Testing sails.js appliactions with mocha

TDD,BDD,Scene test in sails.
js

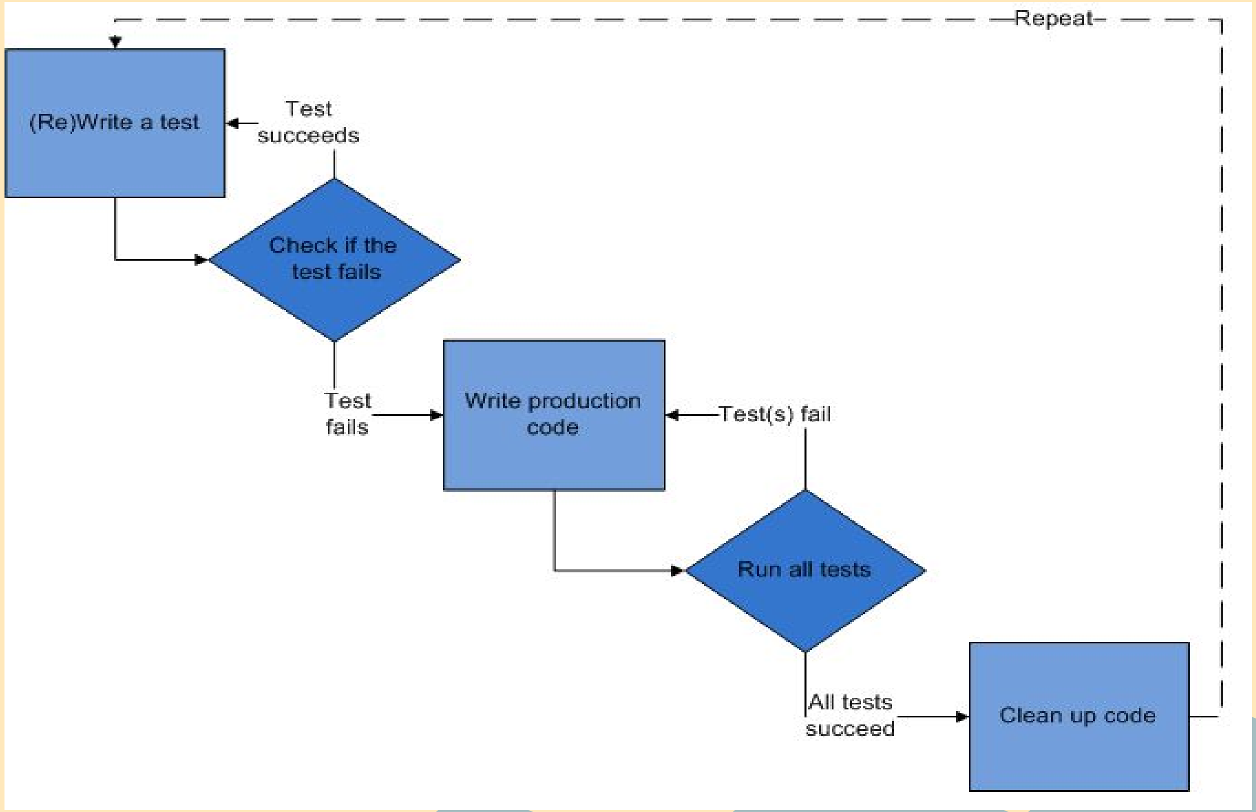
about “TDD”(Test-driven development)

software for TDD:

1. xUnit frameworks (java)
2. TAP results
3. mocha (nodejs)

Test-driven development cycle:

