APIS IT d.o.o.

information systems and

information tehnologies

support agency

si sapis
sis apis
if you are wise, be a bee

# APIS IT - Razvoj J2EE aplikacija korištenjem open source tehnologija


Dražen Grabovac

SW arhitekt

# Sadržaj

- APIS IT
- Uvod
- Arhitektura i dizajn sustava
- Razvojna i runtime okolina
- Maven
- Testiranje
- Kontinuirana integracija i Sonar
- Zaključak
- Pitanja

**'60**

‘64. Osnovan Centar za ekonomski razvoj grada Zagreba
‘69. Instaliran prvi IBM System 360-30

**'70**

‘72. On-line obrada,
‘72. Projekt IS Grada Zagreba
‘74 . Podrška za komunalne poslove

**'80**

‘87. Informatička podrška Univerzijade .''87
‘88. IS Porezne uprave

**'90**

‘91. IS Carinske uprave
‘92. GIS Grada Zagreba
‘96. Parallel sysplex u produkciji
‘97. Transformacija u GZAOP

**'00**

‘00. IS provedbe izbora za RH
‘01. Projekt Zagreb on line
‘02. Pružene prve e-usluge
‘05. Transformacija u APIS IT d.o.o.
‘06. ePoreza i eCarine

**'10**

‘07. Moja uprava
‘08. OIB
‘08. Jedinstvena uplatnica
’09. ZIS
‘10. Umrežena uprava – OIB 2. faza

**5 PODRUČJA PODRŠKE**

1. UPRAVLJANJE PODACIMA
2. POSLOVNI PROCESI
3. APLIKACIJE
4. APLIKATIVNA INFRASTRUKTURA
5. SISTEMSKA INFRASTRUKTURA

47 godine izgradnje ključnih IS sustava RH
razumijevanje poslovnih procesa uprave
najveći i najiskusniji IT integrator u RH

Si sapis sis apis
[ if you're wise, be a bee ]

# Uvod

- Sustavi koji se izrađuju postaju sve kompleksniji i jača opterećenje na razvoj da isporuči proizvod brže i kvalitetnije
- Naknadno se dobivaju korisnički zahtjevi koji nekada mijenjaju i arhitekturu
- Zahtjeva se sve veće povezivanje sa vanjskim sustavima
- Smjernice koje umanjuju probleme prilikom razvoja:
  - Brzo i jednostavno izdavanje nove verzije aplikacije koja je gotovo uvijek spremna za isporuku
  - Kvaliteta koda mora biti konstantno osigurana
  - Ispunjavanje novih zahtjeva ne dovodi u pitanje konzistentnost sustava
  - Visoka pokrivenost testovima, što više automatizirano testirati
  - Jednostavno i sustavno upravljanje zavisnostima aplikacije
  - Integracija aplikacije sa drugim sustavima korištenjem sučelja i standarda

# Arhitektura i dizajn sustava

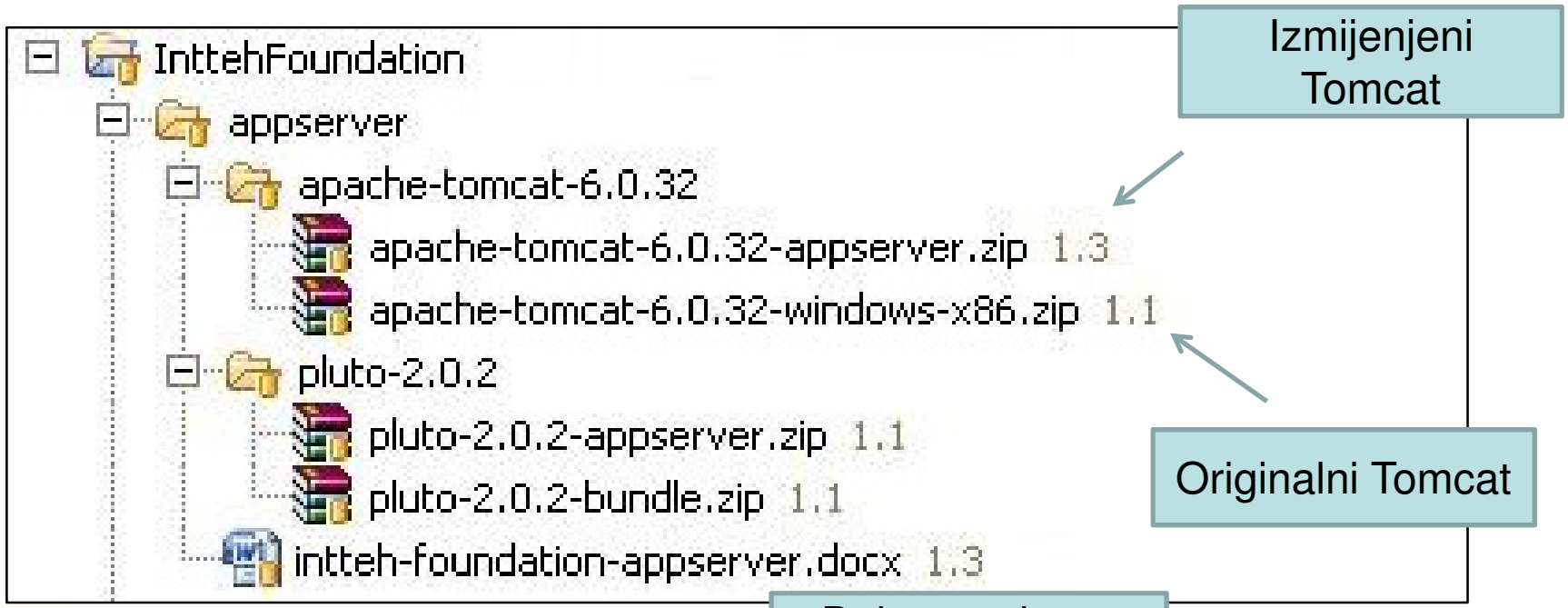
- Spring radni okvir koristimo u svim slojevima aplikacije
  - Olakšava razvoj zbog integracije sa raznim tehnologijama
  - Promovira programiranje korištenjem sučelja i time olakšava testiranje
  - Fleksibilna konfiguracija sa XML-om i anotacijama
- Na prezentacijskom sloju koristimo Spring MVC:
  - Lak za razumijevanje i jednostavan za razvoj
  - Fleksibilan, lako se nadograđuje i testira
  - Podržava razne prezentacijske tehnologije (JSP, JSF, Portal…)
- Servisni sloj je također razvijen sa Spring radnim okvirom:
  - Za pristup bazi se koristi JPA (olakšava razvoj i nismo vezani za jednu bazu)
  - Za komunikaciju između modula koristimo JMS i Spring HttpInvoker
  - Spring olakšava konfiguraciju i integraciju gore spomenutih tehnologija

