

Documentation

Table of contents

Home

Intro

- What is model based testing
- Introduce to MBT Bundle
- Questions and Answers

Beginner

- Start Examples Project
- Getting Started
- Task and Bug
- Tips and Tricks
- Bulk Actions

Intermediate

- Init Project
- Model
- Subject
- Debug
- Weight and Probability
- Workflow and State Machine
- Default Users
- Generator
- Path Reducer
- Test Model
- Report Bug

Advanced

- Deploy
- CI Integration
- Unit Testing

Home

MBT Bundle Documentation

- [Intro](#)
- [Beginner](#)
- [Intermediate](#)
- [Advanced](#)

Intro

[Home](#) / Intro

Take a quick look before starting

- [What is model based testing](#)
- [Introduce to MBT Bundle](#)
- [Questions and Answers](#)

What is model based testing

[Home](#) / [Intro](#) / What-Is-MBT

What is Model Based Testing?

1. In traditional automation testing, we write test case. We must think about how many test cases are enough to cover the feature we want to test.
2. In model based testing, we describe the feature as models. The tool we use will generate and execute them for us.
3. By using model based testing, we can focus on business logic (by defining models), save time from writing and executing test cases. The tool will take care of generating test cases, depend on what we need: regression test (quick) or exploration test (slow)

For more information about model based testing, I suggest to take a look at these 2 slides:

1. [Model-Based Testing](#)
2. [Graph Theory Techniques in Model-Based Testing](#)

Introduce to MBT Bundle

[Home](#) / [Intro](#) / Introduce-to-MBTBundle

Introduce to MBTBundle

1. What

1. It's set of tools to help testers test a software
2. It use Model-Based Testing technique
3. It's built on top of: Symfony, Api Platform

2. Why

1. I know about Model-Based Testing idea when I was an intern
2. I try to find tools to use but they didn't fit me
3. When I have enough development skills, I start develop this tool

3. How

1. It use **generator** to generate test case
2. When a bug is found, it use **reducer** to reduce the reproduce steps
3. When the bug is reduced, it **report** the bug to issues tracking system

4. Who

1. Tester with programing skill, PHP at least
2. Small team want to focus on business logic at beginning, but want to test all features at the end of development

5. When

1. The first commit was on 14 Oct 2017
2. The tool reach version 1.0 in Feb 2019
3. The tool reach version 1.2 in Mar 2019 (ready to test)
4. The tool reach version 1.6 in Apr 2019 (ready for production use)

6. Where

1. It can be used to test any platform (browser, api, mobile, desktop)

Questions and Answers

[Home](#) / [Intro](#) / Q-A

Questions And Answers

1. Q: Do I need programming skill to use it? A: Yes (PHP)
2. Q: Do it have drag & drop feature? A: No (maybe yes in the future, but limited to creating models only)
3. Q: How do I create my models? A: You have to describe the feature you want to test in YAML files (a user-friendly text format)
4. Q: How do I test my models? A: You have to create tasks (and give enough information to the task so that it can know how to execute them e.g. generator, reducer, model)
5. Q: How do I report the bug? A: The tool will automatically report the bug to configured bug tracking system (this tool is not a bug tracking system)
6. Q: Great, I reported the bug to developer, and he said it's fixed, how do I test it? A: You have to do 2 things:
 1. Test the bug again by creating a new replay task (choose the bug that has been fixed)
 2. Create bulk tasks to do exploration test (choose random generator, choose models may related to the bug that has been fixed)
7. Q: I have a project, with a lot of features, with many complex logic to test, where can I begin? A: Try to isolate into small logic, each logic can be describe in multiple models. Don't worry about performance, this tool is built with scalable in mind
8. Q: What if I reported the bug, but developer said it's not a bug? A: Try these:
 1. You are programming, and you are developer. And developer make mistake. Make sure your code work okay
 2. This tool does not know the requirements as well as your team does, make sure to check with your team whether it's a bug, or a feature. If it's a feature, make sure your code is updated
 3. Try to replicate the bug by yourself

Beginner

[Home](#) / Beginner

For people who want to test existing models

- [Start Examples Project](#)
- [Getting Started](#)
- [Task and Bug](#)
- [Tips and Tricks](#)
- [Bulk Actions](#)

Start Examples Project

[Home](#) / [Beginner](#) / Start-Examples-Project

Install 4 dependencies:

- [git](#)
- [docker](#)
- [docker-compose](#)
- [bash for windows 10](#)

Then run these commands:

