

Automatiserad testning – Labb Jenkins

Syfte

Studenten ska efter slutförd kurs kunna designa tester för att testa en produkt på olika nivåer samt lösa sammansatta problem inom automatiserad testning genom att använda olika testverktyg och ramverk. I Labb Jenkins skapar vi en regressionstestsvit med hjälp av våra redan utvecklade enhets och systemtester och automatiserar körningen av dem med hjälp av en Pipeline i Jenkins som triggas av att nya commits pushas.

Mål

Målet med övningen är att pusha redan existerande tester samt ett Pipeline-skript till ett repo på Github för att sedan köra testerna via Jenkins. Därefter inspektera resultaten från testerna i Jenkins.

Stage View



Betygskriterier G

- Förklara syftet med Continuous Testing
- Identifiera hur automatiserad testning hänger ihop med den övriga test- och utvecklingsprocessen
- Rapportera testresultat
- Tillämpa testverktyg i en Continuous Testing-kontext
- Använda versionshanteringssystem i arbete med programmering och testning
 - a. Använda versionshantering vid implementation av automatiserade testfall
 - b. Använda versionshantering för att i ett team genomföra ändringar i en gemensam kodbas

Betygskriterier VG

- Självständigt skapa automatiserade tester i relevanta verktyg
- Självständigt övervaka testning och avgöra vilket testverktyg som passar för problemet

Labs material

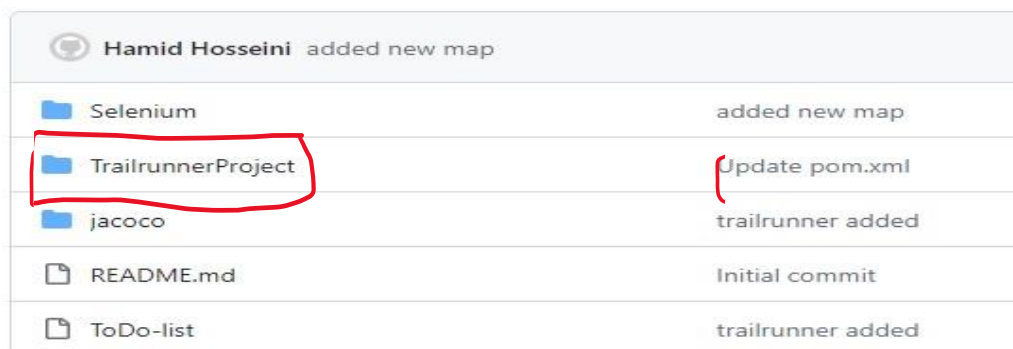
För att kunna utföra alla moment i labben, varje student måste ha tillgång till följande filer.

1. Selenium lab (infotiv car rental)
2. Trailrunner projekt (enhet testning kursen)

Uppgift

Add Selenium lab to repository

1. Kom ihåg att göra git commit och git push efter varje steg. För mer information om hur en Jenkinsfile byggs upp, se här: [Jenkins Pipeline Syntax](#).
2. Skapa en ny branch i repon "Trailrunner" och kalla den "b1".
3. Klona repon till din dator med git bash-kommandot.
4. Byt till den nyligen skapade branchen "b1".
5. Skapa en tom mapp i din lokala repo och ge den namnet "Selenium".
6. Kopiera lab selenium och lägg den i mappen "Selenium". Utför sedan en commit och pusha dina ändringar.
7. Pusha dina ändringar från din lokala repo till din branch "b1".
8. Uppdatera din main branch genom att skapa en pull request från "b1" till "main".
9. Nu bör din main branch i repon "Trailrunner" innehålla följande filer:



Hamid Hosseini added new map		
📁 Selenium		added new map
📁 TrailrunnerProject		Update pom.xml
📁 jacoco		trailrunner added
📄 README.md		Initial commit
📄 ToDo-list		trailrunner added

