Modern projekt eszközök

Pataki Norbert

ELTE Informatikai Kar,
Programozási Nyelvek és
Fordítóprogramok Tanszék
patakino@elte.hu

Projekt eszközök

Tulajdonságok

- Platform-függetlenség
- XML-alapú szintaxis vagy DSL-ek használata
- Absztrakció: dependency
- Paradigma: nem feltétlenül imperatív
- Artifact repository-k
- Bővíthetőség
- Példák
  - ant (`http://ant.apache.org/`)
  - maven (`https://maven.apache.org/`)
  - Gradle (`http://gradle.org/`)

Maven, tulajdonságok

- Szoftver projekt menedzsment eszköz
- Projekt build, tesztek futtatása, függőségek kezelése, dokumentáció kezelése
- Csomagkezelés: Függőségek automatikus letöltése
- A build folyamat deklaratív leírása
- Fix, előre definiált könyvtárszerkezet
- Jellemzően Java, de plugin-ok segítségével más nyelveket is tud kezelni
- `pom.xml`
- `http://maven.apache.org/index.html`

Projekt

- Project Object Model
- A projektet egyértelműen azonosítja a projekt csoportjának, azonosítójának és verziójának hármasa (group, artifact id, version)
- Egy projekt több modulra bontható, a modulokat külön is tudjuk kezelni.

pom.xml

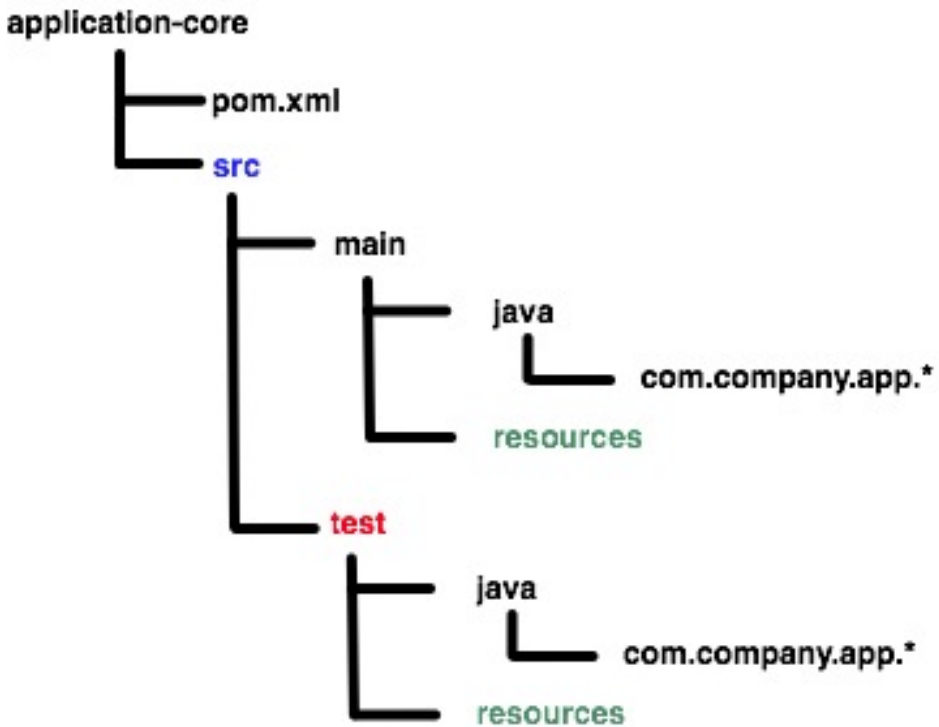
Itt adjuk meg a projekt lényeges információit:

- groupId, artifactId, version megadása
- hogyan fordítsunk
- mi legyen a fordítás eredménye
- tesztesetek
- függőségek

pom.xml

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
    http://maven.apache.org/xsd/maven-4.0.0.xsd">

  <modelVersion>4.0.0</modelVersion>

  <groupId>com.mycompany.software</groupId>
  <artifactId>app</artifactId>
  <version>1.0-SNAPSHOT</version>
  <packaging>pom</packaging>

    ...