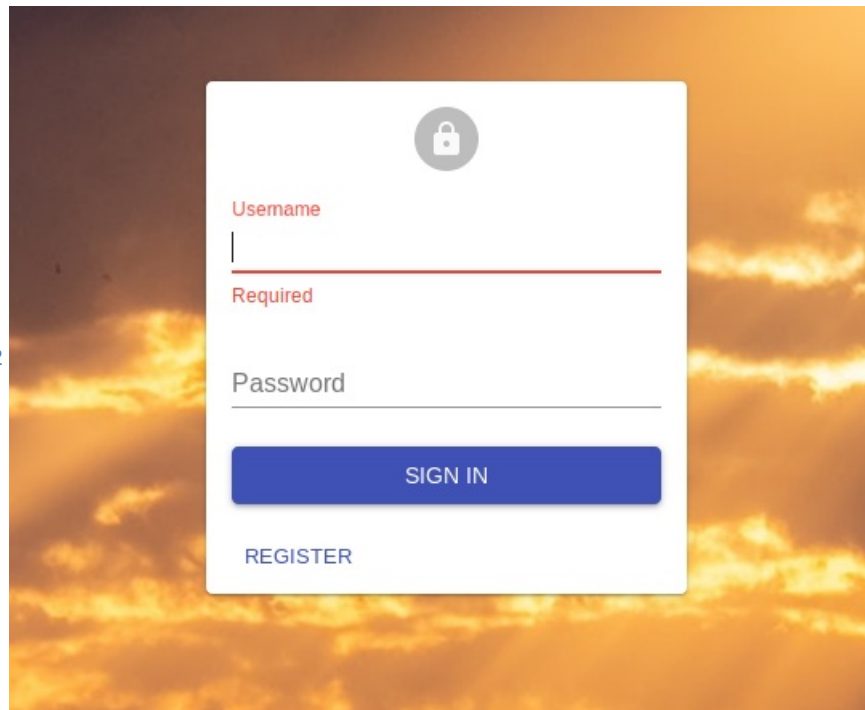
```
$ git clone git@github.com:tienvx/mbt-examples.git
$ cd mbt-examples
$ docker-compose up --scale worker=4
$ # Then open terminal, run these commands once:
$ bash install.sh
```

Getting Started

[Home](#) / [Beginner](#) / Getting-Started

Follow these steps to become familiar with Model Based Testing Bundle:

1. Go to <http://localhost:82>



2. Login with:
 1. admin/admin
 2. user/user
 3. register new user
3. Go to <http://localhost:82/#/tasks>
4. We can:

Title *
Task 1

Models *
Checkout

Generator *
Random

Meta Data

1. [Create a task](#)

Max Path Length
300

Transition Coverage
100

Place Coverage
100

Reducer *
Loop

☒ Take Screenshots

☐ Report Bug

 **SAVE**

Title *
Task 1

Models *
Checkout **Product Page** **Shopping Cart**

Generator *
Random

Meta Data

2. [Create bulk tasks](#)

