DigitalStudio

Script Language Cheat Sheet

Overview

业务脚本（Business Script）为业务逻辑片段，用于描述无法结构化描述的业务逻辑。业务脚本语言作为对元数据配置化开发方式的补充，面向元数据进行编程，帮助用户进行灵活的逻辑扩展。通常在定制过程中需要扩展逻辑的情况下使用，例如：服务实现、数据转换、参数校验等。

Keywords

KeyWord	Description	Example
as	用于脚本中类型转换	`def x = 3 as String` //表示将x转换为String类型
assert	断言，用于验证假设的条件是否为真	`assert 1==2` //在断言中该表达式不成立

break	用于在循环中跳出循环	`for(int i = 0;i<5;i++)`
		`{`
		`if(i==2)`
		`continue;`
		`if(i==3)`
		`break;`
		`}`

builder	数据转换专用关键字，用于构建一个entity实例。
	分为构建当前组件的entity和跨组件的entity。
		1. 构建当前组件的entity实例：
		`builder.UserEntity{};`
		//UserEntity为entity名称
		2. 构建跨组件的entity实例：
		`builder.user(entity:'com.huawei.soa.bdf.UserEntity'){};`
		//user为变量名称

continue	用于跳出当前这一轮循环继续下一轮	`for(int i = 0;i<5;i++)`
		`{`
		`if(i==2)`
		`continue;`
		`if(i==3)`
		`break;`
		`}`

Constant	用于引用常量	`Constant.RERSON_MAX_SIZE` //RERSON_MAX_SIZE为常量名称
def	定义一个弱类型变量	`def x = 3;` `x = "abc";`

default	表示case中条件都不满足时进入的默认分支	`int i =3 ;`
		`switch(i)`
		`{`
		`case 3:`
		`println 3;`
		`break;`
		
		`default:`
		`println "default";`
		`break;`
		`}`

Enums	用于引用枚举值	`Enums.GENDER.male.value` //GENDER为枚举名称，male为枚举项名称，value为枚举项的值。

Keywords

KeyWord	Description	Example
ErrorCode	用于引用错误码	`ErrorCode.CONVERT_DATEFORMATE("2015-01-01","yyyy-mm-dd","yyyymmdd")` //CONVERT_DATEFORMATE为异常名称，"2015-01-01","yyyy-mm-dd","yyyymmdd"为参数列表。

filter	数据转换专用关键字，根据条件过滤入参	`map req.addresses,`
		`as:'addr'filter{addr.id != "44"}{`
		`addresses{`
		`id(addr.id)`
		`address(addr.address)`
		`}`
		`}`
		//示例中filter将会过滤addr中id等于44的元素。

true/false	表示boolean 类型的常量	`Boolean mustIterate = true;` `Boolean isNotTrue = false;`

finally	进行异常处理过程后一定会执行的代码	`try {`
		`// Your code here`
		`} catch (Exception e) {`
		`// List Exception handling code`
		`} finally {`
		`// will execute with or without`
		`}`

for	基本循环语句	`for(int i = 0;i<5;i++)`

in	判断变量是否在一个集合中存在	`def list = [1,2,3];`
		`if(1 in list)`
		`//do something`

I18n	用于引用国际化	`I18n."L10N.OperationSuccess"("submit")`

if/else	表示条件分支	`int x = func()`
		`if(x==3){`
		`//do something`
		`}`
		`else`
		`{`
		`//do something`
		`}`

log	用于引用日志	`log.error "this is a error log"`

map	数据转换专用关键字，用于转换list类型的字段	`map (req.addresses,`
		`as:'addr',false){`
		`addresses{`
		`id(addr.id)`
		`address(addr.address)`
		`}`
		`}` 或
		`map req.addresses,`
		`as:'addr',false,{`
		`addresses{`
		`id(addr.id)`
		`address(addr.address)`
		`}`
		`}`
		//将入参req中的addresses属性取出并取别名为addr，addr一般为list类型。
		false表示遍历列表时，不将列表打平处理，保留之前的结构。默认是true。