</project>
```

Könyvtár szerkezet



Projekt fordítás fázisai

- validate - validate the project is correct and all necessary information is available
- compile – compile the source code of the project
- test – test the compiled source code using a suitable unit testing framework. These tests should not require the code be packaged or deployed
- package – take the compiled code and package it in its distributable format, such as a JAR.
- integration-test – process and deploy the package if necessary into an environment where integration tests can be run

Projekt fordítás fázisai

- verify – run any checks to verify the package is valid and meets quality criteria
- install – install the package into the local repository, for use as a dependency in other projects locally
- deploy - done in an integration or release environment, copies the final package to the remote repository for sharing with other developers and projects.
- Pontosabb lista: http://maven.apache.org/guides/introduction/introduction-to-the-lifecycle.html
- Command line:

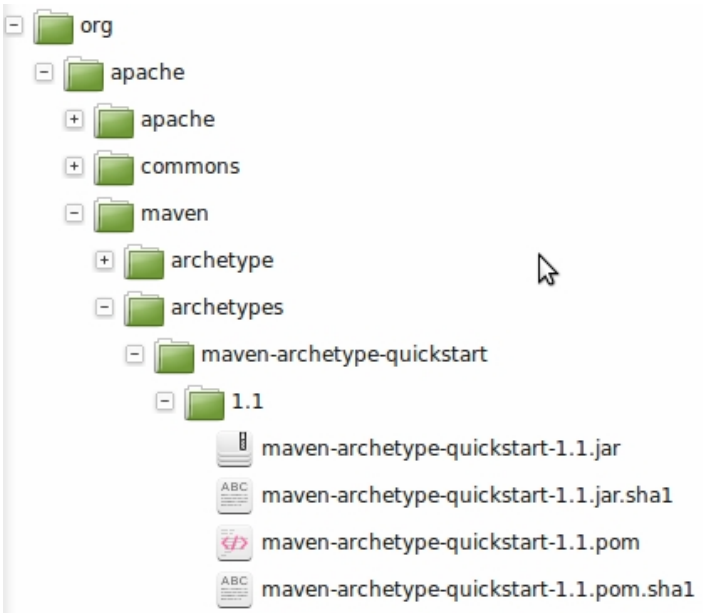
```
mvn clean install
```

Célok (goals)

- A fordítási fázisok célokból épülnek fel
- A cél egy olyan taszk, ami a projekt fordításával illetve menedzselésével kapcsolatos.
- Egy maven parancsnál több cél is megadható, ezek végrehajtásának sorrendje függ a felsorolástól és attól, hogy melyik fázishoz köthetőek.
- Célok-at (goal) lehet definiálni (a pom.xml-ben)

Repository

- Centrális: `http://repo.maven.apache.org`
- Nexus
- Lokális: saját gépünkön található maven repository
  - `~/.m2/repository`



Repository

- Első build-eléskor letölti a függőségeket, plugin-okat, majd a lokális repository-ba tárolja el
- jelentősen megnöveli a hálózati forgalmat, lassítja a build-elési folyamatot
- Maven konfiguráció megváltoztatása, hogy az általunk megadott repo-t/mirror-t használja:
  `~/.m2/settings.xml`

Buildelés eredményei

- Fordítás során keletkezik egy `target` könyvtár, ebbe kerülnek a fordítás során létrehozott fájlok.
- kimenet, pl a `my-app-1.0-SNAPSHOT.jar`
- `classes` könyvtár – osztályok, amelyek a fordítás során keletkeztek (de nem a teszt-osztályok)
- `test-classes` – osztályok, amelyek a teszt-forrásfájlokból keletkeznek
- `maven-archiver` – egy `pom.properties` file, ami leírja a projektet azonosító hármast (groupId, artifactId, version)
- `surefire-reports` – ide kerülnek a tesztelés eredményei

Project hierarchiák

```
<project ...>

  <modelVersion>4.0.0</modelVersion>

  <groupId>com.mycompany.app</groupId>
  <artifactId>parent-app</artifactId>
  <version>1.0-SNAPSHOT</version>
  <packaging>pom</packaging>
  <!-- alprojektek felsorolása-->
  <modules>
    <module>first-child-app</module>
    <module>second-child-app</module>
  </modules>
</project>
```

Project hierarchiák

```
<project ...>

  <modelVersion>4.0.0</modelVersion>

  <parent>
    <groupId>com.mycompany.app</groupId>
    <artifactId>parent-app</artifactId>
    <version>1.0-SNAPSHOT</version>
  </parent>

  <groupId>com.mycompany.app</groupId>
  <artifactId>first-child-app</artifactId>
  <version>1.0-SNAPSHOT</version>
  <packaging>war</packaging>
    ...
```

Plugin-ok

- A maven funkcionalitása az alap dolgokra korlátozódik, bár különböző plugin-ok használatával bármit meg lehet tenni: pl C++, LaTeXfordítás, ant build, javadoc.
- Mindenki írhat magának külön plugin-t.

Pl. javadoc plugin

```
<project ...>
  <build>
    <plugins>
      <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-javadoc-plugin</artifactId>
        <version>2.8.1</version>
        <configuration>
          ...
        </configuration>
      </plugin>
    </plugins>
  </build>
  ...
</project>
```

Függőségek megadása