# Razvojna okolina

- Razvojna okolina prilagođena agilnom razvoju, uspostava za manje od 1 h
- Za servisni sloj koristimo Tomcat sa nekoliko dodataka
  - Tomcat  se pokreće vrlo brzo i ima male zahtjeve za resursima
  - Dodan H2 baza koja je također brza i uzima malo resursa
  - Dodan ActiveMq kojega koristimo za JMS
  - Dodan Bitronix za upravljanje transakcijama
- Za prezentacijski sloj koristimo Pluto
  - Baziran na Tomcatu, pa također radi vrlo brzo
  - Koristimo ga kao Portalnog poslužitelja (implementira JSR-268 specifikaciju)
- STS eclipse
  - Izrađuje je sam SpringSource, pa ima najbolju podršku za Spring
  - Integrirani Maven

# Razvojna okolina

- Razvojne alate držimo na CVS-u, a u njima su:
  - originalni i izmijenjeni Tomcat za servisni sloj
  - originalni i izmijenjeni Pluto za prezentacijski sloj
  - originalni i izmijenjeni STS



# Runtime okolina

- Spring, JPA i kontinuirana integracija nam omogućavaju da imamo drugačiju runtime okolinu
- Runtime okolina omogućava visoku dostupnost i skalabilnost
- IBM Websphere 6.1 (zOS 1.11)
- IBM Portal 6.1 (zLinux 10)
- IBM DB2 9.1 (zOS 1.11)
- IBM MQ 7(zOS 1.11)

# Maven

- Upravljanje projektom, zavisnostima i automatizacija builda
- Standardna struktura projekta i standardni način builda
  - Tko poznaje  Maven, razumije strukturu svakog Maven projekta i kako napraviti build
- Konfiguracija sa pom.xml datotekom
- Mnogo dodataka za razne stvari (testiranje, kompajliranje…)
- Maven se drži principa “konvencija prije konfiguracije”
- Mi konfiguriramo projekt, a razni dodaci kompajliraju datoteke, pokreću testove i sl.
- Ako pokrenemo “mvn package”, napravi se:
  - Kompajliranje Java datoteka
  - Pokrenu se svi unit testovi
  - Naprave se artefakti u direktoriju target

# Maven

- Primjer pom.xml datoteke

```
<project xmlns="http://maven.apache.org/P
  <modelVersion>4.0.0</modelVersion>
  <parent>
    <artifactId>ParentProject</artifactId
    <groupId>hr.apisit.test</groupId>
    <version>1.0.3-SNAPSHOT</version>
  </parent>
  <groupId>hr.apisit.test</groupId>
  <artifactId>TestJava</artifactId>
  <version>1.0.3-SNAPSHOT</version>

  <dependencies>
    <dependency>
      <groupId>commons-cli</groupId>
      <artifactId>commons-cli</artifactId>
    </dependency>
```