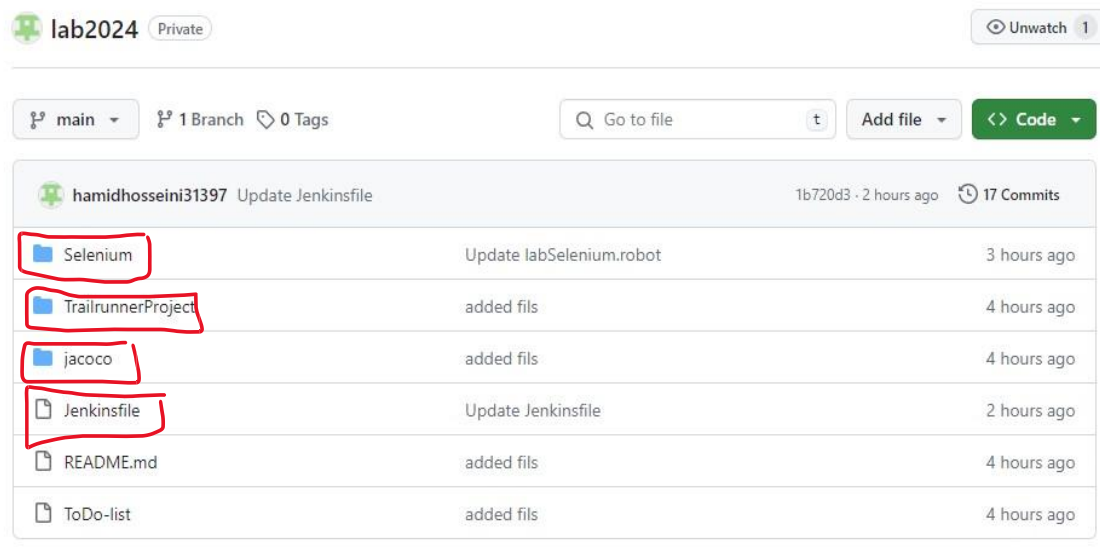
Jenkins Configuration

1. Skapa en Pipeline projekt som heter <ditt_namn> i JenkinsLab
2. Konfigurera din Pipeline att använda Jenkinsfile från ditt repository (förmodligen heter repositoryt "Trailrunner").
3. Projekt måste vara **parameterized** med två paramet (main och b1)

4. För att automatisera byggnadsprocessen bör projektet konfigureras med en **Build Triggers**. Jenkins kommer regelbundet att övervaka versionshanteringssystemet (Git) för att identifiera eventuella ändringar i koden. Om det upptäcker förändringar sedan den senaste byggnaden kommer Jenkins att initiera en ny byggnad.

Add Jenkinsfile to repository

1. Navigera till din b1-gren och lägga till en tom Jenkinsfilen direkt i den grenen.
2. Jenkinsfilen bör inte vara placerad inuti någon mapp.
3. Jenkinsfilen måste ha namnet Jenkinsfile.
4. När du är klar, utför en "push" för att överföra dina ändringar till b1-grenen.
5. Uppdatera sedan din huvudgren (main) från b1-grenen för att inkludera de senaste ändringarna.
6. Se till att din huvudgren (main) innehåller följande filer med den exakta strukturen.

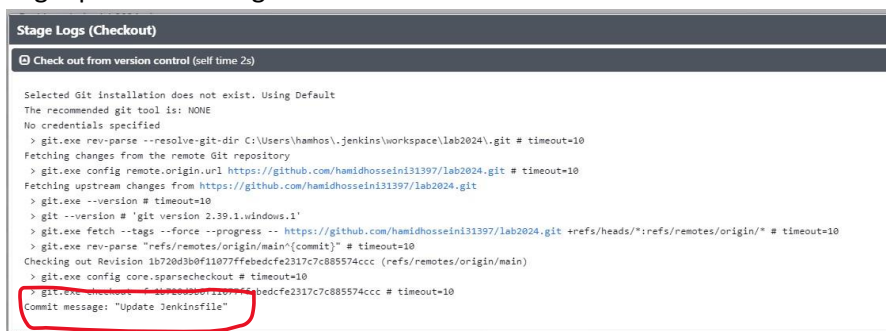


Pipeline Script

Din pipeline måste ha följande stages :

- **Checkout** (Hämtar senaste kodversionen för den valda grenen)

Kontrollera att Checkout stage har hämtat senaste kod version för att builda sedan, klicka på "Logs" på Checkout stage



- Build**(Kompilerar Trailrunner-projektet.)