Max Path Length
300

Transition Coverage
100

Place Coverage
100

Reducer *
Loop

☒ Take Screenshots

 **SAVE**

Title *

Task 1

Generator

Replay

Bug *

The first bug

The last bug

New bug found

Reducer *

Loop


Take Screenshots









































SAVE

3. Create bulk replays

5. Go to <http://localhost:82/#/tasks>







6. Wait for the task to be completed

+ CREATE + BULK CREATE + BULK REPLAY  EXPORT




<input type="checkbox"/>	Id	Bugs	Title	Model	Generator	Reducer	Progress	Status	Created at	Updated at	Take screenshots	Meta data	
<input type="checkbox"/>	/api/tasks/1	/api/bugs/1	Multi Tasks #1 (1)	/api/models/product	random	loop	0	completed	3/18/2019	3/18/2019	✓	["maxPathLength":300,"transitionCoverage":100,"placeCoverage":100]	 SHOW  EDIT
<input type="checkbox"/>	/api/tasks/2	/api/bugs/2	Multi Tasks #1 (2)	/api/models/shopping_cart	random	loop	0	completed	3/18/2019	3/18/2019	✓	["maxPathLength":300,"transitionCoverage":100,"placeCoverage":100]	 SHOW  EDIT
<input type="checkbox"/>	/api/tasks/3	/api/bugs/3	Task 1	/api/models/checkout	random	loop	0	completed	3/18/2019	3/18/2019	✓	["maxPathLength":50,"transitionCoverage":100,"placeCoverage":100]	 SHOW  EDIT
<input type="checkbox"/>	/api/tasks/4	/api/bugs/4	Task 2	/api/models/checkout	random	loop	0	completed	3/18/2019	3/18/2019	✓	["maxPathLength":300,"transitionCoverage":100,"placeCoverage":100]	 SHOW  EDIT
<input type="checkbox"/>	/api/tasks/5		Task 3	/api/models/checkout	random	loop	0	completed	3/18/2019	3/18/2019	✓	["maxPathLength":300,"transitionCoverage":100,"placeCoverage":100]	 SHOW  EDIT
<input type="checkbox"/>	/api/tasks/6		Task 4	/api/models/checkout	random	loop	0	completed	3/19/2019	3/19/2019	✓	["maxPathLength":25,"transitionCoverage":100,"placeCoverage":100]	 SHOW  EDIT
<input type="checkbox"/>	/api/tasks/7	/api/bugs/5	Task 5	/api/models/checkout	random	loop	0	completed	3/19/2019	3/19/2019	✓	["maxPathLength":50,"transitionCoverage":100,"placeCoverage":100]	 SHOW  EDIT
<input type="checkbox"/>	/api/tasks/8		Task 6	/api/models/checkout	random	loop	0	completed	3/19/2019	3/19/2019	✓	["maxPathLength":25,"transitionCoverage":100,"placeCoverage":100]	 SHOW  EDIT
<input type="checkbox"/>	/api/tasks/9	/api/bugs/7	Multi Tasks #2 (1)	/api/models/product	random	loop	0	completed	3/19/2019	3/19/2019	✓	["maxPathLength":100,"transitionCoverage":100,"placeCoverage":100]	 SHOW  EDIT
<input type="checkbox"/>	/api/tasks/10	/api/bugs/6	Multi Tasks #2 (2)	/api/models/shopping_cart	random	loop	0	completed	3/19/2019	3/19/2019	✓	["maxPathLength":100,"transitionCoverage":100,"placeCoverage":100]	 SHOW  EDIT
<input type="checkbox"/>	/api/tasks/11	/api/bugs/8	Task 7	/api/models/checkout	random	loop	0	completed	3/19/2019	3/19/2019	✓	["maxPathLength":300,"transitionCoverage":100,"placeCoverage":100]	 SHOW  EDIT
<input type="checkbox"/>	/api/tasks/12	/api/bugs/9	Task 8	/api/models/checkout	random	loop	0	completed	3/19/2019	3/19/2019	✓	["maxPathLength":300,"transitionCoverage":100,"placeCoverage":100]	 SHOW  EDIT
<input type="checkbox"/>	/api/tasks/13	/api/bugs/10	Task 9	/api/models/checkout	random	loop	0	completed	3/19/2019	3/19/2019	✓	["maxPathLength":300,"transitionCoverage":100,"placeCoverage":100]	 SHOW  EDIT
<input type="checkbox"/>	/api/tasks/14		Multi Tasks #3 (1)	/api/models/checkout	random	loop	0	completed	3/19/2019	3/19/2019	✓	["maxPathLength":300,"transitionCoverage":100,"placeCoverage":100]	 SHOW  EDIT
<input type="checkbox"/>	/api/tasks/15	/api/bugs/12	Multi Tasks #3 (2)	/api/models/product	random	loop	0	completed	3/19/2019	3/19/2019	✓	["maxPathLength":300,"transitionCoverage":100,"placeCoverage":100]	 SHOW  EDIT
<input type="checkbox"/>	/api/tasks/16	/api/bugs/11	Multi Tasks #3 (3)	/api/models/shopping_cart	random	loop	0	completed	3/19/2019	3/19/2019	✓	["maxPathLength":300,"transitionCoverage":100,"placeCoverage":100]	 SHOW  EDIT
<input type="checkbox"/>	/api/tasks/17	/api/bugs/13	Task 10	/api/models/product	random	random	0	completed	3/19/2019	3/19/2019	✓	["maxPathLength":300,"transitionCoverage":100,"placeCoverage":100]	 SHOW  EDIT
<input type="checkbox"/>	/api/tasks/18	/api/bugs/14	Task 11	/api/models/shopping_cart	random	split	0	completed	3/19/2019	3/19/2019	✓	["maxPathLength":300,"transitionCoverage":100,"placeCoverage":100]	 SHOW  EDIT
<input type="checkbox"/>	/api/tasks/19	/api/bugs/16	Multi Replay #1 (1)	/api/models/product	replay	random	0	completed	3/19/2019	3/19/2019	✓	["bugId":"/api/bugs/1"]	 SHOW  EDIT
<input type="checkbox"/>	/api/tasks/20	/api/bugs/15	Multi Replay #1 (2)	/api/models/shopping_cart	replay	random	0	completed	3/19/2019	3/19/2019	✓	["bugId":"/api/bugs/14"]	 SHOW  EDIT