```
<project ...>

  <dependencies>
    <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>3.8.1</version>
      <scope>test</scope>
    </dependency>
  </dependencies>
</project>
```

- groupId + artifactId + version azonosítja a projektet, amitől függünk
- scope megadja, hogy melyik életciklusban használjuk a meghatározott függőséget

Függőségek scope-ja

- A leggyakrabban használt scope-ok:
  - compile – Ez a default, ha nincs megadva. Fordításhoz szükséges függőségek.
  - runtime – Ez azt jelzi, hogy a függőség futásidőben szükséges, fordításkor nem.
  - test – Jelzi, hogy a függőség nem szükséges a normál működéshez, de a teszt-fordításhoz és a tesztek futtatásához kell.

Gradle

- Nyelv-független, Groovy-alapú build system
- projects, build tasks, task relations és artifact dependencies Groovy nyelven írhatóak le
- dependency resolution (akár maven-kompatibilis módon)
- build logika plugin-okban (core-set + third-party plugins)
- gradlew
- Konfiguráció, build gráf

Gradle előnyök

- Inkrementális build
- Párhuzamosság
- Teljes Java funkcionalitás elérhető
- Szabadabb maven-hez képest

Gradle hátrányok

- DSL szintaxis: sokszor nehezen átlátható (pl. zárójelek), alternatív leírások
- Hibaüzenetek

Problémák

- A verziókövetés nem garantálja, hogy a repository-ban lévő kód korrekt