Keywords

KeyWord	Description	Example
null	同java,表示空值	def x = **null** //表示定义变量x为空值
return	同java，表示返回值	**return** true //表示返回值为true

switch/case	基于不同的条件执行不同的动作	int i =3 ;
		switch(i){
		case 3:
		println 3;
		break;
		
		default:
		println "default";
		break;
		}

throw	抛出异常	**throw** ErrorCode.NULL_EXCEPTION //NULL_EXCEPTION为定义好的异常元数据

try/catch	可能抛出异常并需要异常处理的代码放到try语句块中执行/捕获try中可能抛出的异常	**try**
		{
		// Your code here
		}
		catch(Exception e){
		// List Exception handling code
		// here
		}

_this	数据转换专用关键字，用于获取当前正在构建的实例对象。	builder.UserEntity
		{
		def user = _this
		}
		//user等价于UserEntity

$BC	用于访问本bc或跨bc的entity，bo，context	**$BC**("com.huawei.bes.common").UserEntity.make(); $BC("com.huawei.bes.om").UserBo.func(param); $BC("com.huawei.bes.pc").$UserContext.create(map);
$Binding Context	从业务上下文的绑定上下文中根据变量名称获取变量值。	def a = **$BindingContext**.key;

$BP	流程操作前缀。具体使用时需要指定流程全名，指定方式为("xxx.xxx.xxx")。	//查找流程定义
		bpDefinition = **$BP**("com.huawei.soa.bdf.TestBP").find();

$Factor	用于引用因子	def a= **$Factor**_subscriber.id; //$Factor_为引用因子的固定前缀，subscriber为因子名称，id为因子属性名称。
$Request Context	从业务上下文的请求上下文中根据变量名称获取变量值。	def a = **$RequestContext**.key;

Data Type（元数据支持的数据类型参见字段类型说明）

Data Type	Description	Example
byte	字节型 -128～127	**byte** a = 1
short	短整型 -32768～32767	**short** a = 10
int	整型 -2147483648～2147483647	**int** a = 1234
long	长整型 -9223372036854775808 ~ 9223372036854775807	**long** a = 100000030
float	32位浮点型 -3.4E38～3.4E38	**float** a = 12.34
double	64位浮点型 -1.7E308～1.7E308	**double** a = 12.3456565
java.lang.Byte	字节型 -128～127	**Byte** a = 1 或者 def a = 1 as **Byte**
java.lang.Short	短整型 -32768～32767	**Short** a = 10 或者 def a = 10 as **Short**
java.lang.Integer	整型 -2147483648～2147483647	**Integer** a = 1234 或者 def a = 1234 as **Integer**
java.lang.Long	长整型 -9223372036854775808 ~ 9223372036854775807	**Long** a = 1000000 或者 def a = 1000000 as **Long**
java.lang.Float	32位浮点型 -3.4E38～3.4E38	**Float** a = 12.32 或者 def a = 12.32 as **Float**
java.lang.Double	64位浮点型 -1.7E308～1.7E308	**Double** a = 12.332211 或者 def a = 12.332211 as **Double**
java.math.BigDecimal	不可变的任意精度的有符号十进制数	**BigDecimal** a = 12.32g 或者 def a = 12.32g as **BigDecimal**
boolean	布尔类型	**boolean** a = true
Boolean	布尔类型	**Boolean** a = true 或者 def a = true as **Boolean**
char	字符类型	**char** a = 'c'
String	字符串类型	**String** a = "aaaaa"
List	列表类型	def a = [] 或者 def a = ["aa", "bb", "cc"]
Map	映射类型	def a = [:] 或者 def a = ["key1":"value1", "key2":"value2"]
Set	集合类型	Set s1 = ["a", 2] 或者 def s2 = [2, 'a'] as Set
entity	元数据entity类型 **注意：** 该数据类型仅限在数据转换脚本中使用，用于表示跨组件访问元数据。	**builder.user(entity:com.huawei.soa.bdf.UserEntity)**