Rows per page: 1-20 of 20

7. Bug will be displayed in <http://localhost:82/#/bugs>

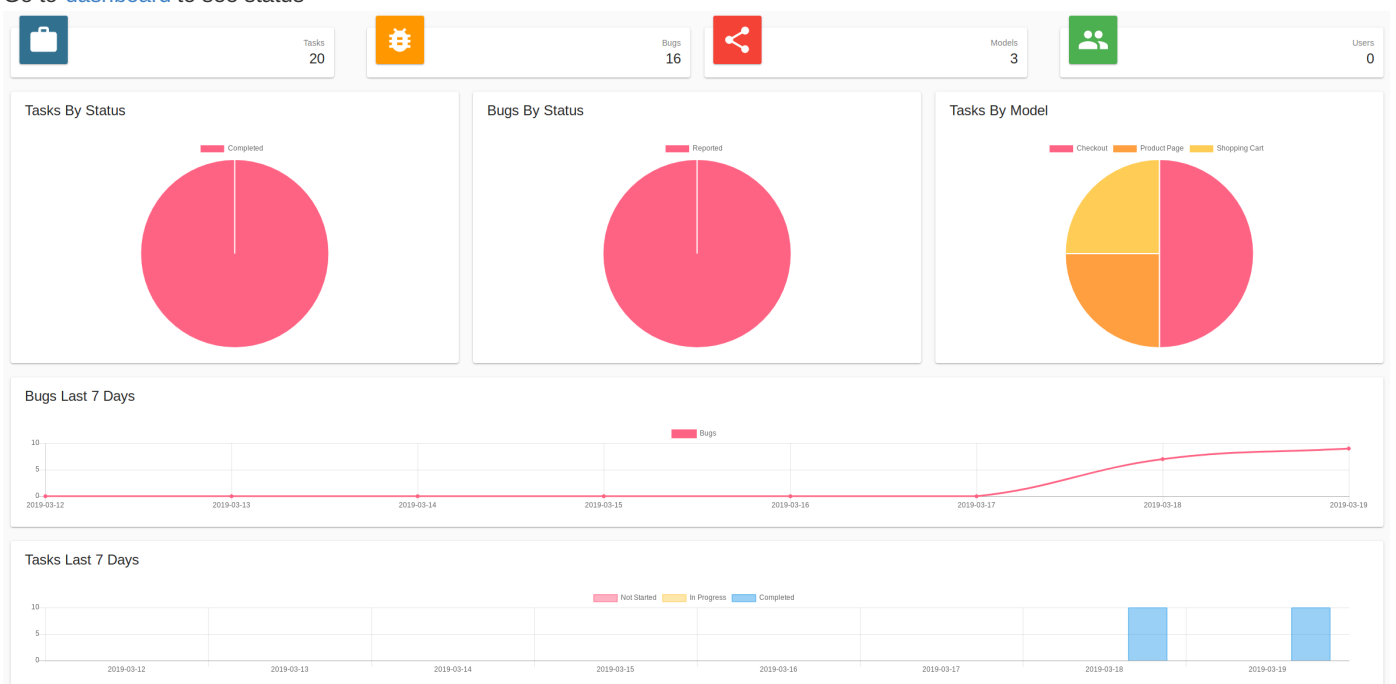
	Step	Transition	Data	Places	Screenshot					
	1			[home]						
<input type="checkbox"/>	/api/bugs/2	New bug found	reported							
	2	viewProductFromHome	["product": "30"]	[product]						
	3	addFromProduct		[product]						
	3					3	/api/tasks/2		0	3/18/2019 3/18/2019
										You need to specify options for this product! Can not add product
	Step	Transition	Data	Places	Screenshot					
	1			[home]						
	2	addProductAndCheckoutNotLoggedIn		[awaitingAccount]						
	3	registerAccount		[awaitingPersonalDetails, awaitingPassword, awaitingBillingAddress]						
<input type="checkbox"/>	/api/bugs/3	New bug found	reported							
	4	fillPassword		[awaitingPersonalDetails, awaitingBillingAddress, passwordFilled]						
	5	fillBillingAddress		[awaitingPersonalDetails, passwordFilled, billingAddressFilled]						
						7	/api/tasks/3		0	3/18/2019 3/18/2019
										Still able to do register account, guest checkout or login when logged in!