  - Honnan derül ki, hogy korrekt-e?
  - Mikor update-lhetik-e a saját másolatukat a fejlesztők?
  - A forráskód mely részét update-lhetik a fejlesztők?
  - Melyik revision-re álljunk vissza, ha az aktuális verzió nem jó?
  - Mi legyen, ha egyik fejlesztői gépen működik egy funkcionalitás, másikon nem?
  - Mikor romlott el, mióta nem jó?
  - Ki a felelőse a problémának? Kinek szóljunk?
- Egy hiba kijavítása elront-e más funkcionalitást?
- Hogyan állítható össze egy release gyorsan?

Definíció

- szoftverfejlesztési gyakorlat
- naponta többszöri (gyakori) integráció
- automatikus fordítás, forráskód elemzés statikus analízissel (lint, style checking, metrikák)
- tesztelés a sikeres fordítás végeztével
- XP fejlesztési módszertan
- `http://martinfowler.com/articles/continuousIntegration.html`

Alapelvek

- Használjunk repository-t (megfelelően) (Single Source Repository)
- A fordítási (build) folyamatot automatizáljuk
- Tegyük a build folyamatot öntesztelővé
- Minden nap mindenki legalább egyszer a változtatásait commit-olja be a főágra
- Jó lenne az összes revision-t lebuildelni egy integrációs gépen

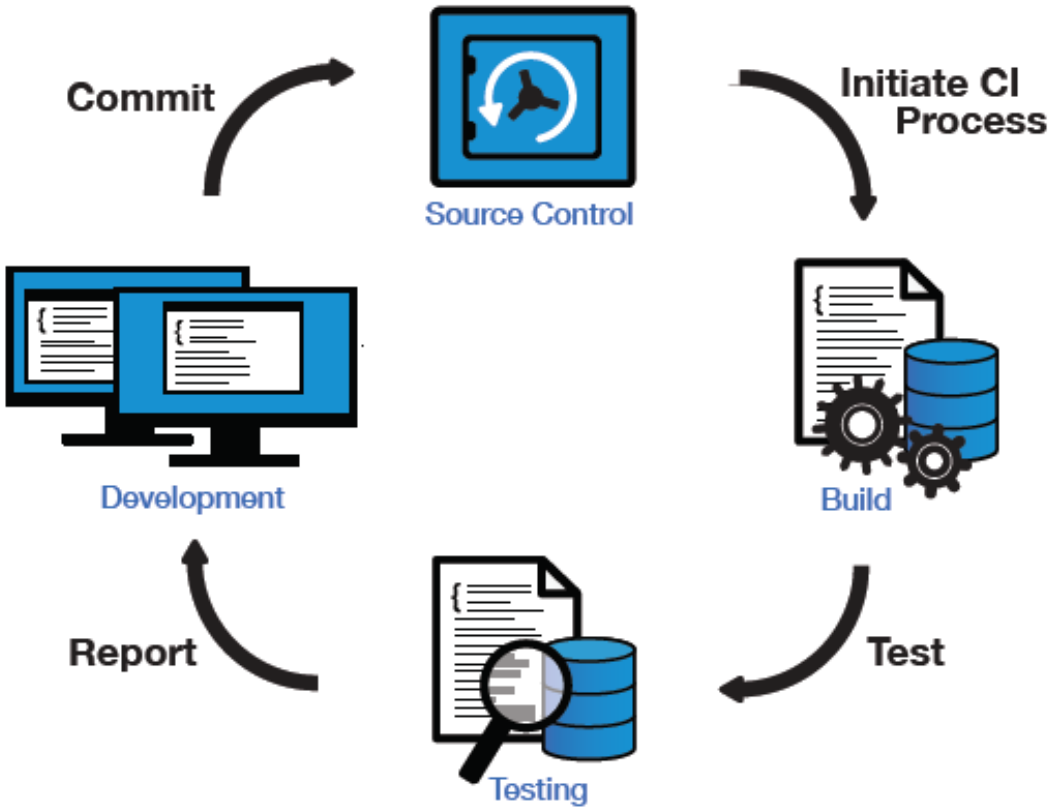
Alapelvek

- Tartsunk a build-elés időtartamát alacsonyan
- Az éles környezet egy másolatán teszteljünk
