

Configuration as code: Teamcity. Первое ЗНАКОМСТВО



Меня зовут Алексей.
Последние 5 лет в ролях
тестировщика-автоматизатора,
фронтенд разработчика и
управленца (тех.лид, тимлид).
В рамках разных проектов, но
одной крупной компании.
От стажёра до ведущего
инженера.

- Teamcity - UI first



- Отлично!


- шаблонные сборки
- мета-раннеры
- наследование
- pipeline
- параметры сборки
- аудит



хранение настроек teamcity в системах контроля версий Teamcity 9.0

- XML отстой, Kotlin DSL ещё нет
- Golang client for TeamCity REST API + Terraform = Terraform Provider for JetBrains TeamCity CI server

- Teamcity DSL

Administration > > > <Root project> > Sample Project (1)  Actions ▾ [Project Home](#)

Project Settings

General Settings

VCS Roots 1

Report Tabs

Parameters

Builds Schedule

Issue Trackers

Connections 1

Shared Resources

Meta-Runners

SSH Keys 1

Maven Settings

Clean-up Rules


Versioned Settings

Configure user roles in this project on the [user accounts](#) page.

Created 12 minutes ago by admin ([view history](#))

Versioned settings

Configuration

On this page you can enable synchronization of the current project settings with the version control: if the project settings are changed, the affected configuration files will be checked in to the version control; if the configuration files are changed in the version control, the changes will be applied to the project. Note that the passwords which are configured in the project and subprojects (e.g. in project's VCS roots) are stored in the configuration files and can be exposed this way. 

Supported version control systems: [Team Foundation Server](#), [Git](#), [Mercurial](#), [Subversion](#), [Perforce](#).

Using version control synchronization settings from a parent project

Synchronization is not enabled in any of the parent projects.


☐ Use settings from a parent project

☐ Synchronization disabled

☒ Synchronization enabled

Project settings

VCS root:

When build starts: 

☒ always use current settings

☐ use current settings by default

☐ use settings from VCS

☐ Show settings changes in builds

Settings format:

Apply

Portable Teamcity DSL

- settings.kts содержит конфигурацию целиком
- uuid и id для сущностей опциональны
- настройка версионирования проекта и настройки VCS не могут быть изменены из DSL

	KotlinDSL	xml(plain)
переносимость	✓	✗
гибкость	✓	✗
расширяемость	✓	✗
обратная совместимость	✓	✓/✗

- никто не любит xml

для
презентации



для прома

Teamcity in action. Интерфейс. Общий вид

□ BuildBasedOnTemplate наименования сборки

разрезы настройки

General Settings

Version Control Settings

Build Step: Command Line

Triggers

Failure Conditions

Build Features

Dependencies

Parameters 16

Agent Requirements

Suggestions

« Hide unconfigured

Last edited 4 hours ago
by Shcherbakov, Alexey
(view history)

Settings are stored in VCS (view history)

View DSL



настройка

Build configuration settings are stored in Kotlin DSL, consider changing the settings in the Kotlin scripts instead of user interface

Based on ☒ CNC-DSL / Sandbox / shcherbakov-test / Teamcity Dsl First Look / TemplateSourceBuild (detach)

Name: *

BuildBasedOnTemplate

Build configuration

ID: * ?

Cnc_Sandbox_ShcherbakovTest_TeamcityDslFirstLook_BuildBasedOnTe

This ID is used in URLs, REST API, HTTP requests to the server, and configuration settings in the TeamCity Data Directory.

Description:

Build configuration type:

Regular

(inherited)

Builds of a regular build configuration can have build steps and are executed on agents.

Build number format: * ?

%build.counter%

(inherited)

The format may include '%build.counter%' as a placeholder for the build counter value, for example, 1.%build.counter%.

It may also contain a reference to any other available parameter, for example, %build.vcs.number.VCSRootName%.

Note: The maximum length of a build number after all substitutions is 256 characters.

Build counter: * ?

5

Reset

Publish artifacts: ?

Even if build fails

(inherited)







Specify the artifacts publishing policy.