8. To view supported models, go to <http://localhost:82/#/models>

	Id	Name	Label	Type	Image		EXPORT
<input type="checkbox"/>	/api/models/checkout	checkout	Checkout	workflow			SHOW
<input type="checkbox"/>	/api/models/product	product	Product Page	workflow			SHOW
<input type="checkbox"/>	/api/models/shopping_cart	shopping_cart	Shopping Cart	state_machine			SHOW
Rows per page: 1-3 of 3							

9. To know how to create your own model (to test your own business logic), please go to [Intermediate](#)

10. Go to [dashboard](#) to see status



Task and Bug

Task

To test a model (or a business logic), all you need to do is create a task. To create a task, we need to provide these information:

+ CREATE + BULK CREATE + BULK REPLAY EXPORT

<input type="checkbox"/>	Id	Bugs	Title	Model	Generator	Reducer	Progress	Status	Created at	Updated at	Take screenshots	Meta data	
<input type="checkbox"/>	/api/tasks/1	/api/bugs/1	Multi Tasks #1 (1)	/api/models/product	random	loop	0	completed	3/18/2019	3/18/2019	✓	{"maxPathLength":300,"transitionCoverage":100,"placeCoverage":100}	SHOW EDIT
<input type="checkbox"/>	/api/tasks/2	/api/bugs/2	Multi Tasks #1 (2)	/api/models/shopping_cart	random	loop	0	completed	3/18/2019	3/18/2019	✓	{"maxPathLength":300,"transitionCoverage":100,"placeCoverage":100}	SHOW EDIT
<input type="checkbox"/>	/api/tasks/3	/api/bugs/3	Task 1	/api/models/checkout	random	loop	0	completed	3/18/2019	3/18/2019	✓	{"maxPathLength":50,"transitionCoverage":100,"placeCoverage":100}	SHOW EDIT
<input type="checkbox"/>	/api/tasks/4	/api/bugs/4	Task 2	/api/models/checkout	random	loop	0	completed	3/18/2019	3/18/2019	✓	{"maxPathLength":300,"transitionCoverage":100,"placeCoverage":100}	SHOW EDIT
<input type="checkbox"/>	/api/tasks/5		Task 3	/api/models/checkout	random	loop	0	completed	3/18/2019	3/18/2019	✓	{"maxPathLength":300,"transitionCoverage":100,"placeCoverage":100}	SHOW EDIT
<input type="checkbox"/>	/api/tasks/6		Task 4	/api/models/checkout	random	loop	0	completed	3/19/2019	3/19/2019	✓	{"maxPathLength":25,"transitionCoverage":100,"placeCoverage":100}	SHOW EDIT
<input type="checkbox"/>	/api/tasks/7	/api/bugs/5	Task 5	/api/models/checkout	random	loop	0	completed	3/19/2019	3/19/2019	✓	{"maxPathLength":50,"transitionCoverage":100,"placeCoverage":100}	SHOW EDIT
<input type="checkbox"/>	/api/tasks/8		Task 6	/api/models/checkout	random	loop	0	completed	3/19/2019	3/19/2019	✓	{"maxPathLength":25,"transitionCoverage":100,"placeCoverage":100}	SHOW EDIT
<input type="checkbox"/>	/api/tasks/9	/api/bugs/7	Multi Tasks #2 (1)	/api/models/product	random	loop	0	completed	3/19/2019	3/19/2019	✓	{"maxPathLength":100,"transitionCoverage":100,"placeCoverage":100}	SHOW EDIT
<input type="checkbox"/>	/api/tasks/10	/api/bugs/6	Multi Tasks #2 (2)	/api/models/shopping_cart	random	loop	0	completed	3/19/2019	3/19/2019	✓	{"maxPathLength":100,"transitionCoverage":100,"placeCoverage":100}	SHOW EDIT
<input type="checkbox"/>	/api/tasks/11	/api/bugs/8	Task 7	/api/models/checkout	random	loop	0	completed	3/19/2019	3/19/2019	✓	{"maxPathLength":300,"transitionCoverage":100,"placeCoverage":100}	SHOW EDIT
<input type="checkbox"/>	/api/tasks/12	/api/bugs/9	Task 8	/api/models/checkout	random	loop	0	completed	3/19/2019	3/19/2019	✓	{"maxPathLength":300,"transitionCoverage":100,"placeCoverage":100}	SHOW EDIT
<input type="checkbox"/>	/api/tasks/13	/api/bugs/10	Task 9	/api/models/checkout	random	loop	0	completed	3/19/2019	3/19/2019	✓	{"maxPathLength":300,"transitionCoverage":100,"placeCoverage":100}	SHOW EDIT
<input type="checkbox"/>	/api/tasks/14		Multi Tasks #3 (1)	/api/models/checkout	random	loop	0	completed	3/19/2019	3/19/2019	✓	{"maxPathLength":300,"transitionCoverage":100,"placeCoverage":100}	SHOW EDIT
<input type="checkbox"/>	/api/tasks/15	/api/bugs/12	Multi Tasks #3 (2)	/api/models/product	random	loop	0	completed	3/19/2019	3/19/2019	✓	{"maxPathLength":300,"transitionCoverage":100,"placeCoverage":100}	SHOW EDIT
<input type="checkbox"/>	/api/tasks/16	/api/bugs/11	Multi Tasks #3 (3)	/api/models/shopping_cart	random	loop	0	completed	3/19/2019	3/19/2019	✓	{"maxPathLength":300,"transitionCoverage":100,"placeCoverage":100}	SHOW EDIT
<input type="checkbox"/>	/api/tasks/17	/api/bugs/13	Task 10	/api/models/product	random	random	0	completed	3/19/2019	3/19/2019	✓	{"maxPathLength":300,"transitionCoverage":100,"placeCoverage":100}	SHOW EDIT
<input type="checkbox"/>	/api/tasks/18	/api/bugs/14	Task 11	/api/models/shopping_cart	random	split	0	completed	3/19/2019	3/19/2019	✓	{"maxPathLength":300,"transitionCoverage":100,"placeCoverage":100}	SHOW EDIT
<input type="checkbox"/>	/api/tasks/19	/api/bugs/16	Multi Replay #1 (1)	/api/models/product	replay	random	0	completed	3/19/2019	3/19/2019	✓	{"bugId":"/api/bugs/1"}	SHOW EDIT
<input type="checkbox"/>	/api/tasks/20	/api/bugs/15	Multi Replay #1 (2)	/api/models/shopping_cart	replay	random	0	completed	3/19/2019	3/19/2019	✓	{"bugId":"/api/bugs/14"}	SHOW EDIT


