Buggy

Выпуск latest

июл. 13, 2019

Общая

Buggy - это надстройка над **TestNG** для быстрого создания/подключения тестового проекта, разработки и вариативнго запуска автотестов.

Основная документация содержит следующие разделы:

Глава 1

Описание

1.1 Позволяет

1. Легко и просто подключить TestNG к проекту с тестами. Как пример, смотреть модуль buggy-min-example.
2. Собирать готовый к исполнению jar.
3. Обрабатывать параметры запуска и расширять уже существующую конфигурацию (JCommander).
4. Автоматически собирать исполняемые тестовые классы в сьюты для последующего запуска в TestNG.
5. Автоматически собирать и добавлять в TestNG необходимые листенеры.
6. «На горячую» перезагружать настройки логирования.
7. Регулировать запуск тестов по типу.
8. Регулировать запуск тесов по компонентам, сервисам или интерфейсам тестируемой системы.
9. Транслировать результаты запуска в различные сервисы.

1.2 Реализовано

1. Управление конфигурацией запускаемых тестов.
2. Расширение существующей конфигурации через собственные интерфейсы.
3. Листенер для IntelliJ IDEA TestNG плагина для запуска тестов из IntelliJ IDEA
4. Атомарное логирование для каждого тестового или конфигурационного метода в отдельный файл.
5. Цветовая дифференциация консольных логов.

6. Механизм мониторинга исполняемых тестов.
7. Базовый Telegram-нотификатор.
8. Модуль к feign клиенту (утилиты).
9. Модуль к okhttp клиенту (утилиты).

1.3 Предстоит

1. Модуль интеграция с TestRail (трансляции результатов).
2. Модуль интеграция с ReportPortal (трансляции результатов).
3. Модуль к Retrofit клиенту (утилиты).
4. Модуль работы с protobuf.

Глава 2

Конфигурация

2.1 PrimaryConfig

- Базовая конфигурация представлена индексируемым интерфейсом `PrimaryConfig`.
- Класс реализации интерфейса `PrimaryConfig` будет автоматически подгружен, создан экземпляр и использован для конфигурации по умолчанию (получить: `Buggy.getPrimaryConfig()`).
- В случае наличия нескольких классов реализации интерфейса `PrimaryConfig`, класс требуемой конфигурации задаётся явно - `Buggy.setPrimaryConfigClass()` (в случае многомодульностьи тестового проекта).
- Класс реализации интерфейса `PrimaryConfig` может быть имплементирован от множества конфигурационных интерфейсов, в том числе и из разных проектов.
- Значения парамтеров „по умолчанию“ переназначаются в классе реализации. Пример `org.touchbit.buggy.example.min.config.Config`:

```
public class Config implements PrimaryConfig {

    public Config() {
        setPrintLogFile(true);
    }
}
```

2.2 SecondaryConfig

- Дополнительная конфигурация (команды) представлена индексируемым интерфейсом `SecondaryConfig`.
- Любой класс реализации интерфейса `SecondaryConfig` будет автоматически подгружен и создан экземпляр (получить список: `Buggy.getSecondaryConfigs()`).

- Класс реализации интерфейса `SecondaryConfig` может быть имплементирован от множества конфигурационных интерфейсов, в том числе и из разных проектов.
- Для класса реализации интерфейса `SecondaryConfig` обязательно наличие аннотации `com.beust.jcommander.Parameters` с объявленным `commandNames`.
- Пример: `org.touchbit.buggy.example.min.config.MinExampleSecondaryConfig`

2.3 Параметры запуска

Таблица 1: Параметры запуска

Keys		Default	Description
–help	-?	false	Вывести информацию с параметрами запуска.
–all		false	При запуске тестов вывести в лог все параметры конфигурации и их значения.
–check		false	Проверить конфигурацию на корректность без запуска тестов.
–version	-v	false	Вывести версию исполяемого jar.
–force	-f	false	Запуск всех тестов без исключения.
–print-suite		false	Вывести информацию о тестовом сьюте.
–print-cause		false	Вывести причину падения/исключения теста.
–print-log		false	Вывести в лог путь к файлу выполненного теста.
–log		logs	Относительный или абсолютный путь к директории ведения логов.
–status		null	Статус с которым следует принудительно завешить прогон тестов.
–threads		50	Количество потоков для исполняемых тестовых методов.
–services	-s	Runtime	Список доступных для тестирования сервисов.
–interface	-i	Runtime	Список доступных для тестирования интерфейсов.
–type	-t	INTEGRATION	Тип проводимого тестирования.
–artifacts-url		null	Url к логам тестов (CI)