- Tartsuk az utolsó futtatható rendszert bárki által könnyen elérhetően
- Mindenki láthassa, hogy mi történik
- Automatizáljuk a telepítési, futtatási (deployment) folyamatot

CI Előnyök

- Gyorsan kiderül, ha nem fordul le a repository-ban lévő forráskód
- Gyorsan kiderül, ha valamelyik teszteseten nem megy át a kódunk
- Tisztább, átláthatóbb, hogy ki vagy mi okozta a hibát.
- Kiderül, hogy melyik revision-t használhatjuk, ha nem fordul az aktuális
- Korábbi build-ek adatainak tárolása
- Fejlesztők informálása automatikusan, pl. email-ben
- Referenciát, hivatkozási alapot biztosít

CI Ciklus


Commit
Source Control
Initiate CI Process
Development
Build
Report
Testing
Test

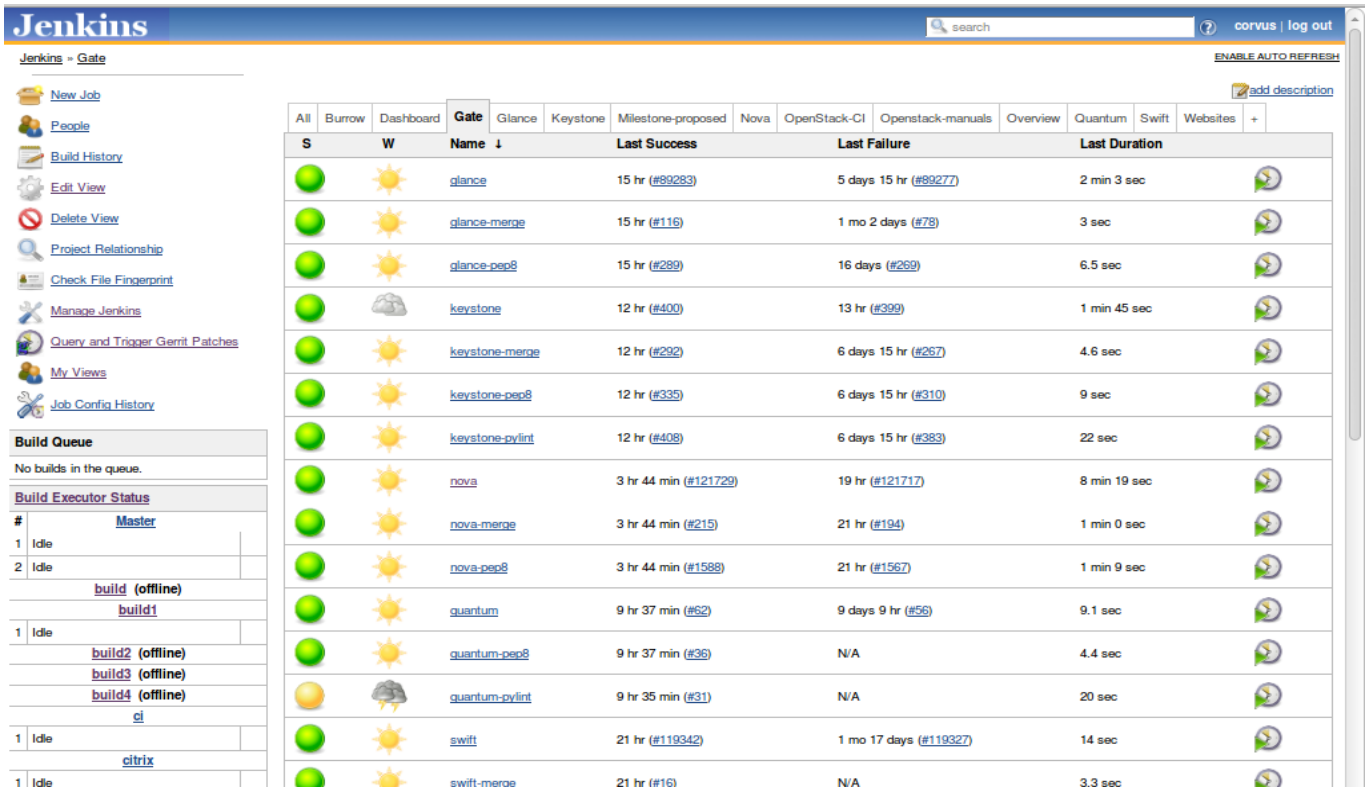
CI Eszközök / Rendszerek

- CruiseControl
- Hudson / **Jenkins**
- IBM Rational Team Concert
- Microsoft Team Foundation Server
- stb.

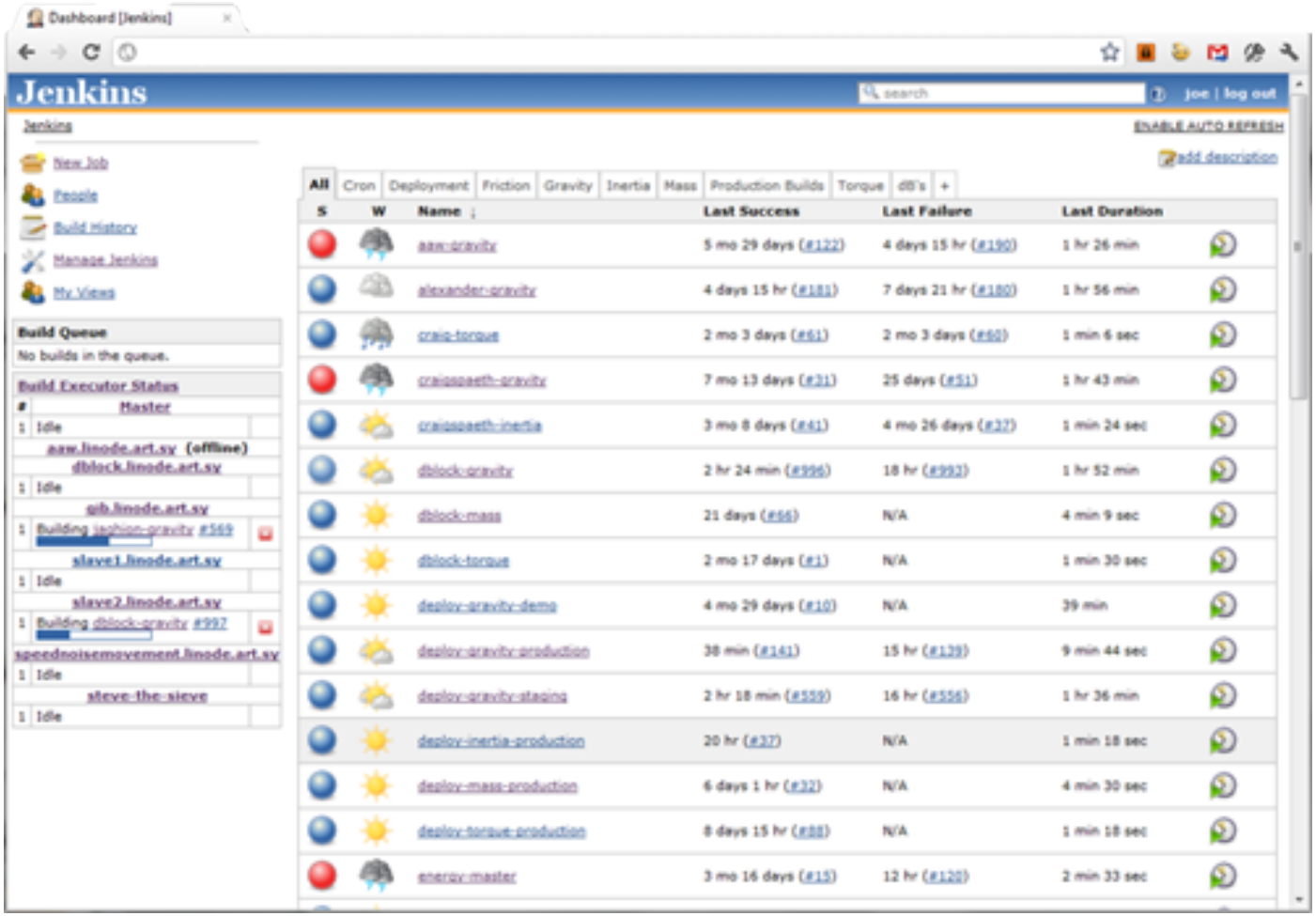
Jenkins

- CI eszköz, rendszer
- Bővíthető, nyílt forrású CI szerver
- `http://jenkins-ci.org/`

Job-ok


Jenkins
search
corvus | log out
Jenkins » Gate
ENABLE AUTO REFRESH
add description
New Job
People
Build History
Edit View
Delete View
Project Relationship
Check File Fingerprint
Manage Jenkins
Query and Trigger Gerrit Patches
My Views
Job Config History
Build Queue