Metadata Operations

Metadata	Syntaxes	Example
Entity	1. Entity.**operation**(Map param) 2. EntityInstance.field 3. EntityInstance.**operation**(Map param) 4. $BC("包名").entity.**operation**(Map param)	NameEntity.**make**([firstname:"Lucy",lastname:"Green"]) def instance = $BC("com.huawei.bes.sm.base.operlog").OperLog instance.field def instance = $BC("com.huawei.bes.sm.base.operlog").OperLog instance.**make**(params) $BC("com.huawei.soa.bdf").NameEntity.make([firstname:"Lucy",l astname:"Green"]);
	findByKey: 根据主键的增删改查	def id = 1; def user = UserEntity.**findByKey**(id);
	findOne: 专家模式查询	def user = UserEntity.**findOne**([id:id]);
	findOneBy${criteria}: 根据criteria查询单条	def userByVc = UserEntity.**findOneBy**QueryById([vId:id]);
	findBy${criteria}: 根据criteria查询Entity列表	def userListByVc = UserEntity.**findBy**QueryByUserType([vUserType:userType]);
	findCountBy${criteria}: 根据criteria查询Entity列表数量	def count = UserEntity.**findCountBy**QueryByUserType([vUserType:userType]);
	make: 构造实例	def name = NameEntity.**make**([firstname:"Lucy",lastname:"Green"]);

	assemble: 组装Entity实例	def name = NameEntity.make([firstname:
		"Tom",lastname:"Green"]);
		def address = AddressEntity.make([address:
		"nanjing",addresstype:"home"]);
		def userView = UserViewEntity.**assemble**([name,address]);

	reAssemble: View Entity类型的实例组装（实例方法）	def newName = NameEntity.make([firstname: "Mike",lastname:"Green"]); userView.**reAssemble**([newName]);
	disassemble: View Entity类型的实例拆解（实例方法）	def name = userView.**disassemble**("com.huawei.soa.bdf.NameEntity");
Business Operation	String BOName.**operation**(Map param)	def user = UserOperation.queryByAgeAndEmail(20,"vip@sina.com");
Business Service	callBusinessService(String servicename,String operation,Map param)	def user = **callBusinessService**("com.huawei.bes.omnichannel.component.Use rEntityService","queryByAgeAndEmail",[20,"vip.sina.com"]);

Business Context	1. $contextName.**create**(map) //创建业务上下文
	2. $contextName.**destroy**(map) //销毁业务上下文
	3. $contextName.**get**(map) //获取业务上下文
	4. $contextName.**exist**(map) //检查业务上下文是否存在
	5. $contextName.**findOrCreate**(map) //查找并创建业务上下文
	
	def a=$contextName.create(map)
	a.bind() //绑定上下文
	
	1. $BindingContext("contextName") //获取绑定上下文
	2. $BindingContext("contextName").item //获取绑定上下文的item
	3. $BindingContext("contextKey")childContext.id //获取绑定上下文中嵌套的item
		$BC("com.huawei.bcName").$contextName.**findOrCreate**(map)

	1. $RequestContext //获取请求上下文 2. $RequestContext.id //获取请求上下文的item	
	$BC("BCName").$contextName.findOrCreate(map) //跨BC调用上下文	

Metadata Operations

Metadata	Syntaxes	Example
Business Event	1. createEvent(String eventName) 2. BEInstance.**operation**(param) //操作事件的方法包括： **setBody**：设置事件的主体 **setCreator**：设置事件的创建者 **publish**：发布事件	def orderEvent = createEvent("com.huawei.bes.om.OrderNotifyEvent"); def entity = ['name':'sqx','id' : 'sss'] orderEvent.**setBody**(entity) orderEvent.**setCreator**("yyy") orderEvent.**publish**();
Business Process	$BP("BPName").**operation**(param)	_
	Find：流程查找	def bp = $BP("com.huawei.soa.bes.common. Fulfillment").**find**()
	define/defineDynamic/defineDynamicNonPersist：流程编排	def map = ["flowVar1" : 1234] def creatorId = "z00123456" // "com.huawei.soa.bes.FulFillmentOrder"为流程名称 def newBpd1Name = $BP("com.huawei.soa.bes.FulFillmentOrder").**define**(map, creatorId) println newBpd1Name

	execute/executeAsyn：流程执行	// script 流程执行
		def map = ["flowVar1" : 1234]
		def creatorId = "z00123456"
		// 执行静态流程，通过流程名同步调用execute方法|异步方式调用
		executeAsyn方法
		def context =
		$BP("com.huawei.soa.bes.FulFillmentOrder").**execute**(map,
		creatorId)
		println context.processInstance
		def contextAsyn =
		$BP("com.huawei.soa.bes.FulFillmentOrder").**executeAsyn(map,**
		creatorId)
		println contextAsyn.processInstance