2.4 Примеры

2.4.1 Вывод параметров запуска

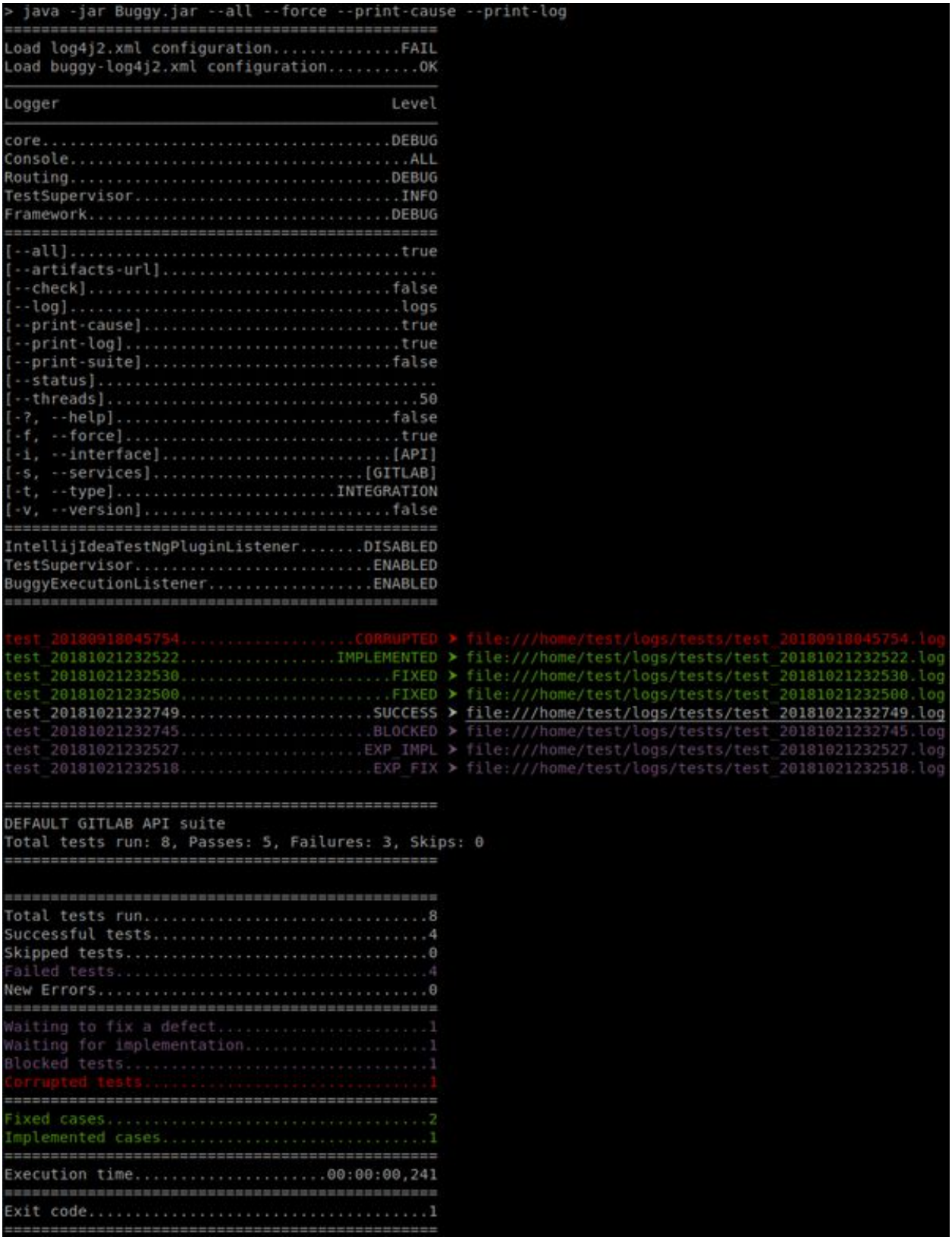
```
$ java -jar buggy-min-example/target/Buggy.jar -?

================================================
Usage: Buggy [options] [command] [command options]
  Options:
    --artifacts-url
      The storage address for the builds (artifacts).
    --check
      Check buggy configuration without test run.
    -f, --force
      Running all tests, including those that fall.
    -?, --help
      Print usage.
```