No builds in the queue.
Build Executor Status
All Burrow Dashboard Gate Glance Keystone Milestone-proposed Nova OpenStack-CI Openstack-manuals Overview Quantum Swift Websites +
S W Name ↓ Last Success Last Failure Last Duration
glance 15 hr (#89283) 5 days 15 hr (#89277) 2 min 3 sec
glance-merge 15 hr (#116) 1 mo 2 days (#78) 3 sec
glance-pep8 15 hr (#289) 16 days (#269) 6.5 sec
keystone 12 hr (#400) 13 hr (#399) 1 min 45 sec
keystone-merge 12 hr (#292) 6 days 15 hr (#267) 4.6 sec
keystone-pep8 12 hr (#335) 6 days 15 hr (#310) 9 sec
keystone-pylint 12 hr (#408) 6 days 15 hr (#383) 22 sec
nova 3 hr 44 min (#121729) 19 hr (#121717) 8 min 19 sec
nova-merge 3 hr 44 min (#215) 21 hr (#194) 1 min 0 sec
nova-pep8 3 hr 44 min (#1588) 21 hr (#1567) 1 min 9 sec
quantum 9 hr 37 min (#62) 9 days 9 hr (#56) 9.1 sec
quantum-pep8 9 hr 37 min (#36) N/A 4.4 sec
quantum-pylint 9 hr 35 min (#31) N/A 20 sec
swift 21 hr (#119342) 1 mo 17 days (#119327) 14 sec
swift-merge 21 hr (#16) N/A 3.3 sec

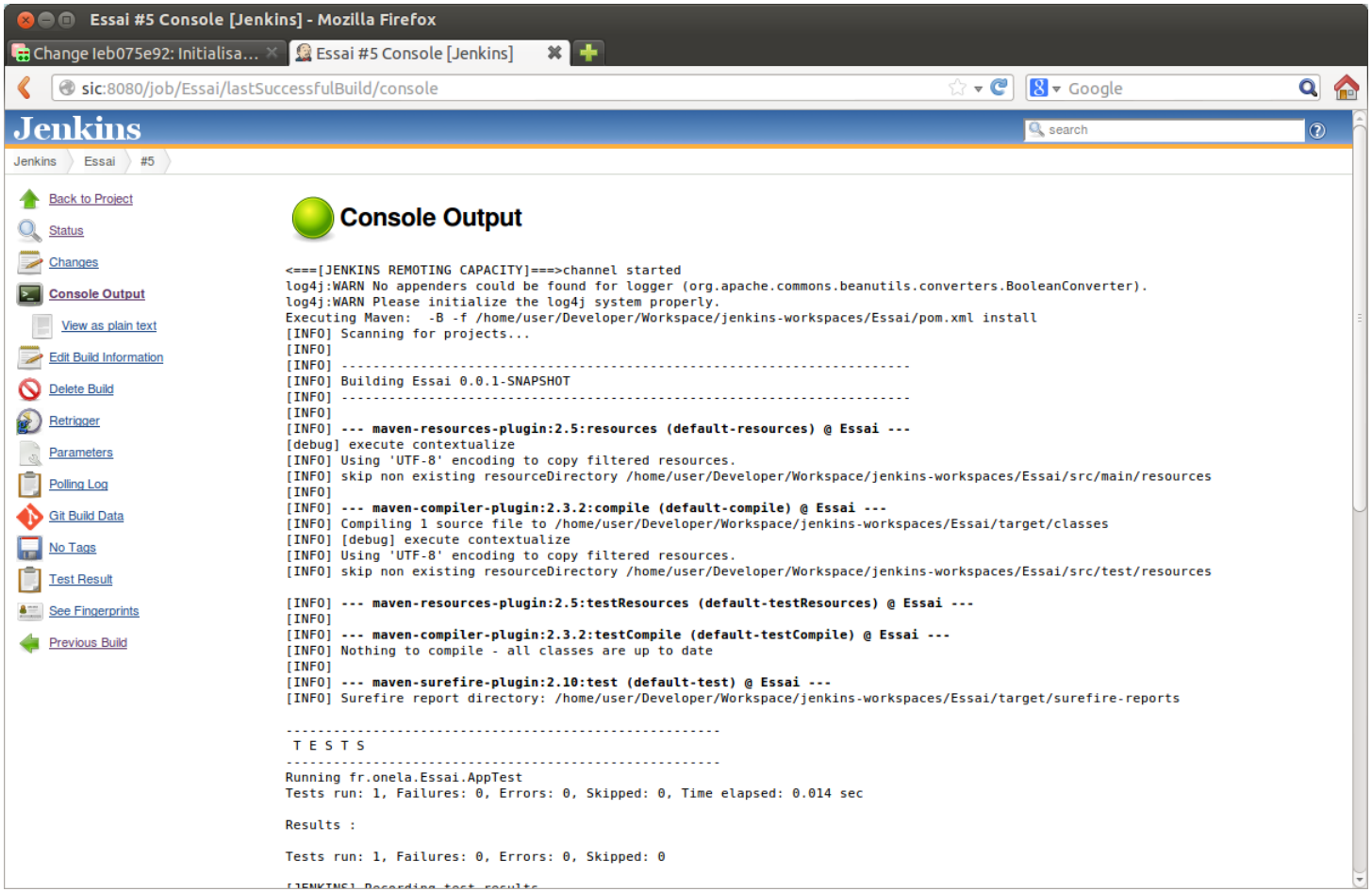
Job-ok


Dashboard [Jenkins]
Jenkins
search
joe | log out
Jenkins
ENABLE AUTO REFRESH
New Job
People
Build History
Manage Jenkins
My Views
add description
Build Queue
No builds in the queue.
Build Executor Status
Master
1 Idle
aaw.linode.art.sy (offline)