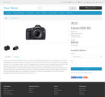
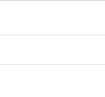




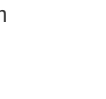
Rows per page 1-20 of 20

1. Title: a short description of the task, to tell the difference between 2 tasks
2. Model: which model you want to test
3. Generator: how to test the model
 1. Random: go through transitions of the model randomly
 2. Probability: go through transitions in a probability defined in the model
 3. All places (state machine only): try to go through transitions that can cover all places as quick as possible
 4. All transitions (state machine only): try to go through all transitions as quick as possible
 5. Replay: Test regression
4. Meta Data: additional information to generator
 1. Max path length: how many steps max (for random and probability generators)
 2. Transition coverage (for random generator)
 3. Place coverage (for random generator)
 4. Bug (for replay generator)
5. Reducer: when a bug is captured, how to reduce the reproduce steps
 1. Random: choose 2 random places in the reproduce path, replace the steps between them by the shortest path, then try to reproduce again, then repeat
 2. Loop: choose 2 places in the reproduce path that are exactly the same, remove all steps between them, then try to reproduce again, then repeat
 3. Split: divide the reproduce path by 2, 3, 4 etc parts, loop through each part, replace the steps between each part by the shortest path, then try to reproduce
6. Capture Screenshots: Should we capture screenshots for the bug we found (for testing UI only)

7. Report Bug: Should we report bug to external places (e.g. email, slack)

Bug

Anything that prevent the **Generator** to complete walking through the model can be captured and reported as a bug. When a bug is found, we can see these information:

<div><input type="checkbox"/></div> <div>/api/bugs/2</div> <div>New bug found</div> <div>reported</div>	Step	Transition	Data	Places	Screenshot	3	/api/tasks/2	You need to specify options for this product! Can not add product	0	3/18/2019	3/18/2019
	1			[home]							
	2	viewProductFromHome	{product:"30"}	[product]							
<div><input type="checkbox"/></div> <div>/api/bugs/3</div> <div>New bug found</div> <div>reported</div>	3	addFromProduct		[product]		7	/api/tasks/3	Still able to do register account, guest checkout or login when logged in!	0	3/18/2019	3/18/2019
	1			[home]							
	2	addProductAndCheckoutNotLoggedIn		[awaitingAccount]							
	3	registerAccount		[awaitingPersonalDetails","awaitingPassword","awaitingBillingAddress]							
	4	fillPassword		[awaitingPersonalDetails","awaitingBillingAddress","passwordFilled]							
	5	fillBillingAddress		[awaitingPersonalDetails","passwordFilled","billingAddressFilled]							

1. Title: you can change the default title of the bug, so that it can tell what is the real problem
2. Status: can be

1. New

2. Reducing

3. Reported
3. Path: Reproduce steps (or path)
4. Length: Number of steps in the reproduce path
5. Task: Which task this bug belong to
6. Bug Message: The real error that prevent the generator to complete its job
7. Messages count
8. Created at
9. Updated at