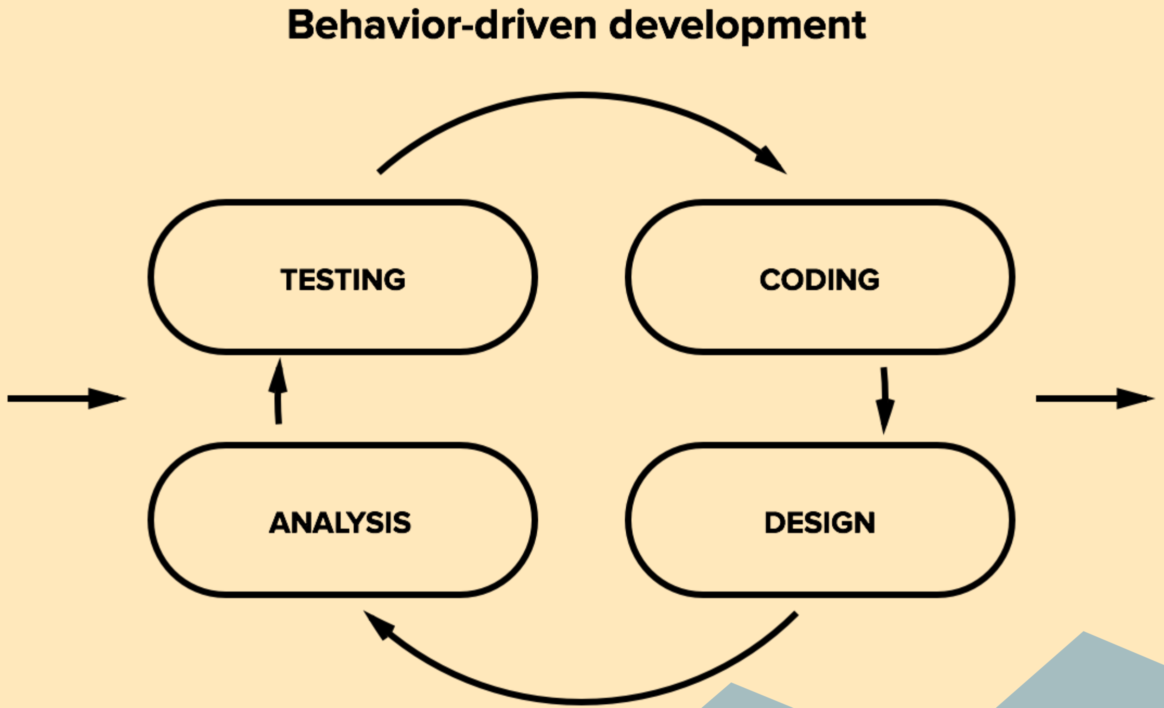
1. **Add a test**
2. **Run all tests and see if the new one fails**
3. **Write some code**
4. **Run tests**
5. **Refactor code**
6. **Repeat**



about “BDD”(Behavior-driven development)

difference between TDD and BDD

1. BDD focuses on the behavioural aspect of the system rather than the implementation aspect of the system that TDD focuses on.
2. BDD gives a clearer understanding as to what the system should do from the perspective of the developer and the customer. TDD only gives the developer an understanding of what the system should do.
3. BDD allows both the developer and the customer to work together to on requirements analysis that is contained within the source code of the system.



TDD in sails.js

example (Test cashflow api) :

```
describe("test cashflow create api!", function() {
  it("should create cashflow and push transaction id to this cashflow !", function(done) {
    CashflowService.create(transactions_id,cashflow,function(err,result){
      result.user_id.should.be.equal(cashflow.user_id);
      result.time.should.be.equal(cashflow.time);
      result.amount.should.be.equal(cashflow.amount);
      result.type.should.be.equal(cashflow.type);
      result.remark.should.be.equal(cashflow.remark);
      result.transactions_ids.should.containEql(transactions_id);
      should.not.exist(err);
      done();
    }) ;
  });
});
```

BDD in sails.js

example (Test User Register):

```
describe("register !",function(){
  it("user register with no error!", function(done) {
   request.post(SERVER_HOST + "/user/register")
    .send(user)
    .end(function(err, res) {
     var result = JSON.parse(res.text);
     var registerFlag = false ;
     if(result.code ==0 || result.code ==1)
      registerFlag = true ;
     registerFlag.should.be.equal(true);
     done();
    });
  });
});
```

Scene Test in our project

example (Test User Register):

1. reset password flow (isResiger_resetPassword_resetPayPassword.js)
2. user register flow (register_login_asset.js)
3. timing provide portfolio value (timing_portfolio_provider.test.js)