Projekti mogu nasljeđivati konfiguraciju od roditelja

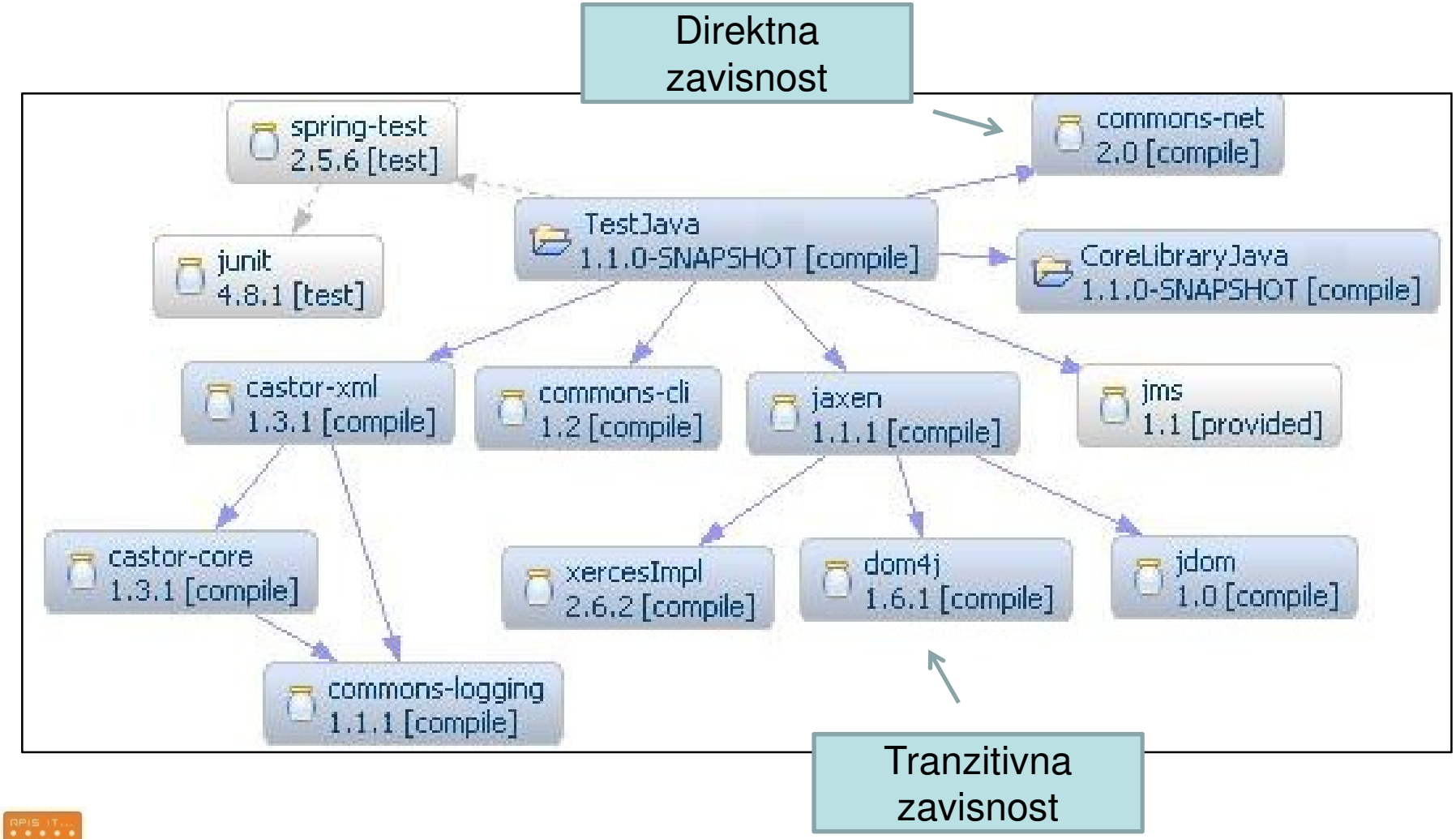
Maven koordinate

Deklarativno opisujemo zavisnosti

# Maven zavisnosti

- Ako nam treba neka biblioteka samo je navedemo (maven koordinate) u pom.xml-u. Maven tada:
  - Skida navedenu biblioteku sa Interneta
  - Skida sve njene zavisnosti
  - Stavlja sve u lokalni repozitorij
  - Ako se koristi proxy, tada se biblioteka cache-ira
- Postoje pretraživači pomoću kojih se pronalaze Maven koordinate
  - http://search.maven.org
  - Unesete commons-lang i dobijete koordinate za pom.xml
- Koristimo Nexus kao interni repozitorij
  - Radi cache korištenih biblioteka sa interneta
  - Služi za objavu internih biblioteka

# Maven zavisnosti



# Testiranje

- Nemam vremena za pisanje testova!
  - Svaki razvojni proces uključuje neki oblik testiranja, automatski ili ručni
  - Automatski testovi rezultiraju bržim razvojnim ciklusima
- Unit testovi
  - Testira se jedna metoda jedne klase
  - Testira se u izolaciji, bez utjecaja drugih komponenti
- Integracijski testovi
  - Integracija komponenti
  - Klasični - npr. DAO komponente
  - Uz pomoć vlastite aplikacije
- End-to-end testovi
  - Testira se grafičko sučelje
  - Prilikom testiranja sustav je potpuno integriran
  - Koristimo Selenium

# Unit testovi

- Testiramo ispravnost pojedine metode u  klasi
- Za svaku klasu koja ima netrivijalni kod
- Preporuke > 90 %
- Kako bi testirali u izolaciji koristimo mockove i stubove
- Primjer – controller:
  - Testiramo svaku metodu controllera
  - Uvjeravamo se da je izlaz controllera ispravan i da je servisni sloj pozvan
  - Potrebno je glumiti servletnog/portletnog poslužitelja i servisni sloj
  - Poslužitelja glumimo uz pomoć stub komponenti (request, response…)
  - Servisni sloj glumimo uz pomoć mocka (Mockito)

# Integracijski testovi

- Klase se kombiniraju i testiraju se kao cjelina
- Događa se poslije unit testiranja
- Spring daje dobru podršku za ovakva testiranja
- Primjer testiranja DAO komponenti:
  - Posebna Spring konfiguracija poveže sve potrebne komponente za test
  - Koristi se H2 baza u memoriji
  - Prije testa se izvršavaju SQL skripte kako bi pripremili H2 bazu
  - Biblioteka Unitils služi za unos i usporedbu testnih podataka

# Integracijski testovi

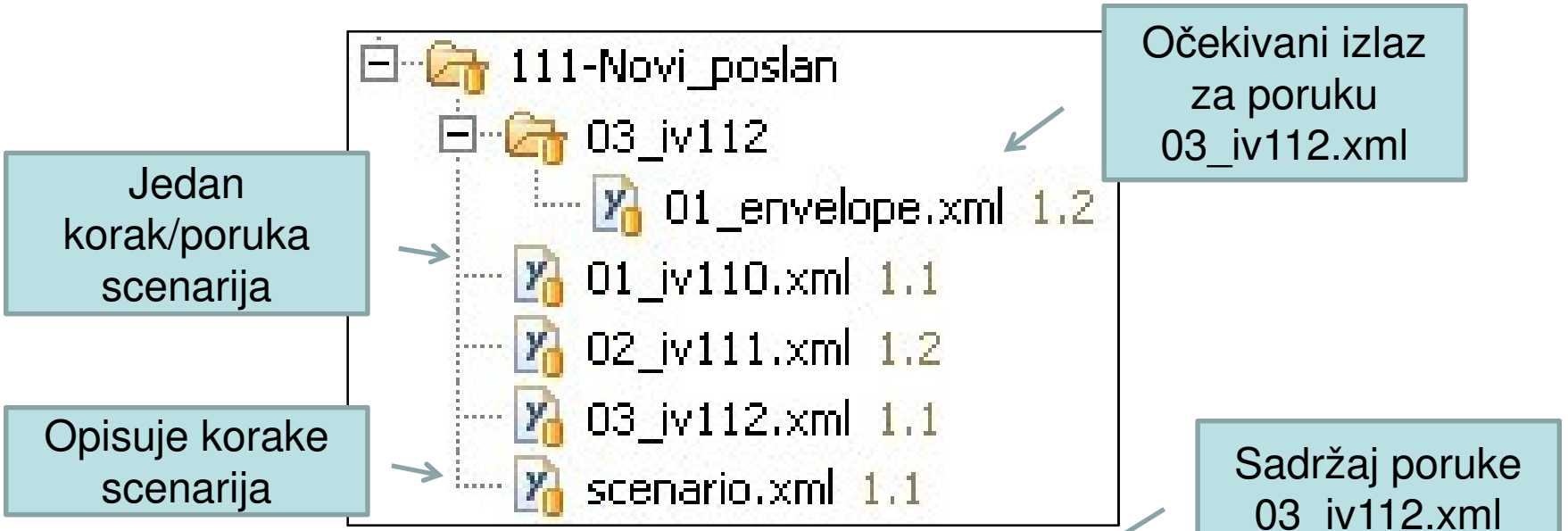
- Zbog poslovnih zahtjeva razmjene XML poruka radimo posebne integracijske testove
- Testovi koji glume poruke od drugih sustava i korisnika
- Provjeravamo izlazne poruke, statuse, iznimke…
- Razvijena vlastita testna aplikacija
  - Izvršava korake scenarija opisane u posebnoj datoteci
  - Koraci scenarija su najčešće XML poruke za servisni sloj
  - Nakon poslane XML poruke aplikacija verificira ispravnost izlaza
  - Verificiraju se promjene najvažnijih polja u bazi, izlazne poruke, njihov sadržaj i sl.
- Testovi se implementiraju prema unaprijed napisanim primjerima
- Noćni build provjerava ispravnost ovih testova