Tips and Tricks

[Home](#) / [Beginner](#) / Tips-Tricks

Tips and tricks

1. Try to split into small models
 1. To make model less complicated
 2. To increase testing speed
2. We can only edit title of tasks and bugs
3. If bug's replicate steps is too long, try to reduce with another path reducer
4. If the bug is fixed, try creating new replay task to test it again

Bulk Actions

There are 4 bulk actions:

1. Create bulk tasks action

Title *

Task 1

Models *

Checkout

Product Page

Shopping Cart

▼

Generator *

Random

Meta Data

Max Path Length

300

Transition Coverage

100

Place Coverage

100

Reducer *

Loop

☒ Take Screenshots

SAVE

2. Create bulk replays action

Title *

Task 1

Generator

Replay

Bug *

The first bug

The last bug

New bug found

▼

Reducer *

Loop


☒ Take Screenshots

SAVE

3. Bulk report bugs action

Bugs *

New bug found

 SAVE


4. Bulk reduce bugs action

Bugs *

New bug found

Reducer *

Transition

 SAVE

Intermediate

[Home](#) / Intermediate

For people who want to write your own models

- [Init Project](#)
- [Model](#)
- [Subject](#)
- [Debug](#)
- [Weight and Probability](#)
- [Workflow and State Machine](#)
- [Default Users](#)
- [Generator](#)
- [Path Reducer](#)
- [Test Model](#)
- [Report Bug](#)

Init Project

[Home](#) / [Intermediate](#) / Init-Project

There are 3 main steps that will need to do when you start testing your project:

1. Create bundle that contain models and subjects

1. Create a blank bundle:

```
$ composer create-project tienvx/mbt-bundle-skeleton my-project
```

2. If you feel lost, clone this bundle and edit it instead

```
$ git@github.com:tienvx/mbt-examples-bundle.git
```

3. To make life easier, we created 3 commands to allow you to generate code from your model

```
$ tests/app/bin/console make:generator name ClassName  
$ tests/app/bin/console make:subject name ClassName  
$ tests/app/bin/console make:reducer name ClassName
```

2. Build docker images that contains the bundle

1. Create 2 docker files (name it Dockerfile):

1. Api

```
FROM tienvx/mbt-api:v1.6.1  
RUN curl -sSL https://getcomposer.org/composer.phar -o /usr/bin/composer && \  
  chmod +x /usr/bin/composer && \  
  composer require -q your-name/your-bundle && \  
  composer clearcache -q && \  
  rm /usr/bin/composer
```

2. Worker

```
FROM tienvx/mbt-worker:v1.6.1  
RUN curl -sSL https://getcomposer.org/composer.phar -o /usr/bin/composer && \  
  chmod +x /usr/bin/composer && \  
  composer require -q your-name/your-bundle && \  
  composer clearcache -q && \  
  rm /usr/bin/composer
```

2. Build them

1. Api

```
$ docker build -t your-name/your-api-image -f path/to/Dockerfile .
```

2. Worker

```
$ docker build -t your-name/your-worker-image -f path/to/Dockerfile .
```

3. Deploy

1. Create docker-compose.yaml file to contains all services. You can use this [example](#)




2. Replace api and worker with your image

```
api:  
  image: "your-name/your-api-image"  
worker:  
  image: "your-name/your-worker-image"
```

Model

[Home](#) / [Intermediate](#) / Model

A model is a yaml file that describe your business logic Here are some example models:

<input type="checkbox"/>	Id	Name	Label	Type	Image	EXPORT
<input type="checkbox"/>	/api/models/checkout	checkout	Checkout	workflow		SHOW
<input type="checkbox"/>	/api/models/product	product	Product Page	workflow		SHOW
<input type="checkbox"/>	/api/models/shopping_cart	shopping_cart	Shopping Cart	state_machine		SHOW

Rows per page: 1-3 of 3

These information are needed when defining a model:

1. type: can be workflow or state_machine
 1. workflow: can be in more than one place simultaneously
 2. state_machine: cannot be in more than one place simultaneously
2. supports
3. metadata
 1. label: describe what model is, to tell difference between 2 models
 2. tags
4. places: list all places in this model
5. initial_place: the first place for **generator** to walk through the model
6. transitions: list all transitions in this model
 1. from: the start place
 2. to: the end place
 3. guard: the condition of this condition, whether the **generator** can go through this transition or not
 4. metadata
 1. label: describe the transition
 2. probability: How often a transition appear in the path
 3. weight: How much effort to do a transition