	continueSync/continueAsyn：继续流程实例执行	def map = ["flowVar1" : 1234] $BP("PI_2345").**continueSync**(map) $BP("PI_2345").**continueAsyn**(map)
	suspend/suspendAsyn：流程挂起	$BP("PI_2345").**suspend**(); $BP("PI_2345").**suspendAsyn**();
	resume/resumeAsyn：流程恢复	$BP("PI_2345").**resume**(); $BP("PI_2345").**resumeAsyn**();
	retry/retryAsyn：流程重试	def params = ["testVar":"001"] $BP("PI_2345").**retry**(params); $BP("PI_2345").**retryAsyn**(params);
	abandon/abandonAsyn：丢弃流程实例	$BP("PI_2345").**abandon**(); $BP("PI_2345").**abandonAsyn**();
	terminate/terminateAsyn：终止流程的执行	$BP("PI_2345").**terminate**(); $BP("PI_2345").**terminateAsyn**();

	continueActivity/continueActivityAsyn：继续执行流程节点
	reexecute/reexecuteAsyn：节点执行
	Skip：跳过节点执行
	unSkip：恢复节点跳过
		def map = ["flowVar1" : 1234]
		def activityName = "testActivityName"
		// continueActivity
		$BP("PI_2345").**continueActivity**(map, activityName)
		$BP("PI_2345").**continueActivityAsyn**(map, activityName)
		// reexecute
		def activities = [activityName]
		$BP("PI_2345").**reexecute**(map, activities)
		$BP("PI_2345").**reexecuteAsyn**(map, activities)
		// skip
		$BP("PI_2345").**skip**(activityName)
		// unSkip
		$BP("PI_2345").**unSkip**(activityName)

System Functions

Category	Function	Description	Example
文本	trim(string)	去除首尾空格	trim(' Sample ') // 返回值为'Sample'
	length(text)	获取字符串的长度	result = "bp_this is a test string.txt" length_char = length（result）; // 取字符串的长度，返回结果为28
	left(text,num_chars)	返回指定位置左侧的子串	prefix_char = left(result, 2); // 取前2个字符，作为前缀，返回结果为“bp”
	right(text,num_chars)	返回指定位置右侧的子串	suffix_char = right(result, 3); // 取后3个字符，作为后缀，返回结果为“txt”
	mid(text, start_num, num_chars)	返回指定位置内的子串	mid_char = mid(result, 3, 21); // 取中间21个字符，从第3个字符开始，返回结果为“this is a test string”
	upper(text)	将字符串大写处理	upper('Sample') // 返回值为SAMPLE
	lower(text)	将字符串小写处理	lower('Sample') // 返回值为sample
	match(text, pattern)	正则表达式匹配	-
数值	abs(number)	取绝对值	abs(-11) // 返回值为11
	max(number1 [,number2]...)	取最大值	max(6,8,3) // 返回值为8
	min(number1 [,number2]...)	取最小值	min(6,8,3) // 返回值为3
	round(number, num_digits)	四舍五入求值 number为待四舍五入数，number_digits为位数	round(4.183,2) // 返回值为4.18

	ceiling(number,significance)	向上舍入
		number为待舍入的数值，significance为基数，number和significance符号相反，则返回错误值#NUM!
			ceiling(2.5,1) //返回值为3
			ceiling(-2.5,-2) //返回值为-4
			ceiling(-2.5,2) //返回值为#NUM!