самый стандартный из всех типов шагов в Teamcity - command line

Runner type:	<div>Command Line</div> <div>Simple command execution</div>
Run:	<div>Custom script</div>
Custom script: *	<div>Enter build script content:</div> <div><div>1</div><div>echo "Это скрипт"</div></div> <div>A platform-specific script, which will be</div>

- похоже на наследование
 - шаблон - сборка, помеченная как шаблон (предок)
 - шаблонная сборка заимствует атрибутику (шаги, параметры...) шаблона (наследник)

Teamcity in action. Шаблонные сборки. UI шаги

Build Step	Parameters Description	
1. Command Line <i>(inherited, disabled)</i>	Custom script: echo template (and 1 more line) Execute: If all previous steps finished successfully	Edit 
2. Command Line <i>(inherited, disabled)</i>	Custom script: echo ChainFirstBuild => ChainFirstBuild.... (and 1 more line) Execute: If all previous steps finished successfully	Edit 
3. Command Line <i>(inherited, disabled)</i>	Custom script: echo ChainFirstBuild => ChainFirstBuild.... (and 1 more line) Execute: If all previous steps finished successfully	Edit 
4. Command Line <i>(inherited, disabled)</i>	Custom script: echo ChainFirstBuild => ChainFirstBuild.... (and 1 more line) Execute: If all previous steps finished successfully	Edit 
5. Command Line <i>(inherited, disabled)</i>	Custom script: echo ChainFirstBuild => ChainFirstBuild.... (and 1 more line) Execute: If all previous steps finished successfully	Edit 
6. Command Line <i>(inherited, disabled)</i>	Custom script: echo ChainFirstBuild => ChainFirstBuild.... (and 1 more line) Execute: If all previous steps finished successfully	Edit  Copy build step... Enable build step
7. Command Line	Custom script: echo another (and 1 more line) Execute: If all previous steps finished successfully	

```
1: object BuildBasedOnTemplate : BuildType({
2:     templates(TemplateSourceBuild) // <-- ссылка на шаблон
3:     params { // <-- собственные параметры
4:         text("param", ...)
5:         ...
6:     }
7:     steps { // <-- собственные шаги
8:         script {
9:             scriptContent = """
10:                 echo another
11:                 echo another
12:             """.trimIndent()
13:         }
14:     }
15:     // ссылки на отключённые ID настроек
16:     disableSettings("TEMPLATE_RUNNER_1")
17:     ...
18:     disableSettings("TEMPLATE_RUNNER_5")
19: })
```


- шаблонные сборки с DSL всё ещё отстой ❌
- DSL ничем не лучше в связях отнаследованного и собственного
✅❌
- с хранением кода шаблонов и связей с ним в VCS появляется шанс контроля ✅
- шаблонные сборки не нужны если вы используете KotlinDSL ✅

- подпрограмма / макрос:
 - слепок билд конфигурации
 - допускает описание входных параметров

метараннеры в интерфейсе

Build Step

Runner type:

FirstLookAtTeamcityKotlinDslMetaRunne ▼

meta runner

Note: The `FirstLookAtTeamcityKotlinDslMetaRunnerExample` meta-runner is defined in the `Teamcity Dsl First Look` project.

Step name:

Optional, specify to distinguish this build step from other steps.

```
1: .teamcity
2: --| pluginData
3: ----| _Self
4: -----| metaRunners
5: -----| matarunner.xml
```

metarunner.xml в файле settings.kts в project явно никак не регистрируется

регистрация в проекте происходит за счёт размещения xml файла
.teamcity/pluginData/_Self/metaRunners/<runner_id>.xml

можно править существующий в проекте (добавленный из UI),
но не создавать новый в проекте. (по крайней мере у меня не


```
1: object BuildWithMetarunner : BuildType({
2:     name = "Build with local metarunner"
3:     ...
4:     steps {
5:         step {
6:             type = "FirstLookAtTeamcityKotlinDslMetaRunnerExample"
7:         }
8:     }
9: })
```