(continues on next page)

(продолжение с предыдущей страницы)

```
  -i, --interface
    List of tested interfaces in the format: NAME,NAME,NAME.
    Default: [API]
  --print-cause
    Print the cause of a fail or skip test in the console log.
  --print-log
    Print the test log file path in the console log
  --print-suite
    Display information on the Suite in the console log.
  -s, --services
    List of tested services in the format: NAME,NAME,NAME.
    Default: [GITLAB]
  --threads
    The number of threads to run the test methods.
    Default: 50
  -t, --type
    Type of tests to run.
    Default: INTEGRATION
    Possible Values: [SMOKE, MODULE, INTEGRATION, SYSTEM]
  -v, --version
    Print program version
Commands:
  network
    Usage: network [options]
      Options:
        --connection-timeout
          Connection timeout for request
          Default: 10
        --host
          Tested host
          Default: http://example.com
        --read-timeout
          Read timeout for response
          Default: 10
        --write-timeout
          Write timeout for request
          Default: 10
```

2.4.2 Запуск тестов с флагами



Глава 3

Подключение

3.1 Maven зависимости

Для подключения проекта необходимо в первую очередь добавить в `pom.xml` репозиторий

```
<repositories>
    <repository>
        <id>touchbit</id>
        <url>https://touchbit.org/repository/</url>
    </repository>
</repositories>
```

Далее есть 2 пути развития событий, простой и правильный.

3.1.1 Простое подключение проекта

Для уменьшения головной боли при создании нового проекта с тестами, рекомендуется унаследоваться от корневого `pom.xml` Buggy.

Для этого необходимо добавить в `pom.xml` на уровне `<project>` родителя:

```
<parent>
    <groupId>org.touchbit.buggy</groupId>
    <artifactId>buggy</artifactId>
    <version>?.?.?</version>
</parent>
```

Это нам даёт следующие приемущества:

1. Версия gdfgdggdfgdggdfgdg

3.2 Конфигурирование проекта

Глава 4

Зависимости

Глава 5

buggy-core

Глава 6

buggy-min-example

Глава 7

buggy-testrail

7.1 TestRail

If you plan to use one TestRail Run ID for autotests, then you need to make changes to the TestRail database.

1. Connect to DB (for example mysql)

```
mysql -u 'testrail@192.168.1.1' -pPASSWORD testrail
```

2. Remove restrictions from statuses

```
UPDATE statuses SET is_final=0
```

7.2 Using

Совет: More clearly the module connecting mechanism is shown in the *buggy-min-example* module.

7.2.1 Configuring pom.xml

If buggy is not used as a parent project

```
<dependency>
    <groupId>org.touchbit.buggy</groupId>
    <artifactId>buggy-testrail</artifactId>
    <version>0.3.0</version>
    <scope>compile</scope>
</dependency>
```

If buggy is used as parent project

```
<dependency>
    <groupId>org.touchbit.buggy</groupId>
    <artifactId>buggy-testrail</artifactId>
</dependency>
```

7.2.2 Configuring project

StatusMapper

Совет: If you do not use custom statuses, then this item can be skipped.

To manage statuses and transfer custom statuses, you need to create a class implementing the *StatusMapper<>* interface and define your own mapping for the statuses used. This is necessary in order for match statuses used in Buggy with TestRail statuses. For example:

```
import org.touchbit.buggy.core.model.Status;
import org.touchbit.buggy.testrail.StatusMapper;
import org.touchbit.testrail4j.core.type.Statuses;

public class StatusMap implements StatusMapper<Status> {

    @Override
    public long getId(Status status) {
        switch (status) {
            case SUCCESS:       return Statuses.PASSED.getId();
            case BLOCKED:       return Statuses.BLOCKED.getId();
            case UNTESTED:      return Statuses.UNTESTED.getId();
            case FAILED:        return Statuses.FAILED.getId();
            case FIXED:         return Statuses.CUSTOM_STATUS1.getId();
            case IMPLEMENTED:   return Statuses.CUSTOM_STATUS2.getId();
            case CORRUPTED:     return Statuses.CUSTOM_STATUS3.getId();
            case EXP_FIX:       return Statuses.CUSTOM_STATUS4.getId();
            case SKIP:          return Statuses.CUSTOM_STATUS5.getId();
            case EXP_IMPL:      return Statuses.CUSTOM_STATUS6.getId();
            default:
                throw new RuntimeException("Unhandled status received: " + status);
        }
    }
}
```

