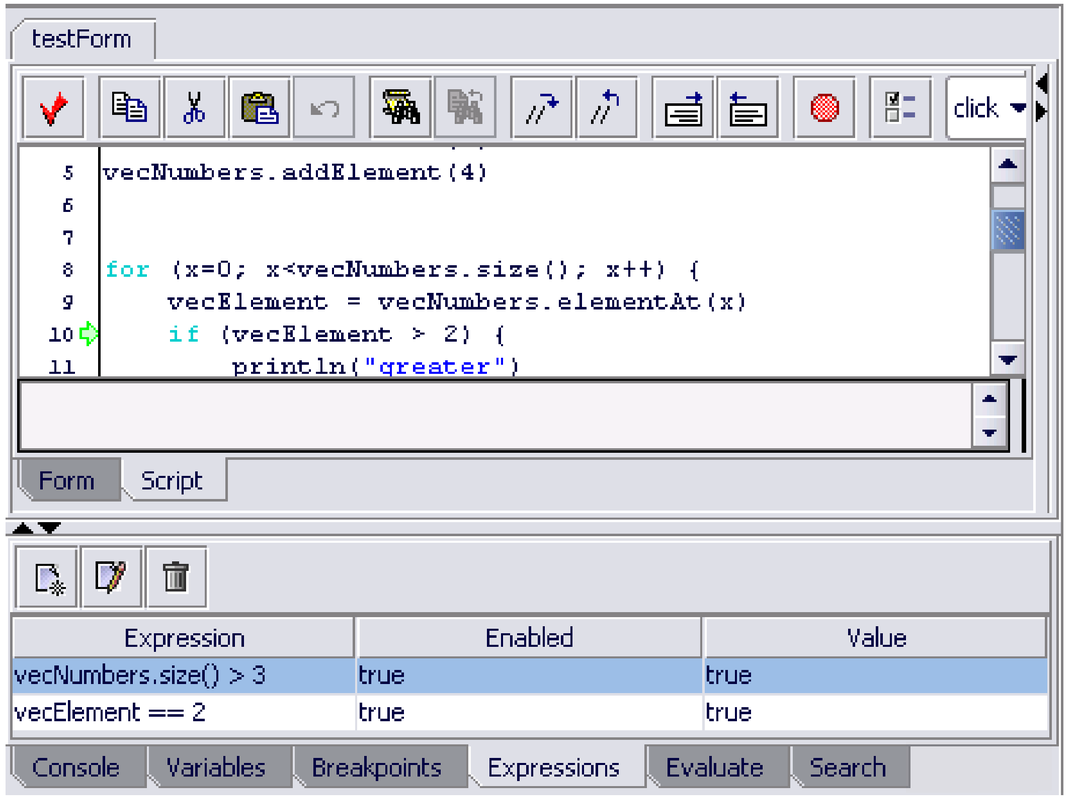


FTPC编程调试

- Process Designer**表达式清单（2）**

  创建表达式清单，设置范围

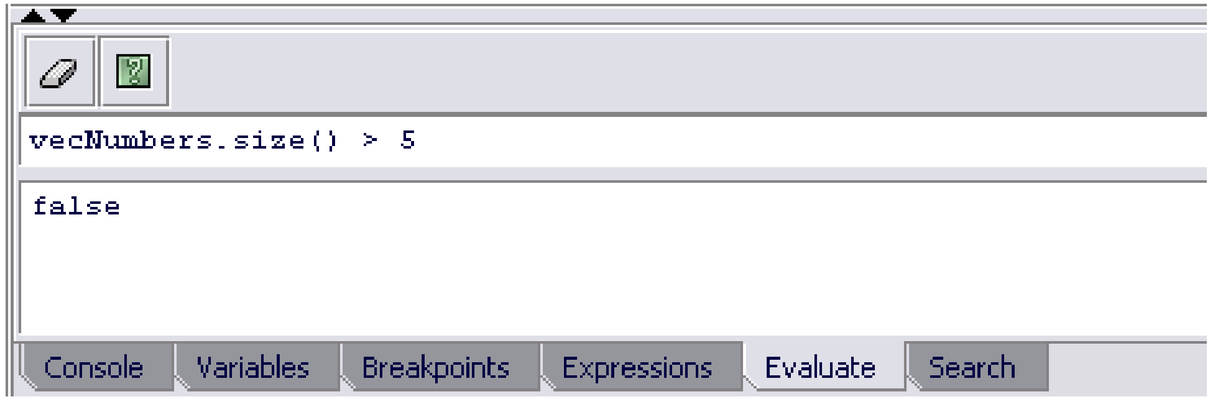




FTPC编程调试

- Process Designer**评估**

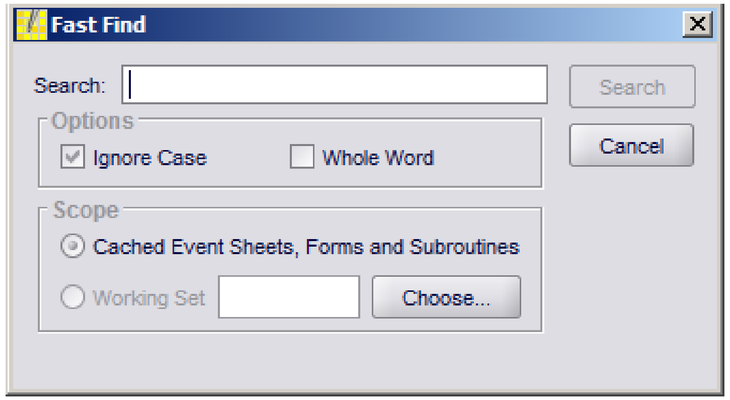
  创建表达式清单，评估真假或输出值

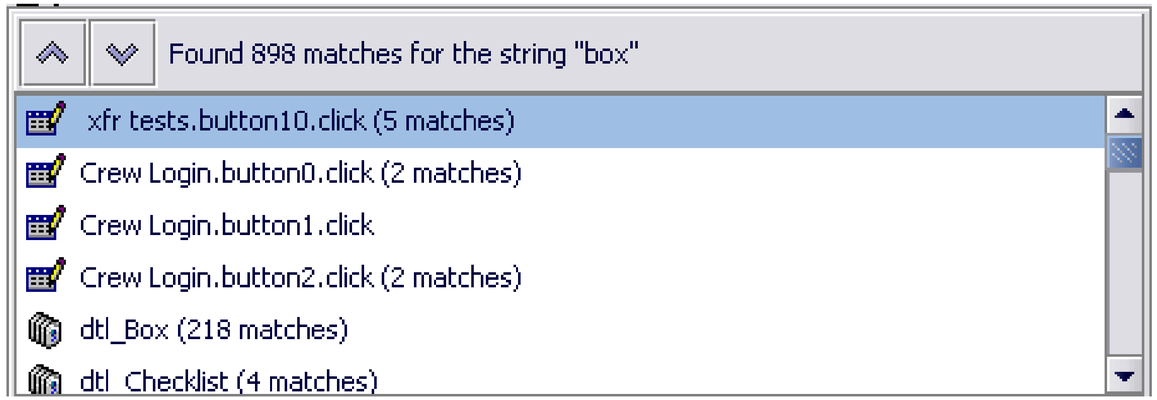


