DevOps

Wednesday, August 23, 2023 10:23 AM

DEV+ OPS

TDT -Test Driven Development

1.Groovy - Java
2.Python

Operations -

- Linux - shell scripting
- Git + GitHub
- CI/CD - pipelines - Jenkins
- Docker
- Kubernetes
- IAC - Infrastructure as Code [Terraform + Ansible]

Monitoring Tools -

Splunk

Deployment -
AWS

Collaboration of Development Team and Operations Team

Groovy
Requirements -
JDK
Eclipse
JVM - interprets bytecode to native code executes the class file

Groovy is an object oriented language which is based on Java platform. Groovy 1.0 was released in January 2, 2007 with Groovy 2.4 as the current major release. Groovy is distributed via the Apache License v 2.0.

Object oriented language
Supports functional Programming
Supports Dynamic Typing - def age =25

Waterfall Model

Requirement Analysis
System Design
Implementation
Integration and Testing
Deployment
Maintenance

JIRA - tool for planning

Demo seen

Agile

Scrum - holistic development approach
Follows a common for most of the projects. Incremental agile SD framework.

SCRUM Roles -
Product owner
Scrum Master
Scrum Team

Scrum Ceremonies
Sprint planning
Sprint review
Sprint retrospective
Backlog Refinement

Scrum artifacts
Product backlog
Sprint backlog
Burndown Chart

PRODUCT OWNER
SCRUM MASTER
SCRUM TEAM

Story points are added to mark a user story as important

Doable items at the sprint is called as sprint backlog

Picked from the product backlog.

Functional Programming -
Functional programming is a programming paradigm in which we try to bind everything in pure mathematical functions style. It is a declarative type of programming style. We can return a function, pass a function as a parameter etc.

GROOVY Day2

24-08-2023

Package - com.myapp.demo

Class file

First program -

```
packagecom.myapp.demo

println('My first Groovy Script')
```

Primitive Datatype - part of stack memory
User Defined Datatype or Reference Datatype - part of heap memory

Primitive Datatype	Reference Datatypes
Int	Integer
Char	Character
Byte	Float
Short	Boolean
Long	Long
Float	
Double	
boolean	

Sort objects :

small case

```
packagecom.myapp.demo

defproducts=[
newProduct(123,'Iphone10',434374),
newProduct(456,'Iphone11',434355),
newProduct(789,'Iphone12',434326),
]

defproduct=newProduct(123,'Iphone10',43434)

printproducts.sort{it.price}.productName

@Sortable(includes="price")

printlnproducts.sort()

        @ToString(includes=["productName","price"])

printlnproducts.sort({p1,p2->p1.price-p2.price})
```

Comparator - Compares the p1 and p2 and then return 1 if p1>p2 else return -1 and return 0 if all are same

MIN Max -

```
defproducts=[
newProduct(123,'Iphone10',434374),
```

ArrayList -

```
defproducts=['Iphone14','Oneplus11','SamsungFlip']

printlnproducts.class
```

Loop

```
for(productinproducts)
printlnproduct
```

Closure:

```
{
        Parameter ->
}
```

Ex -

```
products.forEach{product->printlnproduct}
```

Or

```
products.each{printlnit}
```

We pass closure as a parameter and then call it using the closure Keyword

```
Def myFunc(val){
val*val
}

printlnmyFunc(myFunc(2))

defnewFunc(val,closure){
closure(val*val)
}

newFunc(100,{println it})

defcb={println it}

defcb1={
val->printlnval
}

newFunc(100,cb1)
```

```
newProduct(456,'Iphone11',534355),
newProduct(789,'Iphone12',634326),
]

defres=products.findAll{product->product.price>=500000}
printlnres.max()
printlnres.min()
printlnproducts.average{
it.price
}


defstats=res.stream().mapToDouble{it.price}.summaryStatistics()
printlnstats.getMin()
printlnstats.getMax()
printlnstats.getAverage()
```

Reduction Step or reducing

DSL - Domain Specific Language

Ex -

```
Lunchorder {
          Soup:{ " Sweet Corn"},
          Starter : {"Fried chicken "},
          MainCourse: {"Biryani"},
      }
```

Closure inside function

Closure 1

Closure 2

Body 2

```
    [
        currency:{
          Currency currency->
           [
           to:{
           account->

    account.balance+=convert(currency,amount)
                                  }
                           ]
               }
    ]
```

What is a closure?
Closures are anonymous in nature and can be assigned to a variable that can be executed later.
It is a code snippet.

```
packagecom.myapp.demo

classLunchOrder{
Stringsoups
Stringstarters
StringmainCourse
Stringdesserts
defmeals=[]
staticdefcreate(Closureclosure)
{
    LunchOrderlunchOrder=newLunchOrder()
    closure.delegate=lunchOrder
    closure()
}
defsoups(soups)
{
    this.soups=soups
}
defstarters(starters)
{
    this.starters=starters
}
defmainCourse(mainCourse)
{
    this.mainCourse=mainCourse
}
defdesserts(desserts)
{
```

```
printlnLunchOrder.create{
soups"chicken soup"
starters"baby corn"
mainCourse"butter chicken"
desserts"ice cream"
text
}
```

```
    this.mainCourse=mainCourse
}
defdesserts(desserts)
{
    this.desserts=desserts
}
defgetText(){
doPrint(this)
}
    staticdoPrint(LunchOrderorder)
{
"Lunch Oder \n Soups : ${order.soups} \n 
Starter : ${order.starters} \n Main 
Course: ${order.mainCourse} \n Desserts: 
${order.desserts} \n"
}
}
```

Python