DefaultTestRailListener

Create a listener inherited from the class *DefaultTestRailListener* and, if necessary, pass your specific StatusMapper to the super class. For example:

```
import org.touchbit.buggy.testrail.listeners.DefaultTestRailListener;

public class TestRailListener extends DefaultTestRailListener {

    public TestRailListener() {
        super(new StatusMap());
    }
```

(continues on next page)

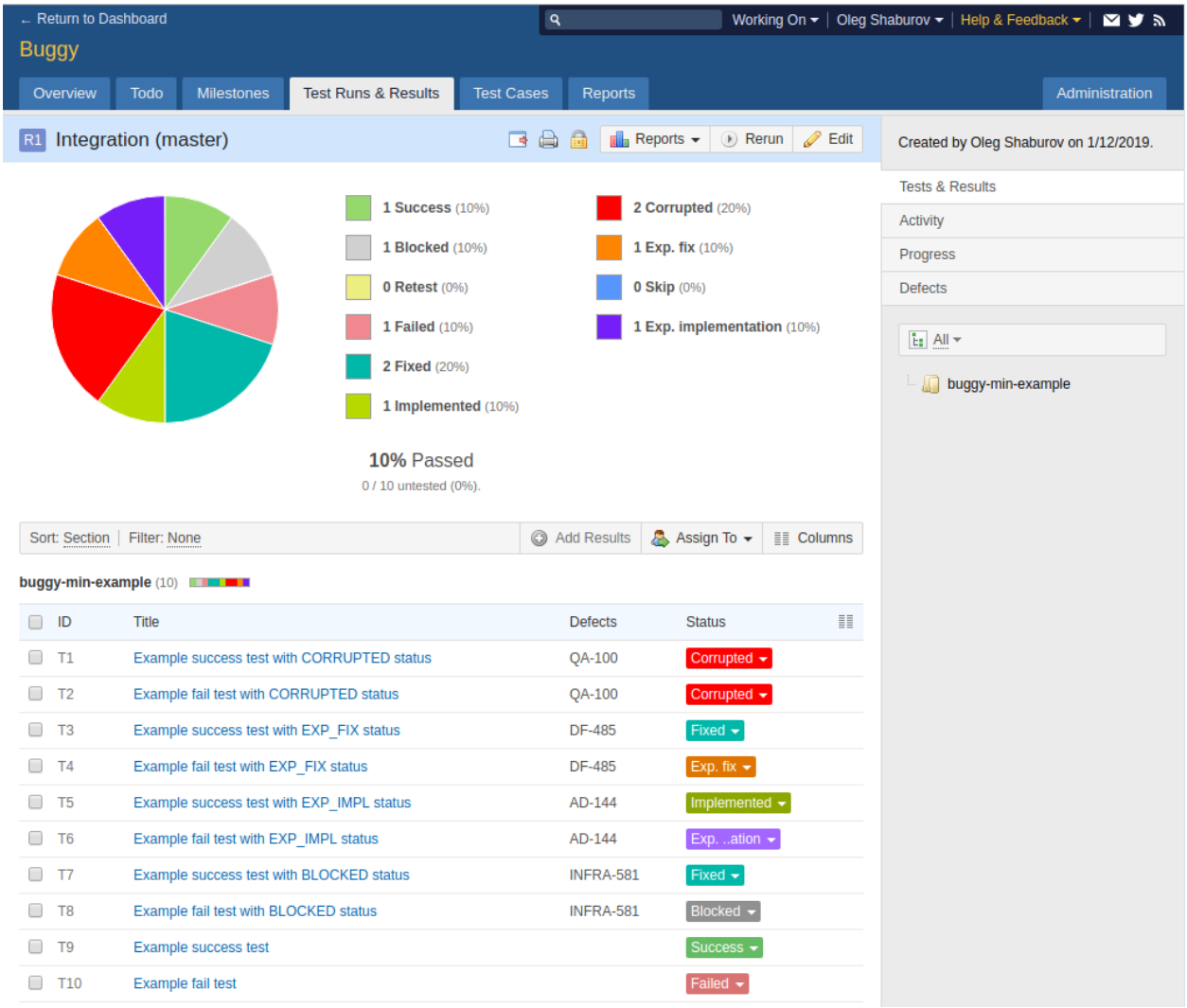
(продолжение с предыдущей страницы)

```
}
```