FTPC编程调试

- Process Designer**搜索**

在缓存中搜索匹配关键字，显示在搜索结果列表中，用户可以上下遍历相关内容





FTPC编程调试

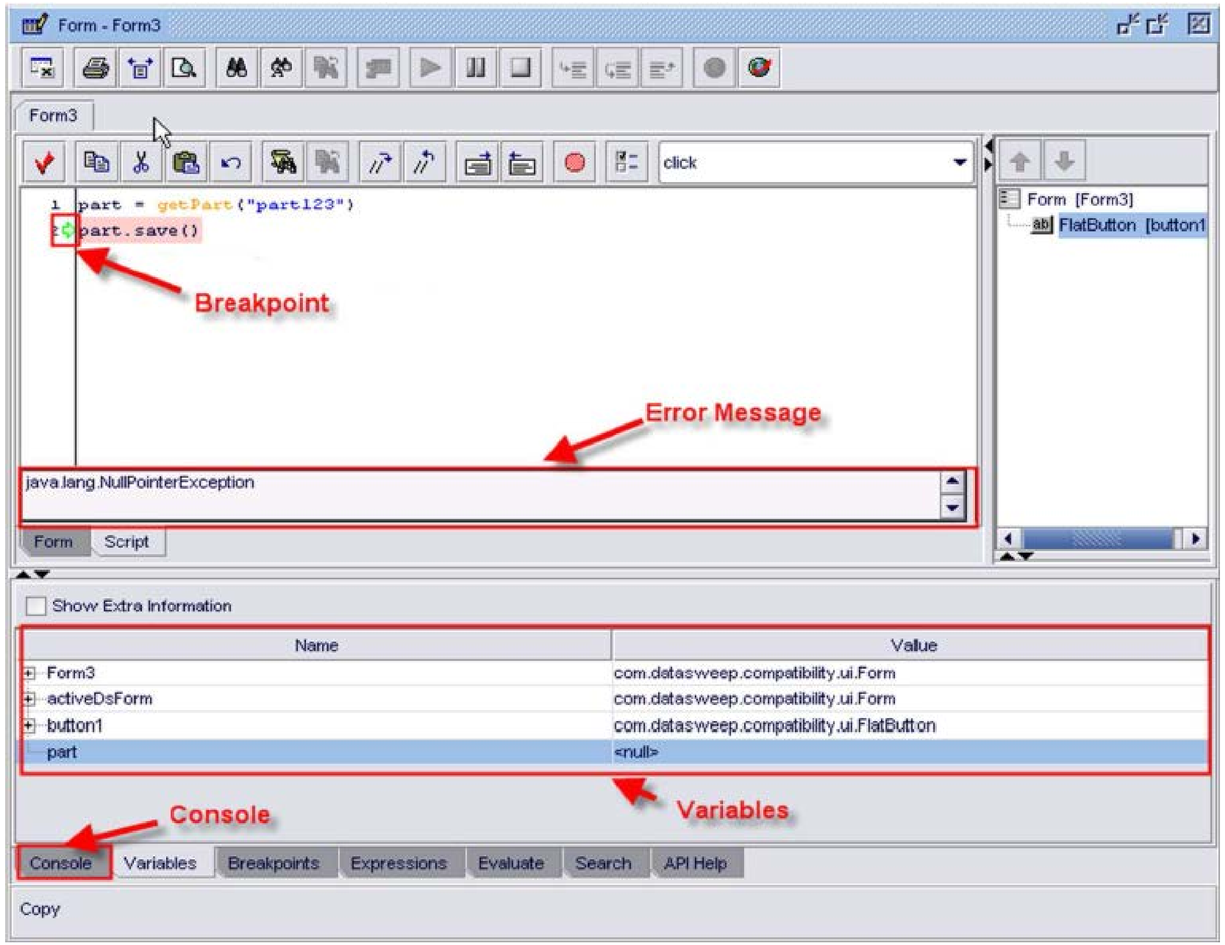
- **在代码里获得最近一次的代码调用返回，如果有错误则显示对话框**

response = getLastResponse()

checkAndDisplayResponse(response)

FTPC编程调试

- Process Designer**调试样例**



FTPC编程调试

- **Demo**

内容

Rockwell Automation

- FTPC编程环境
- FTPC编程语言
- FTPC界面编程
- FTPC编程调试
- QA

QA

Rockwell Automation

TechED™

Inspire. Educate. Innovate.

LISTEN. THINK. SOLVE.®

谢谢

PUBLIC INFORMATION

敬请关注罗克韦尔自动化微信平台

www.rockwellautomationteched.com

Allen-Bradley • Rockwell Software

Rockwell Automation