метараннер код

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <meta-runner name="meta runner">
3   <description>meta runner</description>
4   <settings>
5     <parameters>
6       <param name="Password" value="credentials:SON:cd2ceb46-ba73-4ce9-87d4-94a2f4d3b9fe" spec="password display='hidden'" />
7       <param name="Username" value="cloud.mf.rable" spec="text validationMode='any' display='hidden'" />
8       <param name="cnc_name" value="CNC" spec="text display='hidden' readOnly='true' validationMode='any'" />
9       <param name="cnc_old_name" value="CNC_7" spec="text display='hidden' readOnly='true' validationMode='any'" />
10      <param name="last_version" value="7.13.0" spec="text display='hidden' validationMode='any'" />
11      <param name="repo_dev" value="Dev" spec="text display='hidden' readOnly='true' validationMode='any'" />
12      <param name="repo_qa" value="generic-qa-local" spec="text display='hidden' readOnly='true' validationMode='any'" />
13      <param name="repo_release" value="generic-release-local" spec="text display='hidden' readOnly='true' validationMode='any'" />
14      <param name="tech_stash.password" value="credentials:SON:97a4602e-989b-4e02-85e9-e21e7c611348" spec="password display='hidden' readOnly='true'" />
15      <param name="tech_stash.user" value="rnd_infl_cnc" spec="text display='hidden' readOnly='true' validationMode='any'" />
16    </parameters>
17    <build-runners>
18      <runner name="" type="simpleRunner">
19        <parameters>
20          <param name="org.jfrog.artifactory.selectedDeployableServer.downloadSpecSource" value="Job configuration" />
21          <param name="org.jfrog.artifactory.selectedDeployableServer.uploadSpecSource" value="Job configuration" />
22          <param name="org.jfrog.artifactory.selectedDeployableServer.useSpecs" value="false" />
23          <param name="script.content" value="echo test" />
24          <param name="teancity.step.mode" value="default" />
25          <param name="use.custom.script" value="true" />
26        </parameters>
27      </runner>
28    </build-runners>
29    <requirements>
30      <contains id="RQ_14097" name="system.agent.name" value="RHEI7" />
31    </requirements>
32  </settings>
33 </meta-runner>
34
35
```

Since DSL is a code in Kotlin programming language, all paradigms supported by this language are available. For instance, instead of using TeamCity templates, one can create a function or class which will encapsulate project common settings. For those who have programming skills it allows for more natural reuse of build configuration settings.

<https://www.jetbrains.com/help/teamcity/kotlin-dsl.html#KotlinDSL-HowKotlinDSLWorks>

pipeline через зависимость артефактов




```
1  project {  
2      buildType(ChainFirstBuild)  
3      buildType(ChainSecondBuild)  
4      // ...  
5  }
```

```
1: object ChainFirstBuild : BuildType({  
2:     name = "ChainFirstBuild"  
3:     ...  
4:     artifactRules = "ChainFirstBuild.txt" // <-- опубликовать артефакт  
5: })
```

pipeline через зависимость артефактов. Конфигурация второй сборки

```
1: object ChainSecondBuild : BuildType({
2:     name = "ChainSecondBuild"
3:     ...
4:     steps {
5:         script {
6:             scriptContent = """
7:                 ls -a
8:             """.trimIndent() // <-- вывести список файлов
9:         }
10:    }
11:    ...
12:    dependencies {
13:        artifacts(ChainFirstBuild) { // <-- зависимость от артефакта
14:            artifactRules = "ChainFirstBuild.txt" // <-- забрать артефакт
15:        }
16:    }
17: })
```

параметры сборки в UI

Configuration Parameters

Configuration parameters are not passed into build, can be used in references only. ?

Name	Value		
param	param	Edit	Delete
param1	slave	Edit	Delete
param10	param	Edit	Delete
param2	param	Edit	Delete
param3	param	Edit	Delete
param4	param	Edit	Delete
param5	param	Edit	Delete
param6	param	Edit	Delete
param7	param	Edit	Delete
param8	param	Edit	Delete
param9	param	Edit	Delete

```
1: object ChainFirstBuild : BuildType({
2:     name = "ChainFirstBuild"
3:     check(maxRunningBuilds == 0) {
4:         "Unexpected option value: maxRunningBuilds = $maxRunningBuilds"
5:     }
6:
7:     params {
8:         text("param", "param", label = "label"... ) // <-- параметры
9:         ...
10:    }
11:    steps {
12:        script {
13:            ...
14:        }
15:        ...
16:    })
```