# Integracijski testovi

- Nastojali smo opisati kompletan sustav uz pomoć primjera
- Dokument sadrži stotinjak scenarija s kojima se pokušala pokriti kompletna funkcionalnost servisnog sloja

| Akcija | UC/SC | Komentar | Izlaz | Status draft |
|---|---|---|---|---|
| IV110 | 0101/1 | Ekonomski subjekt ima ispravne podatke | - | **110** |
| IV111 | 0101/1,2,3 | Korisnik unosi podatke o ekonomskom subjektu | - | **110** |
| IV112 | 0101/4 | Korisnik šalje podatke o ekonomskom subjektu u EU | **IER01** | **111** |

# Integracijski testovi – testne poruke




Sadržaj poruke 03_iv112.xml

```
<IV112 xmlns="http://apisit.hr/id/iv112"
  <office>${office}</office>
  <number>
    <ct:country.code>${countryCode}</ct:country.cod
    <ct:national.number>${nationalNumber}</ct:natio
  </number>
</IV112>
```

Varijable koje aplikacija zamjenjuje

# Integracijski testovi – scenario.xml

```
<Scenario mode="EU" xmlns:xsi="http://www.w3.org/2001
    <Message name="01_iv110">
      <AssertStatus status="110" statusType="draft"/>
    </Message>
    <Message name="02_iv111">
      <AssertStatus status="110" statusType="draft"/>
    </Message>
    <Message name="03_iv112">
      <AssertStatus status="111" statusType="draft"/>
      <AssertMessage name="01_envelope"/>
      <AssertXml name="01_envelope.xml" />
    </Message>
```

Ulazna poruka

Očekivani izlaz

# End-to-end testovi

- Koristimo ih za testiranje grafičkog sučelja jer ono nije pokriveno unit i integracijskim testovima
- Ovakvim testovima testiramo potpuno integrirani sustav
- Koristimo  Selenium:
  - WebDriver ima jednostavan API i razne implementacije (Chrome, IE, Firefox…)
  - Direktno poziva funkcionalnosti preglednika (prije je bio potreban server)
  - Testove pišemo kao i obične unit testove
- Implementacija:
  - Unaprijed smo osmislili i opisali poslovne scenarije u dokumentu
  - Svaki scenarij se vrti sa unaprijed pripremljenim podacima
  - Testovi se izvode u Firefoxu,  glumi se korisnika te verificira rezultate
  - Za neispravne se dobije screenshot
  - Napravili smo vlastiti API kako bi još dodatno olakšali razvoj testova

# End-to-end testovi

Primjer opisa testnog scenarija

| Akcija | Očekivani izlaz |
|---|---|
| Ulogiravanje korisnika **Korisnik01** | Popunjene stavke menija |
| Klik na **prvi** link | lista **vrijednosti** |
| Izabir vrijednosti **1121** | Korisniku su vidljivi svi tabovi. |
| Klik na tab '**Priprema**' | polje za unos **broja** |
| Unosimo eori broj **123456** | **pronađen** 123456 u tablici |

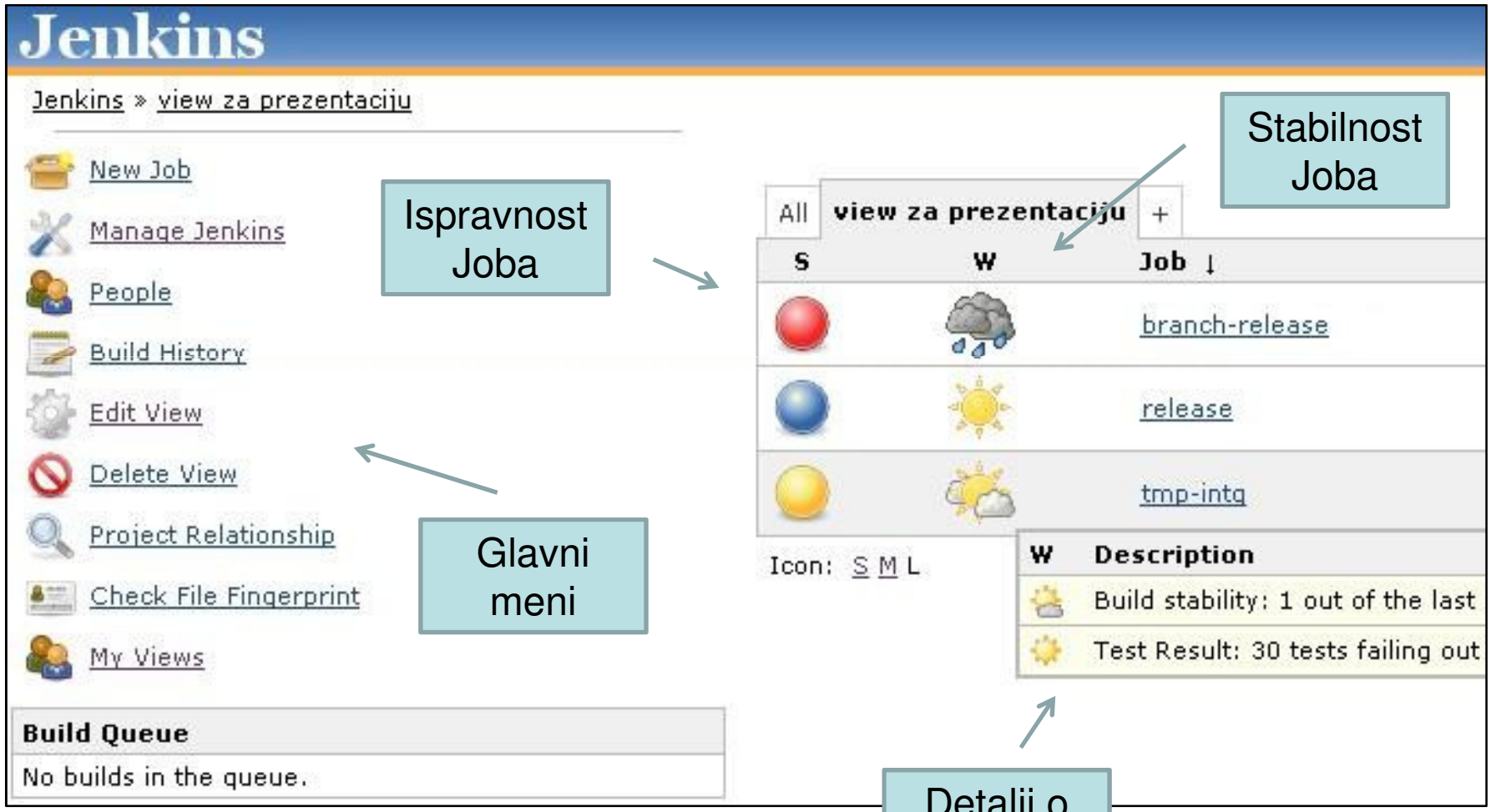
Jednostavan API za pisanje testova

```
enter( "Korisnik01", "1121" );
waitWebElement(
    By.name( "search01_tabPanel01" ) ).click();
waitWebElement(
    By.id( "search02_inputText01" ) ).sendKeys( "123456" );
waitWebElement(
    By.id( "search02_submit" ) ).click();
assertText( By.xpath( "//table[@id='dataTable01']//tr/td[1]//label" ),
    "123456" );
```