	floor(number)	向下取整	floor(9.999999) //返回值为9.0
	sqrt(number)	求平方根	sqrt(4) //返回值为16
时间	now()	获取当前日期时间，精度是年月日时分秒 返回值类型：java.sql.Timestamp	currntTime = now()
	today()	获取当前日期，精度是年月日 返回值类型：java.sql.Date	currntDate = today()
	time()	获取当前时间，精度是时分秒 返回值类型：java.sql.Time	currentTime = time()
	date(year,month,day)	创建自定义日期，精度为年月日 • 参数类型：均为int • 返回值类型：java.sql.Date	myDate = date(2016, 12,12)
	date(text)	创建自定义日期，精度为年月日 • 参数类型：String，且仅支持传入的字符串类型格式为“yyyy-[m]m-[d]d” • 返回值类型：java.sql.Date	myDate = date("2016-12-12")
	datetime(year,month,day,hours,minutes,second,milliseconds)	创建自定义日期时间，精度为年月日时分秒 • 参数类型：均为int • 返回值类型：java.sql.Timestamp	myTime = datetime(2016, 12, 12, 18, 25, 14, 123)
	datetime(text)	创建自定义日期时间，精度为年月日时分秒 • 参数类型：String，仅支持传入的字符串格式为“yyyy-[m]m-[d]d hh:mm:ss[.f...]” • 返回值类型：java.sql.Timestamp	myDateTime = datetime("2016-12-15 12:12:12.123")
校验	eval_validator(field, validation_rule_name)	对指定field应用validation_rule_name指定的校验规则	-
逻辑	and(expression,expression)	等价于运算符“&&”	and(Price<1,Quantity<1) //等价于Price<1 && Quantity<1
	isBlank(obj)	对象不能null的系统函数，支持文本	
	isNumber(text)	确定文本值是否为数字，如果是则返回true，反之返回false	-
	not(conditionExpression)	等价于运算符“！”	if(not(Status="Closed"), 1, 0)
	or(conditionExpression)	等价于运算符“||”	-

Built-in Functions

Precedence	Operators	Example

callBusinessService(String servicename, String operation name,Map param)	服务调用	//构造NameEntity的实例
		def name =
		NameEntity.make(["firstName":"Tom","lastName":"Green"]);
		
		//调用com.huwei.ide.test.component.UserEntityBOService的queryByNameEntity方法，传参为entity类型，出参为entity类型
		def user =
		callBusinessService("com.huwei.ide.test.component.UserEntityBOService","queryByNameEntity",name);

createEvent(String eventName)	事件发布	//定义事件及内容
		def orderEvent =
		createEvent("com.huawei.bes.om.OrderNotifyEvent");
		def entity = ['name':'sqx','id' : 'sss']
		
		orderEvent.setBody(entity) //设置orderEvent事件的主体
		orderEvent.setCreator("yyy") //设置orderEvent事件的创建者
		orderEvent.publish(); //发布orderEvent事件

runScript(String name,Map params)	调用其他脚本	//定义上下文
		def context = ['name':'sqx','id' : 'sss'];
		
		//调用名称为scriptName的脚本
		def result = runScript("scriptName",context);

Object transformation(String transfrormationName, Map transformParams)	执行数据转换	1. 直接访问Facade接口 ScriptFacade facade = FacadeContextFactory.getFacadeContext().getScriptFacade(); UserEntity value = (UserEntity)facade.transformation("user_transform", param); 2. 脚本中直接执行数据转换 transformation("user_transform", param)

Built-in Variables

Variable	Description	Type
trigger	仅在trigger的脚本中使用	1. BOTrigger类型：com.huawei.soa.bdf.component.business.operation.trigger.source.TriggerSource 2. Bstrigger类型：com.huawei.soa.bdf.service.trigger.TriggerSource 3. Event Trigger类型：com.huawei.soa.bdf.event.trigger.BeTriggerContext 4. Entity Trigger类型：com.huawei.soa.bdf.data.trigger.EntityTriggerContext
_header	仅在BStrigger的脚本中使用，用于获取BS消息头（包括服务调用的源地址目标地址、服务名、方法名、协议类型、服务分组、服务版本号等）	Map
currentEvent	仅在eventtrigger中使用，用于获取当前trigger的event实例	com.huawei.soa.bdf.event.service.impl.BusinessEventImpl
ec	仅在action中使用，用于在脚本中获取执行上下文	com.huawei.soa.bdf.context.ExecutionContext
targetEntity	仅在entity的trigger中使用，用于获取trigger的入参（目标Entity）	当前entity

Operators

Precedence	Operators	Description
1	?. !	安全访问避免空指针异常，逻辑非
2	**	乘方
3	++ -- + -（一元运算符）	递增/减，一元加，一元减
4	* / %	乘法，除法，取余
5	+ -	加法，减法
6	< <= > >= in instanceof	小于，小于等于，大于，大于等于，成员，实例
7	== != <=>	等于，不等于，比较
8	&&	逻辑和
9	||	逻辑或
10	= *= /= %= += -=	各种赋值运算符