Kontrollera att ditt bygge går igenom utan fel. Under sammanfattningen för bygget, klicka på "Logs":

```
Shell Script -- mvn compile (self time 2s)

[INFO] from pom.xml
[INFO] -----[ jar ]-----
[INFO]
[INFO] --- jacoco:0.8.11:prepare-agent (prepare-agent) @ trailrunner ---
[INFO] argLine set to -javaagent:C:\\Users\\hamhos\\.m2\\repository\\org\\jacoco\\org.jacoco.agent\\0.8.11\\org.jacoco.agent-0.8.11-runtime.jar=destfile=C:\\Users\\hamhos\\.jenkins\\workspace\\lab2024\\target\\jacoco.
exec
[INFO]
[INFO] --- resources:3.3.1:resources (default-resources) @ trailrunner ---
[WARNING] Using platform encoding (UTF-8 actually) to copy filtered resources, i.e. build is platform dependent!
[INFO] skip non existing resourceDirectory C:\\Users\\hamhos\\.jenkins\\workspace\\lab2024\\src\\main\\resources
[INFO]
[INFO] --- compiler:3.11.0:compile (default-compile) @ trailrunner ---
[INFO] Nothing to compile - all classes are up to date
[INFO]
[INFO] BUILD SUCCESS
[INFO]
[INFO] Total time: 0.511 s
[INFO] Finished at: 2024-02-22T11:01:30+01:00
[INFO] -----
```

- Test**(Kör alla testfall för Trailrunner-projektet.)

Kontrollera att dina test fall har körts korrekt, genom att klicka på "Logs " på Test stag.

```
[INFO] T E S T S
[INFO] -----
[INFO] Running se.iths.DatabaseTest
[INFO] Tests run: 4, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.09 s - in se.iths.DatabaseTest
[INFO] Running se.iths.RunTest
[INFO] Tests run: 5, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.024 s - in se.iths.RunTest
[INFO] Running se.iths.UserTest
[ERROR] Java HotSpot(TM) 64-Bit Server VM warning: Sharing is only supported for boot loader classes because bootstrap classpath has been appended
[INFO] Tests run: 7, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 1.165 s - in se.iths.UserTest
[INFO]
[INFO] Results:
[INFO]
[INFO] Tests run: 16, Failures: 0, Errors: 0, Skipped: 0
[INFO]
```

- Post Test**(Publicerar testresultatet)

Kontrollera att ditt test resultat genom att klicka på "Test Result " och "se.iths"

Test Result : se.iths

0 failures (±0) 16 tests (±0) Took 1.8 sec. [Add description](#)

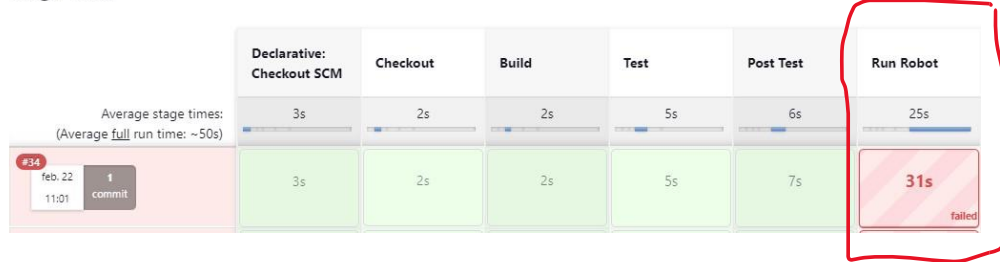
All Tests

Class	Duration	Fail	(diff)	Skip	(diff)	Pass	(diff)	Total	(diff)
DatabaseTest	73 ms	0		0		4		4	
RunTest	26 ms	0		0		5		5	
UserTest	1.7 sec	0		0		7		7	

- Run Robot and Post Test**(Kör Selenium-labbet och publicerar resultatet)

Kontrollera att dina testfall har körts. man kan se direkt på Stage view

Stage View



Om man vill se varför testen har filat man kan klicka på "Logs"

Stage Logs (Run Robot)

```
Windows Batch Script -- python -m robot C:/Users/hamhos/jenkins/workspace/lab2024/Selenium (self time 31s)

C:/Users/hamhos/jenkins/workspace/lab2024/python -m robot C:/Users/hamhos/jenkins/workspace/lab2024/Selenium
=====
Selenium
=====
Selenium.LabSelenium
=====
[ WARN ] The chromedriver version (120.0.6099.109) detected in PATH at C:/Users/hamhos/Desktop/webdriver/chromedriver.exe might not be compatible with the detected chrome version (121.0.6167.185); currently, chromedriver 121.0.6167.184 is recommended for chrome 121.*, so it is advised to delete the driver in PATH and retry
[ FAIL ]
Button with locator '//tbody/tr[1]/td[5]/form[1]/input[4]' not found.

negativDateTest1
[ WARN ] The chromedriver version (120.0.6099.109) detected in PATH at C:/Users/hamhos/Desktop/webdriver/chromedriver.exe might not be compatible with the detected chrome version (121.0.6167.185); currently, chromedriver 121.0.6167.184 is recommended for chrome 121.*, so it is advised to delete the driver in PATH and retry
[ PASS ]

negativDateTest2
[ WARN ] The chromedriver version (120.0.6099.109) detected in PATH at C:/Users/hamhos/Desktop/webdriver/chromedriver.exe might not be compatible with the detected chrome version (121.0.6167.185); currently, chromedriver 121.0.6167.184 is recommended for chrome 121.*, so it is advised to delete the driver in PATH and retry
[ PASS ]

Error signal -- Robot Framework tests failed (self time 15ms)

Configure robot framework report collection (self time 40ms)
```