dblock.linode.art.sy
1 Idle
gib.linode.art.sy
1 Building jashion-gravity #569
slave1.linode.art.sy
1 Idle
slave2.linode.art.sy
1 Building dblock-gravity #997
speednoisemovement.linode.art.sy
1 Idle
steve-the-sieve
1 Idle
All
Cron
Deployment
Friction
Gravity
Inertia
Mass
Production Builds
Torque
dB's
+
S
W
Name
Last Success
Last Failure
Last Duration
aaw-gravity 5 mo 29 days (#122) 4 days 15 hr (#190) 1 hr 26 min
alexander-gravity 4 days 15 hr (#181) 7 days 21 hr (#180) 1 hr 56 min
craig-torque 2 mo 3 days (#61) 2 mo 3 days (#60) 1 min 6 sec
craigspaeth-gravity 7 mo 13 days (#31) 25 days (#51) 1 hr 43 min
craigspaeth-inertia 3 mo 8 days (#41) 4 mo 26 days (#37) 1 min 24 sec
dblock-gravity 2 hr 24 min (#996) 18 hr (#992) 1 hr 52 min
dblock-mass 21 days (#66) N/A 4 min 9 sec
dblock-torque 2 mo 17 days (#1) N/A 1 min 30 sec
deploy-gravity-demo 4 mo 29 days (#10) N/A 39 min
deploy-gravity-production 38 min (#141) 15 hr (#139) 9 min 44 sec
deploy-gravity-staging 2 hr 18 min (#559) 16 hr (#556) 1 hr 36 min
deploy-inertia-production 20 hr (#37) N/A 1 min 18 sec
deploy-mass-production 6 days 1 hr (#32) N/A 4 min 30 sec
deploy-torque-production 8 days 15 hr (#88) N/A 1 min 18 sec
energy-master 3 mo 16 days (#15) 12 hr (#120) 2 min 33 sec

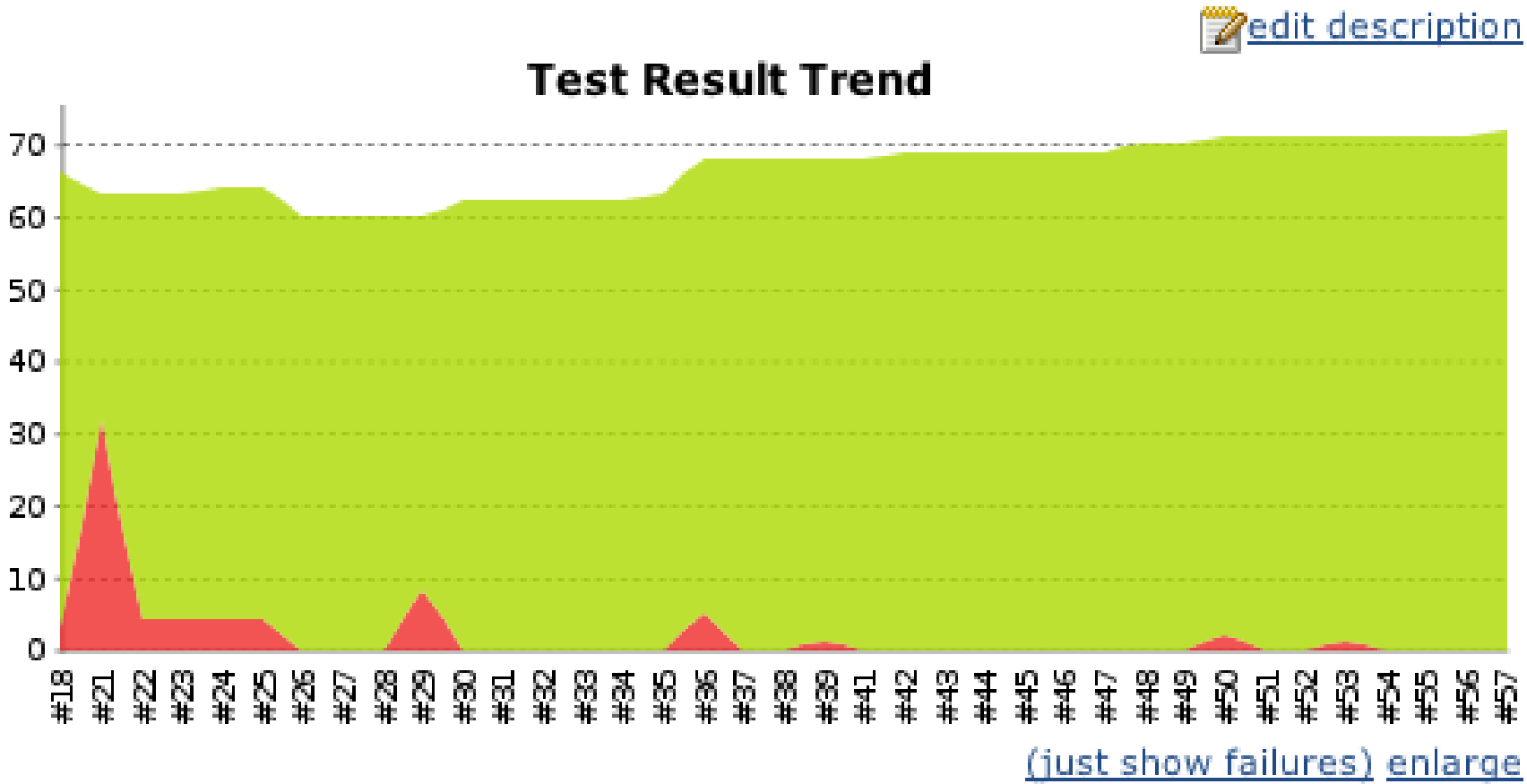
