Cheatography

Gherkin++ Cheat Sheet

by ZH (openkeyword) via cheatography.com/68516/cs/17281/

Categories

(I)	**Input category** - Input categories can be preconditions or input variables.
(O)	**Output category** - Expected result data of the test case.
(A)	**Action category** - An action is a simple element of the test process.
(F)	**Feature call**

Choices

(S)	**Single** - Select the choice only once.
(D)	**Default** - Is the default value of the choices.

Example:

Categorie
paying method (O): card**(D)**; on site; money transfer
price (O): 0; 100; 1000

Constraint
WHEN …. **THEN** price **IS** 100

The generated test will be:
… paying method=card, price=100

Constraints

Constraint name/test name	each constraint may have a name, which is inherited to the generated test
GIVEN	describes preconditions, and can be omitted In Multilayer structures the *outputs are suppressed*
WHEN	contains the inputs and obligatory
THEN	contains the output and obligatory
AND	connects two **GIVEN**/**WHEN**/**THEN** expressions
IS/**ARE**	connects a category and a choice of this category, such as MyCat **IS** MyChoice.

Note 1: The basic difference between **GIVEN** and **WHEN** is that if a **GIVEN** contains a existing test (see Multilayer structure syntax elements), then *outputs are suppressed*, only input will remain.

Note 2: The original Gherkin syntax requires the **GIVEN** - **WHEN** - **THEN** series.
However, state transition testing requires **WHEN** - **THEN** - **WHEN** - **THEN** - **WHEN** - **THEN** series as well. 4Test permits this syntax.

Table

Extraction

PRECONDITION - make models simpler. In lots of the cases when we would like to end-toend test a feature, we have to reach the feature to be tested. This requires to set some preconditions, i.e. the necessary input values. It’s not reasonable to include these preconditions to each constraint, and it’s reasonable to separate the preconditions and the tests for the feature. **PRECONDITION** does this.

AC: Acceptance Criterion

SUB *[AC-Name]*:	SUB-keyword defines an AC as SUB call.
[AC-Name] **:**	Acceptance Criterion

Use Case Step Counter

@1 **@2**	First arder Step
@1a **@1b**	Second order Step
@1 **@1a** **@1b**	

OKW

Input-Caregory

```
CN (I): CHOICE -> SetValue( CN, CHOICE )
```

Action-Category

```
button (A): CHOICE -> ClickOn( CHOICE )
Button (A): CHOICE -> ClickOn( CHOICE )
CN (A): clicked -> ClickOn( CN )
```

Output-Caregory

```
CN (O): CHOICE -> VerifyValue( CN, CHOICE )
CN (O): #caption CHOICE -> VerifyCaption( CN, CHOICE )
CN (O): #label CHOICE -> VerifyLabel( CN, CHOICE )
CN (O): #placeholder CHOICE -> VerifyPlaceholder( CN, CHOICE )
CN (O): #tooltip CHOICE -> VerifyTooltip( CN, CHOICE )
CN (O): #exist yes -> VerifyExists( CN, 'YES' )
CN (O): @exist no -> VerifyExists( CN, 'NO' )
CN (O): @exist yes -> VerifyIsActive( CN, 'YES' )
CN (O): @exist no -> VerifyIsActive( CN, 'NO' )
CN (O): @exist yes -> VerifyHasFocus( CN, 'YES' )
CN (O): @exist no -> VerifyHasFocus( CN, 'NO' )
```

Note:
CN: **C**ategorie **N**ame

WHEN original price **IS** 20 **AND** reduction is 10 **THEN** total price IS 19

WHEN original price **IS** 100 **AND** reduction is 12 THEN total price IS 88

The same in one constraint:

WHEN original price IS 20 | 100 **AND**
reduction **IS** 10 | 12
THEN total price **IS** 19 | 88

Tables reducing the number of constraint if they differ in only the choices. If you are a test analyst, you will understand this constraint and your model will be shorter. On the other hand, the generated test cases will remain understandable for everybody.

```
Boolean VerifyExists();
Boolean VerifyIsActive();
Boolean VerifyHasFocus();
```