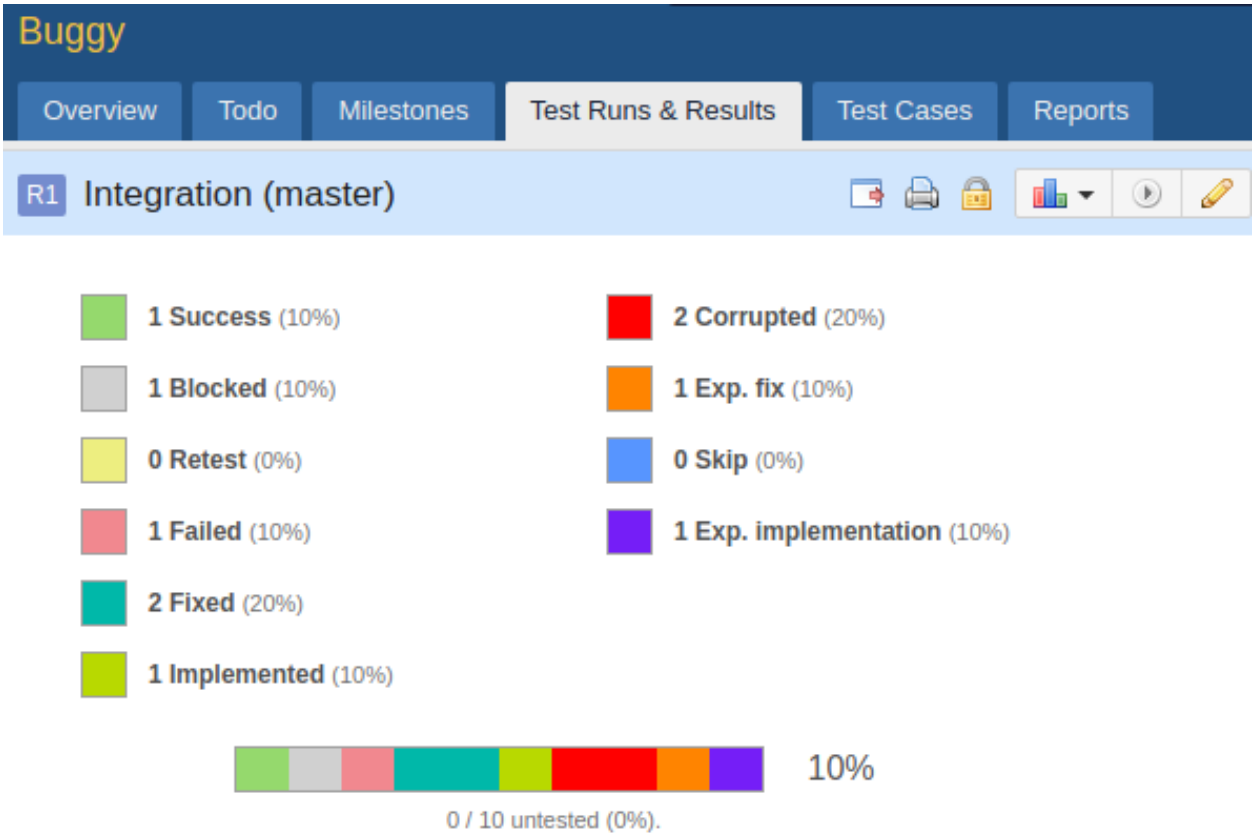
Or develop your own listener like *DefaultTestRailListener* with your own implementation of the *afterInvocation (IInvokedMethod, ITestResult)* method

7.3 Examples

7.3.1 Overview



7.3.2 Custom StatusMapper



Sort: Section | Filter: None

buggy-min-example (10)

	ID	Title	Defects	Status
☐	T1	Example success test with CORRUPTED status	QA-100	Corrupted
☐	T2	Example fail test with CORRUPTED status	QA-100	Corrupted
☐	T3	Example success test with EXP_FIX status	DF-485	Fixed
☐	T4	Example fail test with EXP_FIX status	DF-485	Exp. fix
☐	T5	Example success test with EXP_IMPL status	AD-144	Implemented
☐	T6	Example fail test with EXP_IMPL status	AD-144	Exp. ..ation
☐	T7	Example success test with BLOCKED status	INFRA-581	Fixed
☐	T8	Example fail test with BLOCKED status	INFRA-581	Blocked
☐	T9	Example success test		Success
☐	T10	Example fail test		Failed

7.3.3 Default StatusMapper