**Gherkin**

(здесь и далее - перевод и легкая адаптация статьи [“Writing Features - Gherkin Language”][Writing Features])

**Gherkin** - человеко-читаемый язык для описания поведения системы, который использует отступы для задания структуры документа, (пробелы или символы табуляции). Каждая строчка начинается с одного из ключевых слов и описывает один из шагов.

Пример:

Функция: Короткое, но исчерпывающее описание требуемого функционала

Для того, чтобы достичь определенных целей

В качестве определенного участника взаимодействия с системой

Я хочу получить определенную пользу

Сценарий: Какая-то определенная бизнес-ситуация

Дано какое-то условие

И ещё одно условие

Когда предпринимается какое-то действие участником

И им делается ещё что-то

И вдобавок он совершил что-то ещё

То получается какой-то проверяемый результат

И что-то ещё случается, что мы можем проверить

Обработчик разбивает файл с тестами на функции, сценарии и входящие в них шаги. Давайте разберем этот пример:

1. Строка **Функция(Feature) : Короткое, но исчерпывающее описание требуемого функционала** начинает собой описание функционала и дает ему название. Следующие три строчки не обрабатываются и не несут никакой смысловой нагрузке для обработчика тестов, но они задают контекст тестирования и одновременно описывают, какую пользу мы получим от этого функционала.
2. Строка **Сценарий(Scenario): Какая-то определенная бизнес-ситуация** начинает сценарий и содержит его описание.
3. Следующие 7 строчек описывают шаги теста, каждому из которых впоследствии будет сопоставлен определенный программный код, выполняющий описанное действие. Сопоставлению подлежат части строк лежащие после ключевых слов “Дано”, “И”, “Когда” и т.д.

**Функции (Feature)**

Каждая функция описывается в отдельном файле с расширением *.feature*. Первая строчка должна начинаться с ключевого слова “Функция:”, за которой могут идти три строчки с описанием, размеченные отступами. Каждая функция обычно состоит из списка сценариев.

Каждый сценарий состоит из списка *шагов*, каждый из которых должен начинаться с одного из ключевых слов:

* Дано **(Givens)**
* Когда **(Whens)**
* То **(Thens)**
* Но **(But)**
* И **(And)**

Шаги *“Но”* и *“И”* существуют исключительно для удобства чтения и по своим функциям повторяют ключевое слово, с которого начиналась предыдущая строчка.

Вдобавок к сценариям, описание функционала может также содержать *структуры сценариев* и *предыстории*.

**Сценарий (Scenario)**

Сценарий представляет собой одну из ключевых структур в языке *Gherkin*. Каждый сценарий начинается с ключевого слова **“Сценарий(Scenario):”**, и может содержать в себе название сценария. Описание функционала может содержать в себе один или больше сценариев, и каждый сценарий состоит из одного или более шага.

Каждый из следующих сценариев содержит три шага:

Сценарий: Вася создает новую запись

Дано я вошел в систему как Вася

Когда я пытаюсь добавить запись в справочник "Лекарства"

То мне должен быть ответ "Ваша запись успешно добавлена."

Сценарий: Вася не может добавлять запись в справочник лечений

Дано я вошел в систему как Вася

Когда я пытаюсь добавить запись в справочник "Виды лечений"

То мне должен быть ответ "У вас нет прав доступа!"

**Структура сценария (Scenario Outline)**

Достаточно часто приходится писать множество мелких сценариев, которые различаются буквально парой переменных. Эти повторения могут быстро надоесть:

Сценарий: удалить 5 записей из 12

Дано есть 12 записей

Когда я удаляю 5 записей

То у меня должно остаться 7 записей

Сценарий: удалить 5 записей из 20

Дано есть 20 записей

Когда я удаляю 5 записей

То у меня должно остаться 15 записей

***Структуры сценариев(Scenario Outline)*** позволяют нам более кратко описывать подобные наборы сценариев с помощью шаблонов:

Структура сценария: удаление записей

Дано есть <было> записей

Когда я удаляю <удалено> записей

То у меня должно остаться <остаток> записей

Примеры:

| было | удалено | остаток |

| 12 | 5 | 7 |

| 20 | 5 | 15 |

**или**

Фича(Feature): Сравнение данных в csv-файлах

Схема сценария(Scenario Outline): Сравнение двух CSV-файлов

Даны два файла с входными данными "<file1>" и "<file2>".