Jobok futtatása

- SCM, Software configuration management (`*/5 * * * *`)
- Másik job befejeződésével vagy meghívásával (pipeline)
- Timer, cron job jelleggel (`00 23 * * *`)
- Manuálisan indítva

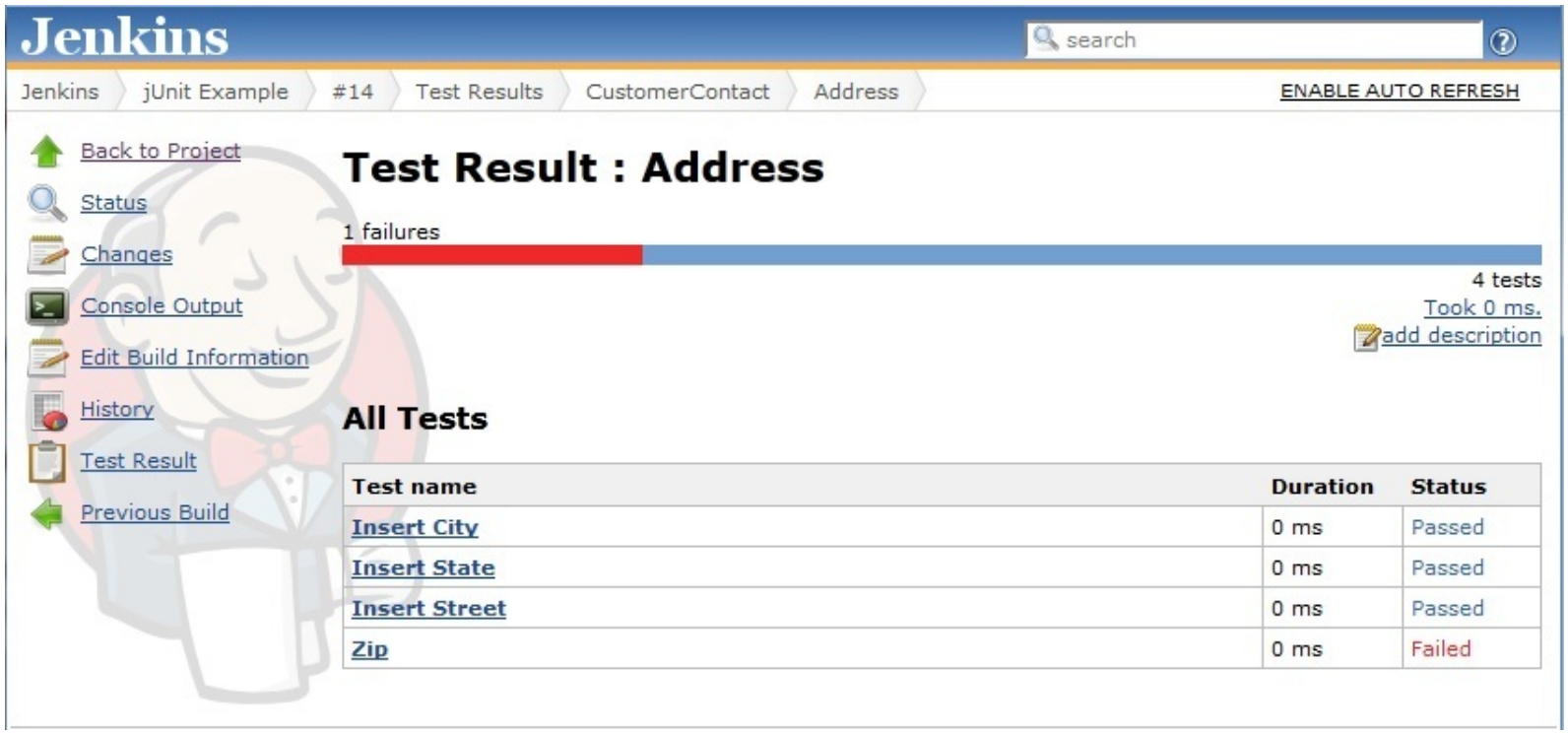
Console output



Test trend


edit description
Test Result Trend
70
60
50
40
30
20
10
0
#18 #21 #22 #23 #24 #25 #26 #27 #28 #29 #30 #31 #32 #33 #34 #35 #36 #37 #38 #39 #41 #42 #43 #44 #45 #46 #47 #48 #49 #50 #51 #52 #53 #54 #55 #56 #57
(just show failures) enlarge

Test futtatás



Plugin-ok

- Több, mint 600 plugin
- SCM: svn, perforce, git, mercurial, clear case stb.
- shell, batch futtatás
- Build: MSBuild, CMake, ant, maven, groovy, stb.
- Test: CppTest, JUnit, Selenium, stb.
- Static analyzers: Cppcheck, CheckStyle, stb.
- Code coverage: gcov, Clover, stb.
- Egyebek: claim, scp, radiator, lava lamp, GUI, stb.

Plugin-ok


myadmin_ci_deploy #61
gwt-commons #20
myadmin_distros #214
myadmin_ci #305
walldisplay disabled job #1
walldisplay_parameter_job (danny super duper) #40
walldisplay-plugin #6
walldisplay_failed_job #56
walldisplay_good_job #50

Összefoglalás

- Modern build eszközök: platform-függetlenség
- Continuous integration: megkönnyíti a csoportos szoftverfejlesztést
- Jenkins: CI server, plugin-ok