# Kontinuirana integracija

- Kontinuirano osiguranje kvalitete koda, a ne na kraju razvoja
- Ranije uočavanje i ispravljanje problema
- Lakše upravljanje promjenama koda, automatizirano testiranje se događa kontinuirano tijekom razvoja
- Omogućava razvoj na jednoj, deploy na drugoj platformi. Razvoj je na platformi Tomcat/Pluto/H2 , a deploy na WAS/Portal/DB2.
- Koristimo Jenkins/Hudson. Popularan i ima mnogo dodataka.
- Radimo tri vrste poslova sa Jenkinsom: build kod svake izmjene na CVS-u, noćni build sa svim testovima, releaseove
- Šalje mailove ako padne build ili testovi. Mail uvijek dobiju arhitekti te programer zbog kojega je build neuspješan

# Kontinuirana integracija - Jenkins



# Kontinuirana integracija - Jenkins

- Build kod svake izmjene na CVS-u
- Provjerava svakih 30 min ima li promjena
- Obavlja sljedeće stvari:
  - Kompajliranje
  - Build
  - Pokretanje unit testova
- Ako ne uspije neki od ovih koraka, dobije se mail

# Kontinuirana integracija - Jenkins

- Noćni build sa svim testovima
- Pokreće se u ponoć ako ima promjena
- Obavlja sljedeće stvari:
  - Kompajliranje
  - Build
  - Pokretanje unit testova
  - Deploy na Websphere/Portal/DB2
  - Pokretanje integracijskih testova
  - Pokretanje end-to-end testova