För att kontrollera testresultat för Robotframework man klicka på "Robot Results"

Robot Framework Test Results

Executed: 20240222 11:01:44.533
Duration: 0:00:18.212 (-0:00:00.017)
Status: 0 critical test, 0 passed, 0 failed, 0 skipped
5 test total (±0), 4 passed, 1 failed, 0 skipped
Results: [report.html](#)
[log.html](#)
[Original result files](#)

Även ni kan se testresultatet här:



För varje stage som du skrivit kod för i ditt pipelineskript:

- Utför en "push" för att överföra de specifika ändringarna till din b1-gren på Gitrepositoriet.

För varje stage som du har pushat till b1:

- Öppna en "pull request" (förfrågan om att integrera ändringar) från b1 till huvudgrenen (main) när den stage är helt färdigt och är redo att hamna i main branch.

För varje stage måste de inkluderas i huvudgrenen (main) genom pull request:

- Efter att pull request har granskats och godkänts, utför en "merge" för att inkludera ändringarna från b1 till main-grenen.

Frågor

Svara på följande frågor (max 1 sida)

- Förklara syftet med Continuous Testing och hur det används under labben.
- Hur kopplar automatiserad testning ihop med den bredare test- och utvecklingsprocessen, och kan du ge ett exempel från labben som illustrerar detta?
- Vilken del i din Pipeline tycker du ger störst vinst i att köra via Jenkins?
- Vad hade du velat lära dig mer eller förstå bättre kopplat till Jenkins?
- Vad var svårast med labben?

Uppgift för Väl Godkänt (VG)

Utred ytterligare testverktyg som kan utnyttjas av Jenkins och utöka din Pipeline med ett (1) verktyg. Detta skulle kunna vara ett verktyg för linting (kodgranskning), kodtäckning av unittest eller ett verktyg för att visa teststatistik.

För att få betyget VG, skicka in en (1) av följande:

- En log som visar resultatet av ett bygge där ditt verktyg körs och producerar någon form av resultat. Eller:
- Om du inte lyckas köra verktyget, skicka in en textfil där du beskriver varför och när ditt föreslagna verktyg skulle kunna vara användbart i en Pipeline.

Inlämning & Deadline

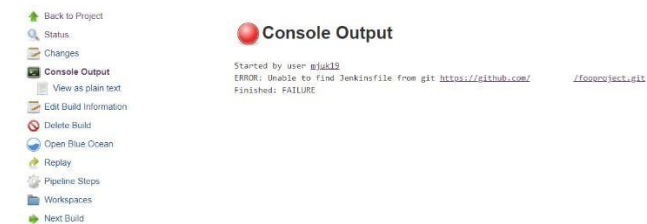
- Inlämning sker via ITHS-distans. Deadline: 2024-03-20 kl. 23:55. Återkoppling sker senast 10e april.
- Muntlig presentation: 2024-03-22

Vad ska finnas med?

Lämna in följande filer för labben:

- Jenkinsfile
- Console log av "Successful Build"
- Console log av "Post Test"
- Robot Framework log från "Run Robot"
- Textfil med svar på frågor
- Logg eller text enligt uppgift för VG enligt ovan.

Felsökning



I bilden nedan så är Jenkinsfile inte korrekt:



Om du vill se "Open-report.html" och "Open-log.html" men den kraschar och öppnar inte länkar gör följande steg

- Click on Manage Jenkins from left side panel.
- Click on Script Console
- Copy this into the field

```
System.setProperty("hudson.model.DirectoryBrowserSupport.CSP","sandbox allow-scripts; default-src 'none'; img-src 'self' data: ; style-src 'self' 'unsafe-inline' data: ; script-src 'self' 'unsafe-inline' 'unsafe-eval' ;")
```

- Klik on RUN