Preparation Unit Test In Sailsjs

For our test suite, we use mocha. Before you start building your test cases, you should first organise your ./test directory structure, for example in the following way.

```
./myApp
├── api
├── assets
├── ...
├── test
│   ├── unit
│   │   ├── controllers
│   │   │   └── UsersController.test.js
│   │   ├── models
│   │   │   └── Users.test.js
│   │   └── ...
│   ├── fixtures
│   ├── ...
│   ├── bootstrap.test.js  // This file is useful when you want to execute some code before and after running your tests
│   └── mocha.opts
└── views
```

Tools, tips for Unit Test (1)

- shouldjs (BDD style assertions for node.js)
- assert (nodejs module)
- superagent (the http client use for BDD test)
- timekeeper (the time mock use for scene test)
- barrels (data init.. just in sails.js framework)

Tools, tips for Unit Test (2)

- grunt(run Mocha Test as a development task)
- istanbul(generate code coverage)
- makefile (build your appliaction’s Test)

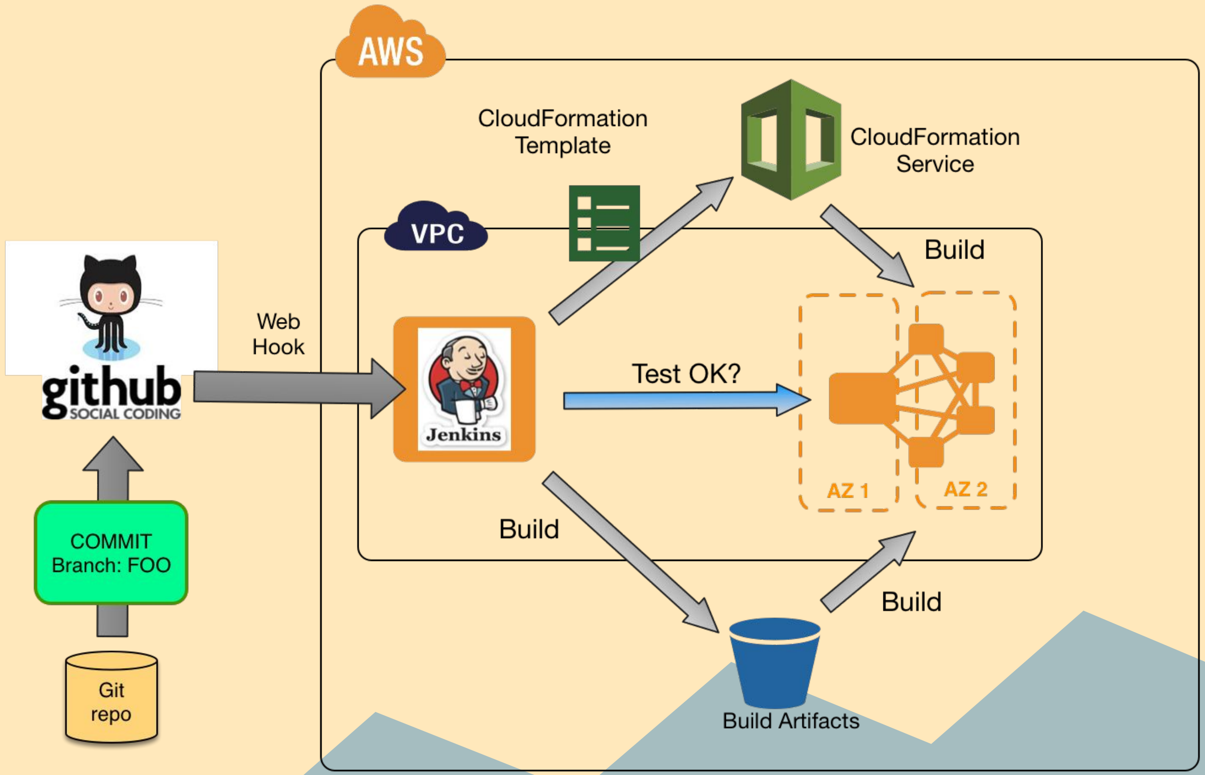
Integrating Test with Jenkins

what is Jenkins?

In a nutshell Jenkins CI is the leading open-source continuous integration server. Built with Java, it provides 1009 plugins to support building and testing virtually any project.