# Kontinuirana integracija - Jenkins

- Nestabilan noćni build zbog selenium testa

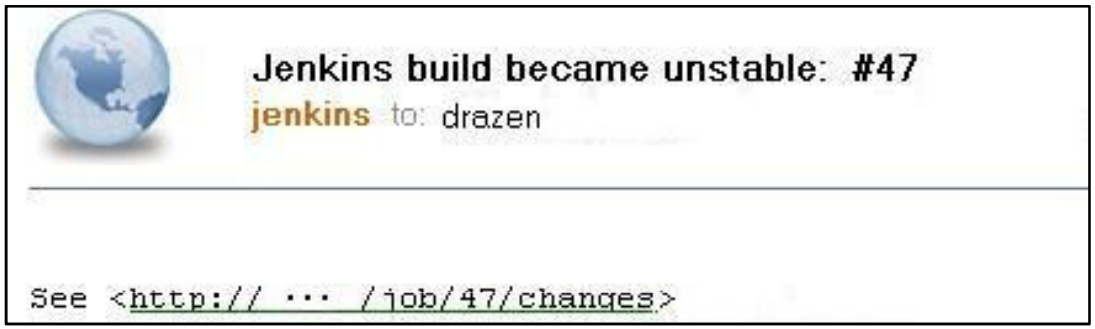

Screenshot selenium testa

Promjene koje su pokrenule build

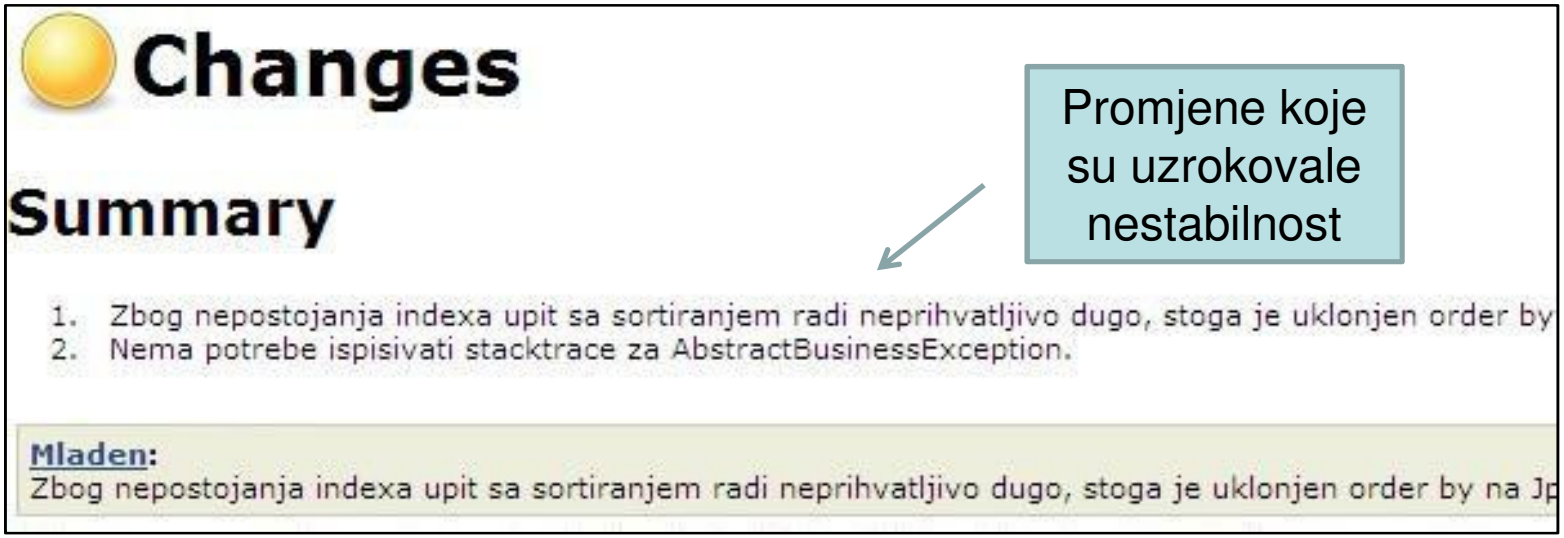
Build je nestabilan zbog selenium testa

# Kontinuirana integracija - Jenkins

- Nestabilan noćni build - mail




Mail sa linkom prema promjenama

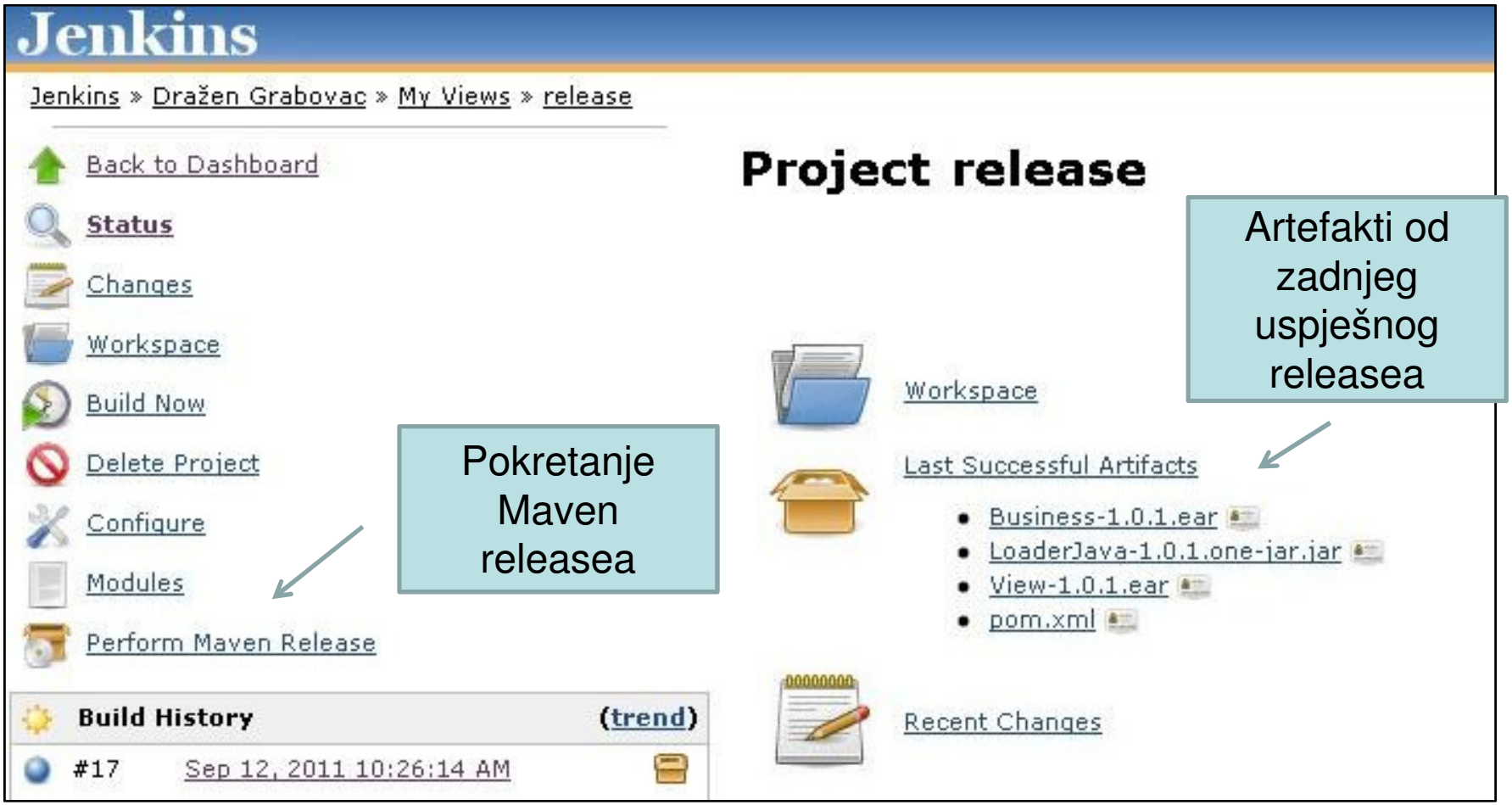



Promjene koje su uzrokovale nestabilnost

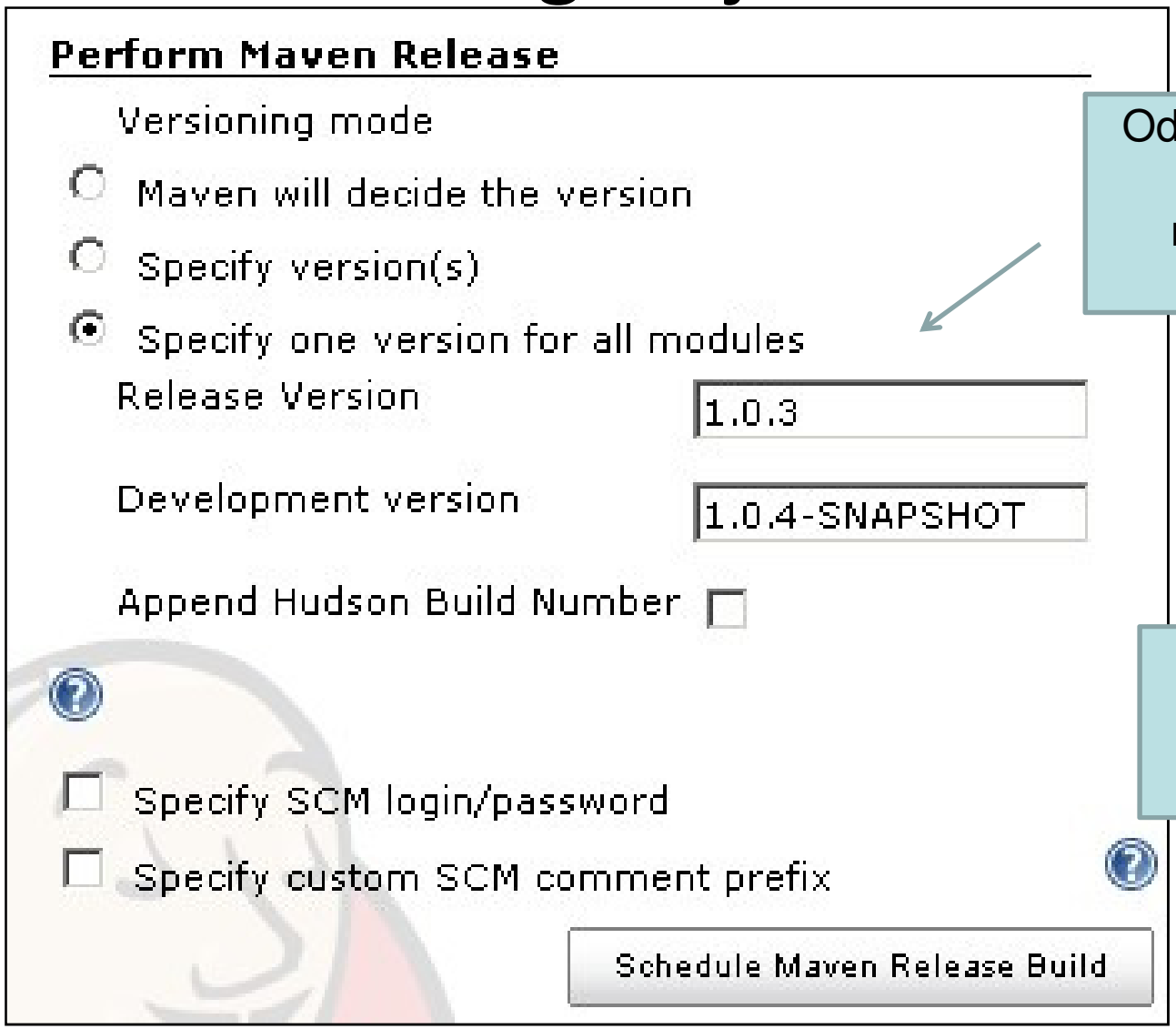
# Kontinuirana integracija - Jenkins

- Release – radimo ga po potrebi
- Razvoj je uvijek u radnoj (SNAPSHOT) verziji za razliku od releasea
- Integriran je sa Jenkinsom (dodatak)
- Obavlja sljedeće stvari za npr. verziju 1.0.3-SNAPSHOT :
  - Uzima projekt sa CVS-a i mijenja verziju u release (1.0.3-SNAPSHOT -> 1.0.3)
  - Radi s njime build i pokreće unit testove
  - Radi tag i commit release verzije ( 1.0.3 -> CVS)
  - Mijenja verziju u novu radnu i stavlja je na CVS (1.0.4-SNAPSHOT -> CVS)
  - Skida tagiranu release verziju (1.0.3) i sa njom proizvodi artefakte koji završe na potrebnim lokacijama (Maven repozitoriju i mrežnom disku)