Когда мы сравниваем эти два файла

Содержимое первого файла должно "<matchOrNot>" содержимому второго.

Примеры:

| file1 | file2 | matchOrNot |

| input1.csv | input2.csv | match |

| input3.csv | input4.csv | not match |

Шаги указанные в структуре сценария не выполняются напрямую, но используются для подстановки в них значений из таблицы примеров. Каждая строчка таблицы будет обрабатываться как отдельный сценарий с указанными значениями вместо заглушек “было”, “удалено” и “стало”.

**Предыстории (Background)**

Предыстории позволяют вам добавить определенный контекст ко всем сценариям в пределах функции. По сути, предыстория - сценарий без имени, состоящий из шагов. Основное отличие в запуске: предыстория запускается перед каждым сценарием:

Функция: поддержка многих справочников

Предыстория:

Дано есть пользователь с именем "Вася"

И есть справочник "Лекарства"

И у пользователя "Вася" есть право на запись в "Лекарство"

И есть справочник "Виды лечений"

Сценарий: Вася создает новую запись

Дано я вошел в систему как Вася

Когда я пытаюсь добавить запись в справочник "Лекарства"

То мне должен быть ответ "Ваша запись успешно добавлена."

Сценарий: Вася не может добавлять запись в справочник лечений

Дано я вошел в систему как Вася

Когда я пытаюсь добавить запись в справочник "Виды лечений"

То мне должен быть ответ "У вас нет прав доступа!"

**Шаги**

Функции состоят из шагов, также известных как *Данные*, *Действия* и *Результаты*.

**Данные (Givens)**

Назначение шагов *Дано* состоит в **приведение системы в известное состояние** перед тем как пользователь (или внешняя система) начнет взаимодействие с системой (в шагах *Когда*). Также можно рассматривать их как предусловия.

Пример: создавать объекты сущностей или настраивать БД

Дано нет пользователей в базе

Дано база данных пустая

Пример: вход пользователя в систему (исключение к правилу “никаких взаимодействий в шаге Дано”)

Дано я вошел в систему как "Вася"

**Действия (Whens)**

Назначение шагов *Когда* состоит в **описании ключевого действия, совершаемого пользователем**.

Пример: взаимодействие со страницей

Когда я открыл форму добавления учреждения

Когда я ввел "Институт радости" в поле "Наименование"

Когда я выбрал в поле "Тип" значение "Институт"

Когда я нажал на кнопку "Сохранить"

**Результаты (Thens)**

Назначение шагов *То* состоит в **наблюдении результатов выполнения действий**. Наблюдения должны быть связаны с явной пользой, которая указаны в описании функции. Также необходимо помнить, что должен проверяться *вывод системы* (отчеты, интерфейс, сообщения), а не что-то глубоко закопанное в систему.

**Предлоги (And, But)**

Если у вас есть несколько шагов *Дано*, *Когда*, или *То* то вы можете писать так:

Сценарий: множественные данные

Дано что-то первое

Дано что-то второе

Дано и что-то ещё

Когда я открою свои глаза

То я увижу что-то

То чего-то я не увижу

...или можете использовать шаги *И* и *Но*, превращая свой сценарий в нечто более читаемое:

Сценарий: множественные данные

Дано что-то первое

И что-то второе

И и что-то ещё

Когда я открою свои глаза

То я увижу что-то

Но чего-то я не увижу

**Таблицы**

Регулярные выражения, с помощью которых программисты получают данные из текстового описания шагов позволяют получать небольшие куски данных из самой строчки. Но бывает и такое, что необходимо прловести тесты с разными наборами данных. И здесь нам на помощь придут таблицы:

Сценарий:

Дано существуют следующие пользователи:

| Логин | E-mail | Пароль |

| user1 | user1@mail.ru | pass1 |

| joe | joe@gmail.com | hey |

| heyho | hey@hoe.com | joe |