```
1: .teamcity
2: [
3: --| pluginData
4: ----| _Self
5: -----| metaRunners
6: -----| matarunner.xml
7: ]
8: [
9: --| patches
10: ----| entities
11: ]
12: --| pom.xml
13: [--| Readme.md]
14: --| settings.kts
```


3 рецепта использования версионирования настроек

изменение / подход	git first	ui first	mixed (git + ui)
ui	✓	✗	✓ + патчи
code	✓	✗	✓ + патчи

1. Юра потыкал в интерфейсе
2. teamcity делает коммит от имени Юры на изменение сборки в проекте

1. Юра поправил код, сделал коммит
2. teamcity видит коммит
3. применено

1. Кристина потыкала UI
2. Юра вынужден применить патч с UI перед изменениями

```
1: changeBuildType(RelativeId("BuildBasedOnTemplate")) {  
2:     check(paused == false) {  
3:         "Unexpected paused: '$paused'"  
4:     }  
5:     paused = true  
6: }
```

UI first - никак

во всех остальных (Git first, mixed):

- код скажет всё за вас
- больше не надо бэкапить, гит поминит всё
- немного изучить Kotlin
- разработка конфигурации тоже разработка!

Как попробовать фичу представления настроек в DSL

☐ check repository branches for closed issues

General Settings

Version Control Settings 1

Build Steps 2

Triggers

Failure Conditions

Build Features

Dependencies

Parameters 17

Agent Requirements 1

Suggestions

« Hide unconfigured

Last edited 22 days ago
by Shcherbakov, Alexey
(view history)

Settings are stored in VCS
(view history)

View DSL ?

Build configuration se

Name: *

Build configuration

ID: * ?

Description:

Build configuration ty

Build number format:

Edit in UI

Repository: * ?

Publish artifacts: ?

Artifact paths: ?

General Settings

Version Control Settings ?

Build Steps ?

Triggers

Failure Conditions

Build Features

Dependencies

Parameters ?

Agent Requirements ?

Suggestions

« Hide unconfigured

Last edited 22 days ago

by Shcherbakov, Alexey

(view history)

Settings are stored in VCS (view history)

Edit in UI

Repository: * ?

Publish artifacts: ?

Artifact paths: ?

Build configuration settings are stored in Kotlin DSL

Copy to clipboard

package _self.buildTypes

import jetbrains.buildserver.configs.kotlin

import jetbrains.buildserver.configs.kotlin

import jetbrains.buildserver.configs.kotlin

object CheckRepositoryBranchesForClosedIssue

name = "check repository branches for cl

description = "npowepa senok penosensepa

maxRunningBuilds = 10

params {

param("env.TDRA_AUTH_PASSWORD", "Ma

param("env.BSTBUCKET_AUTH_PASSWORD",

param("env.BSTBUCKET_AUTH_USERID", "

text("env.CNC_UTILS_PACKAGE", "kenc/

param("env.TDRA_AUTH_USERID", "kenc/

param("env.CNC_UTILS_VCSROOT", "kenc

text("env.CNC_UTILS_CONFIG_PATH", "c

}

vcs {

root(Dnc)

}

steps {

script {

name = "env"

scriptContent = """

/bin/cat <idm> >npmrc

registry=https://npm.billing

//https://npm.billing.ru/reg

ION

"""

script {

name = "poc check repository bra

scriptContent = """poc --usercon

Newline- nr ccs

Уже сейчас достаточно зайти в любую сборку и нажать на кнопку View DSL

1. Создать свой, не пустой репозиторий
2. Создать Teamcity проект/подпроект
3. Включить версионирование проекта
4. Дождаться initial commit от teamcity в гит

1. Включены плагины maven, kotlin
2. Добавить проект в maven (ПКМ на pom.xml)
3. Выполнить install (maven)

если у вас в организации используется локальное зеркало maven,

то надо добавить исключение для репозитория dsl

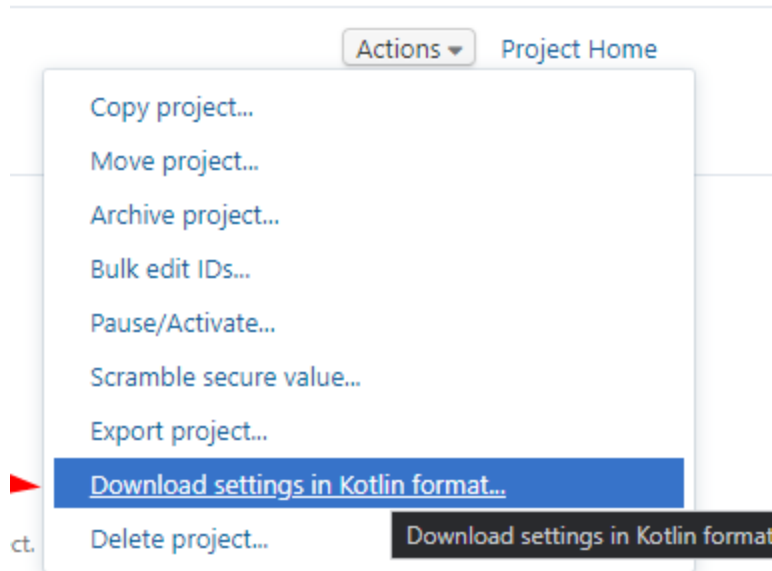
делается это путём внесения изменений в ваш конфигурационный файл для maven. Для пользователей windows путь примерно следующий:

```
C:\Users\alexey.shcherbakov\.m2\settings.xml
```

настройка (в секцию settings > mirrors)

```
1: ...
2: <settings ...>
3:   <servers>
4:   ...
5:   </servers>
6:   <mirrors>
7:     <mirror>
8:       <mirrorOf>*,!jetbrains-all,!teamcity-server</mirrorOf>
9:       <name>maven</name>
10:      <url>https://${TEAMCITY_SERVER_URL}:${TEAMCITY_SERVER_PORT}/artifactory/maven</url>
11:      <id>maven</id>
12:    </mirror>
13:    ...
14:  </mirrors>
15:  ...
16: </settings>
```

- заходим в проект



- поместить содержимое скаченного архива в папку .teamcity репозитория

на всемя перехода чтобы не заафффектить существующие сборки
рекомендуется выглючить триггеры, билд фичи

создать проект teamcity

Create Project

From a repository URL

 Manually

Parent project: *

CNC / Sandbox / shcherbakov-test



Repository URL: *

ssh://git@stash.billing.ru:2222/~alexey.shcherbakov/teamcity-dsl-first-look.git



A VCS repository URL. Supported formats: http(s)://, svn://, git://, etc. as well as URLs in Maven format.

Username:

Provide username if access to repository requires authentication.

Password:



Provide password if access to repository requires authentication.

Proceed

создать проект на основе версионизируемых настроек

Administration /  <Root project> /  CNC /  Sandbox /  shcherbakov-test

Create Project From URL

✓ The connection to the VCS repository has been verified

⚠ The VCS repository contains **.teamcity/settings.kts** file with settings of some project. Would you like to import these settings?

☒ Import settings from **.teamcity/settings.kts**

☐ Do not import settings, create project from scratch

Project name: *

Teamcity Dsl First Look

VCS root:

(Git) ssh://git@stash.billing.ru:2222/~alexey.shcherbakov/teamcity-dsl-first-look.git

Proceed

Cancel

Teamcity увидит ваш файл settings.kts

Если вы создаёте подпроект в версионизируемом проекте (как в моём случае), то будет необходимо "отчудить" новый подпроект от основного

после создания проекта заходим в подпроект

- там будет стоять Use settings from a parent project
- выбрать Synchronization enabled
- далее импортируем настройки

импорт может занять продолжительное время в случае
большого проекта

1. Знакомство с конфигурацией
2. Откатов состояния конфигурации всего проекта до последнего рабочего
3. Анализ существующих сборок
4. Перенос сборки из одного проекта в другой методом копирования
5. Аудит

- Цикл статей по построению сборок с использованием Teamcity DSL
- Документация Kotlin DSL
- swampUP 2019 | Learn How to Use Kotlin DSL for Configuring the Builds in TeamCity
- Рецепты TeamCity. Доклад Яндекс.Такси