# Kontinuirana integracija - Jenkins



# Kontinuirana integracija - Jenkins



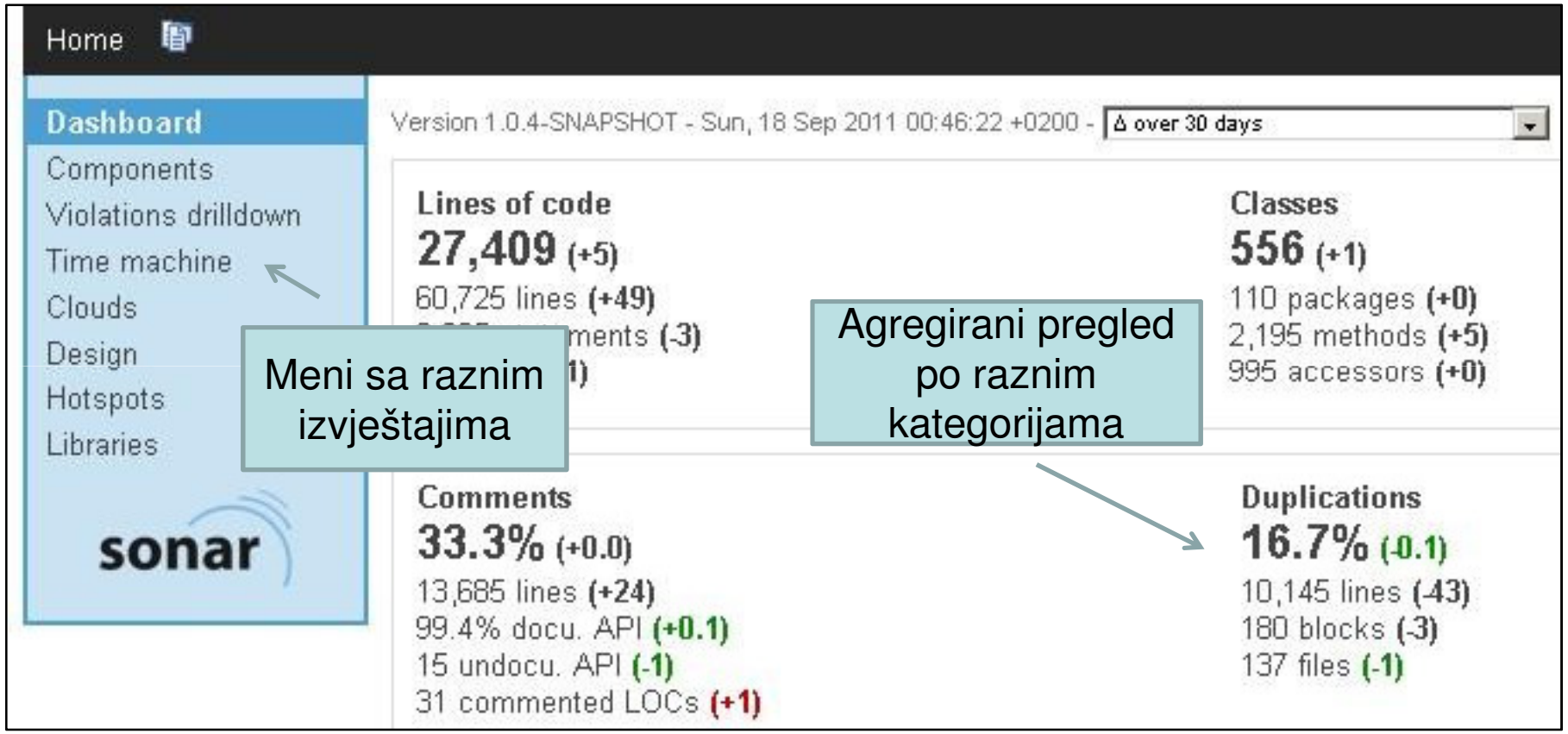

Određujemo radnu i release verziju

Pokreće izradu releasea

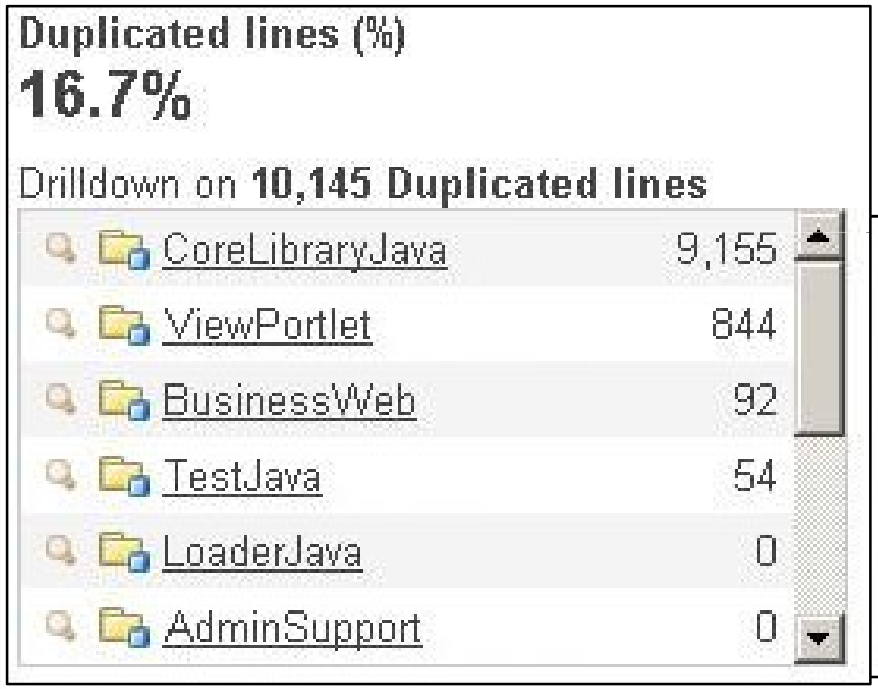
# Sonar

- Kako bi kvaliteta koda bila uvijek osigurana, potrebno ju je nadzirati
- Sonar je alat za upravljanje kvalitetom koda
- Sagledava kvalitetu sa 7 aspekata:
  - komentari
  - arhitektura i dizajn
  - dupliciranje koda
  - unit testovi
  - kompleksnost
  - potencijalni bugovi
  - pravila kodiranja
- Integriran je sa Jenkinsom (dodatak)
- Noćni build šalje podatke Sonaru

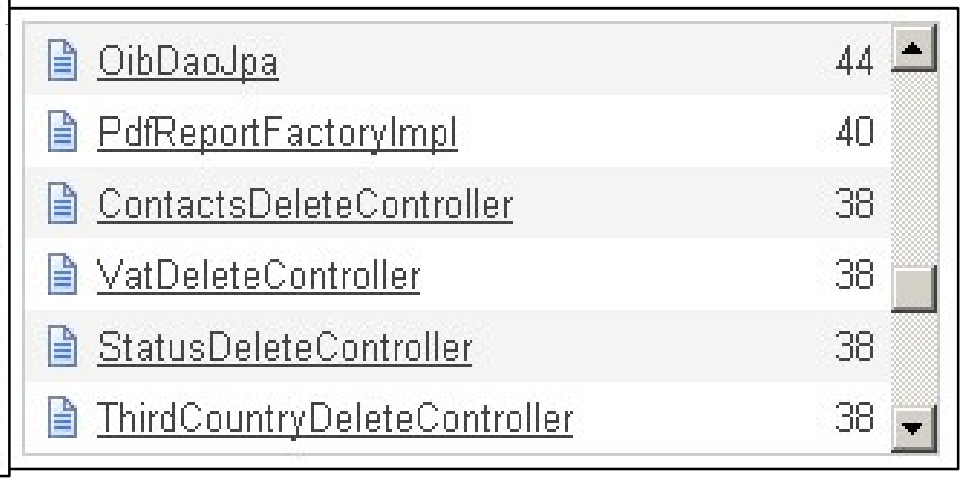
# Sonar


Home
Dashboard
Components
Violations drilldown
Time machine
Clouds
Design
Hotspots
Libraries
sonar
Version 1.0.4-SNAPSHOT - Sun, 18 Sep 2011 00:46:22 +0200 -
Δ over 30 days
Lines of code
27,409 (+5)
60,725 lines (+49)
Classes
556 (+1)
110 packages (+0)
2,195 methods (+5)
995 accessors (+0)
Comments
33.3% (+0.0)
13,685 lines (+24)
99.4% docu. API (+0.1)
15 undocu. API (-1)
31 commented LOCs (+1)
Duplications
16.7% (-0.1)
10,145 lines (-43)
180 blocks (-3)
137 files (-1)
Meni sa raznim izvještajima
Agregirani pregled po raznim kategorijama

# Sonar – primjer dupliciranog koda




Dupliciranje po klasama




Dupliciranje po modulima

# Sonar – primjer dupliciranog koda

| | Nb lines | From line | File | From line |
|---|---|---|---|---|
| collapse | 22 | 156 | Same file | 187 |

```
156     List<Object[]> resultList = find
157     for( Object[] object : resultLis
158         OibUnperformedAnnouncement o
```

Isti kod u istoj klasi na dva mjesta

| | Nb lines | From line | File | From line |
|---|---|---|---|---|
| collapse | 22 | 187 | Same file | 156 |

```
187     List<Object[]> resultList = find
188     for( Object[] object : resultLis
189         OibUnperformedAnnouncement o
```

# Zaključak

- Spring radni okvir olakšava razvoj i testiranje na svakom sloju aplikacije
- Brza i lagana radna okolina može ima veliki utjecaj na produktivnost
- Maven olakšava upravljanje zavisnostima, te uvodi standarde u strukturu i upravlja projektom
- Kvalitetno testiranje svakog sloja aplikacije osigurava kvalitetan proizvod. Potrebno je automatski testirati što više i što ranije, pogotovo kada se nema vremena
- Jenkins pomaže da svaki dan znamo integrira li se naša aplikacija uspješno i rade li nam svi testovi
- Sonar daje lijep pregled kvalitete koda i olakšava nam da se držimo dobrih praksi i standarda

# Hvala na pažnji

PITANJA???