CI best practices

1. Automate the build
2. Make the build self-testing
3. Everyone commits to the baseline every day
4. Every commit (to baseline) should be built
5. Keep the build fast
6. Test in a clone of the production environment
7. Make it easy to get the latest deliverables
8. Everyone can see the results of the



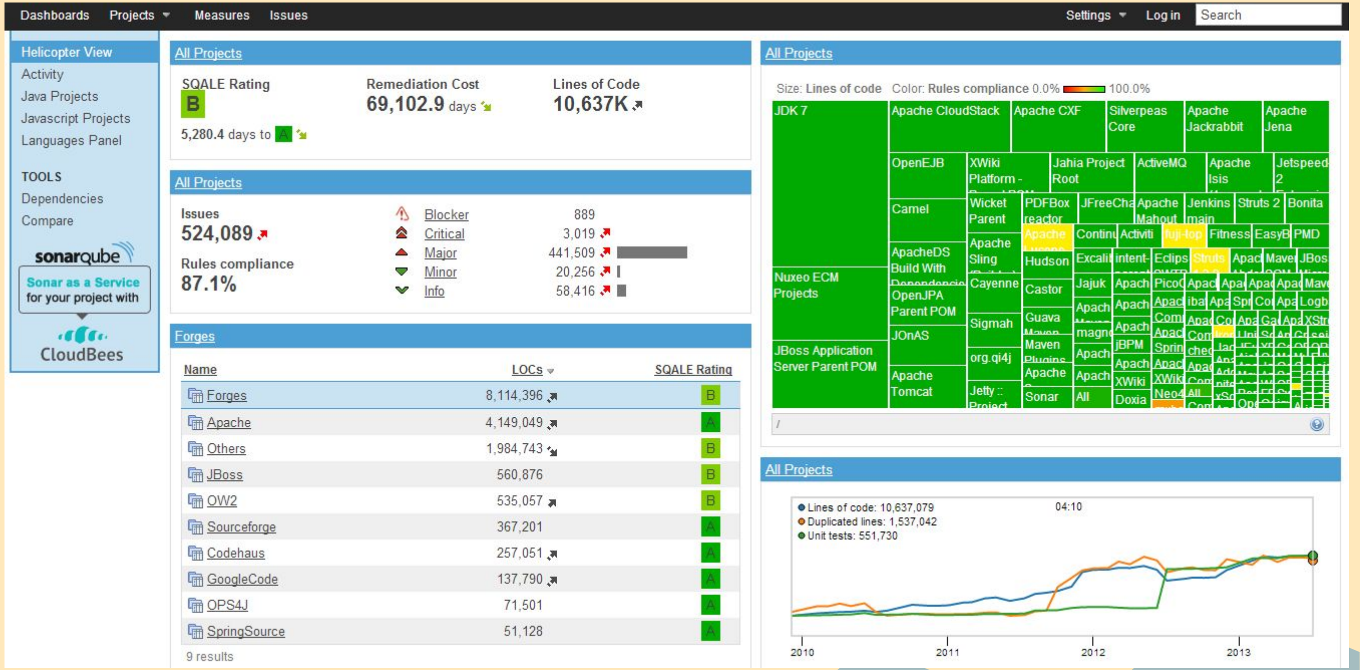
Inspection of code quality with SonarQube

what is SonarQube?

SonarQube (formerly Sonar [1]) is an open source platform for continuous inspection of code quality.

Offers reports on:

1. duplicated code
2. coding standards
3. unit tests
4. code coverage
5. complex code
6. comments
7. architecture



All Projects

SQALE Rating B
5,280.4 days to A

Remediation Cost
69,102.9 days

Lines of Code
10,637K

All Projects

Issues
524,089

Rules compliance
87.1%

Blocker	889
Critical	3,019
Major	441,509
Minor	20,256
Info	58,416

Forges

Name	LOCs	SQALE Rating
Forges	8,114,396	B
Apache	4,149,049	A
Others	1,984,743	B
JBoss	560,876	B
OW2	535,057	B
Sourceforge	367,201	A
Codehaus	257,051	A
GoogleCode	137,790	A
OPS4J	71,501	A
SpringSource	51,128	A

9 results

All Projects

● Lines of code: 10,637,079
● Duplicated lines: 1,537,042
● Unit tests: 551,730
04:10

Q&A

- 测试单条ok，整体跑不ok—— 数据存在依赖
- 实现代码被改，测试代码跟着改 —— 加多了工作，时间增多了
-