Here are some useful tips:

1. Try to make your model simple
2. Try to split your business logic into small models
3. Try to use directories to manage your models

Subject

[Home](#) / [Intermediate](#) / Subject

A subject will help model interact with the system under test

To create a subject for a new model, run this command

```
$ tests/app/bin/console make:subject name ClassName
```

Here are some useful tips:

1. Subject can be use to test any project, from api to web applications
2. Subject must implement `SubjectInterface`, which must have method `support()` to tell which model it belong to
3. When you see a bug, throw new exception, but only if value of the property `testing` is equals to `true`

Debug

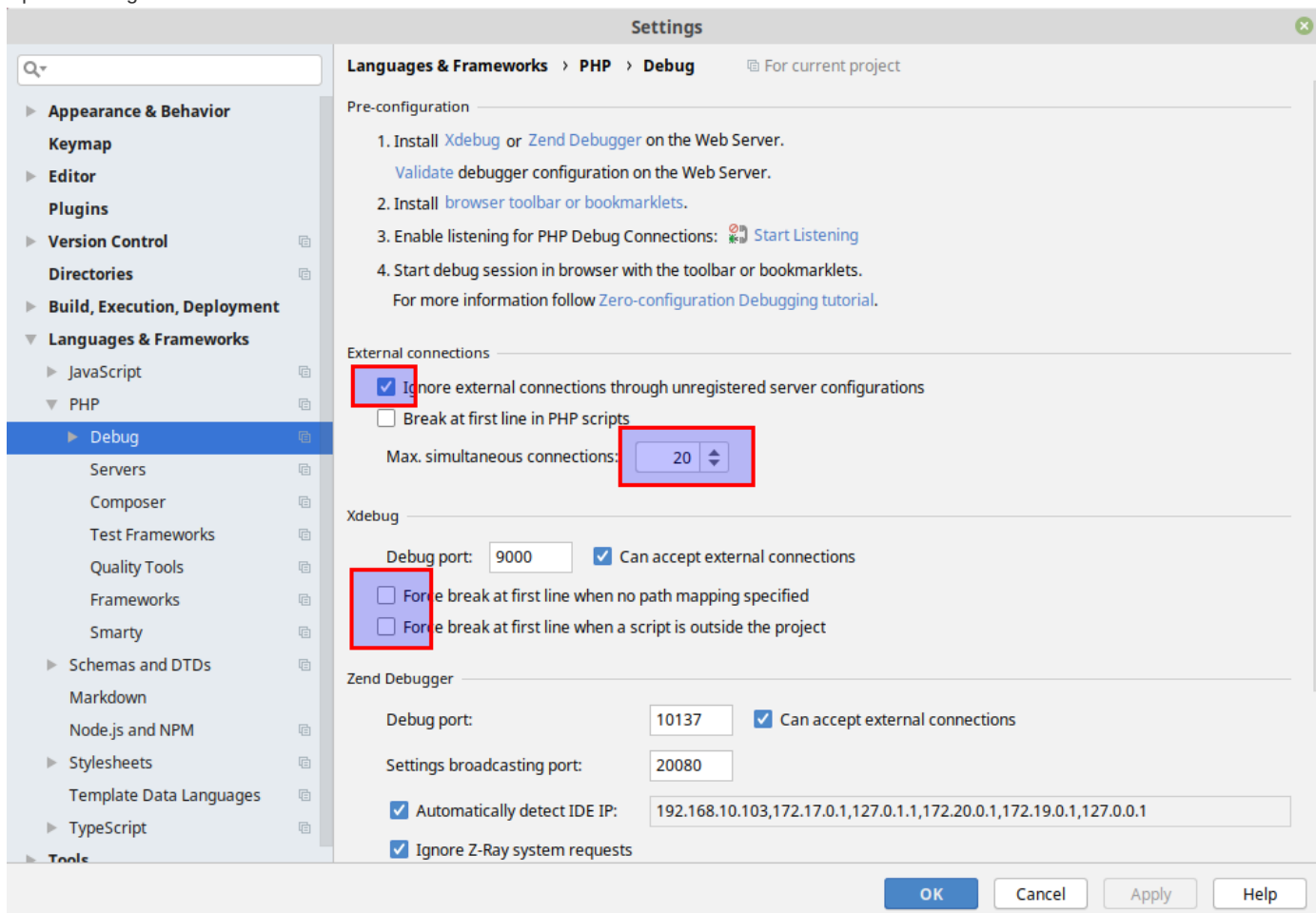
[Home](#) / [Intermediate](#) / Debug

Debug Api

1. Apply this patch

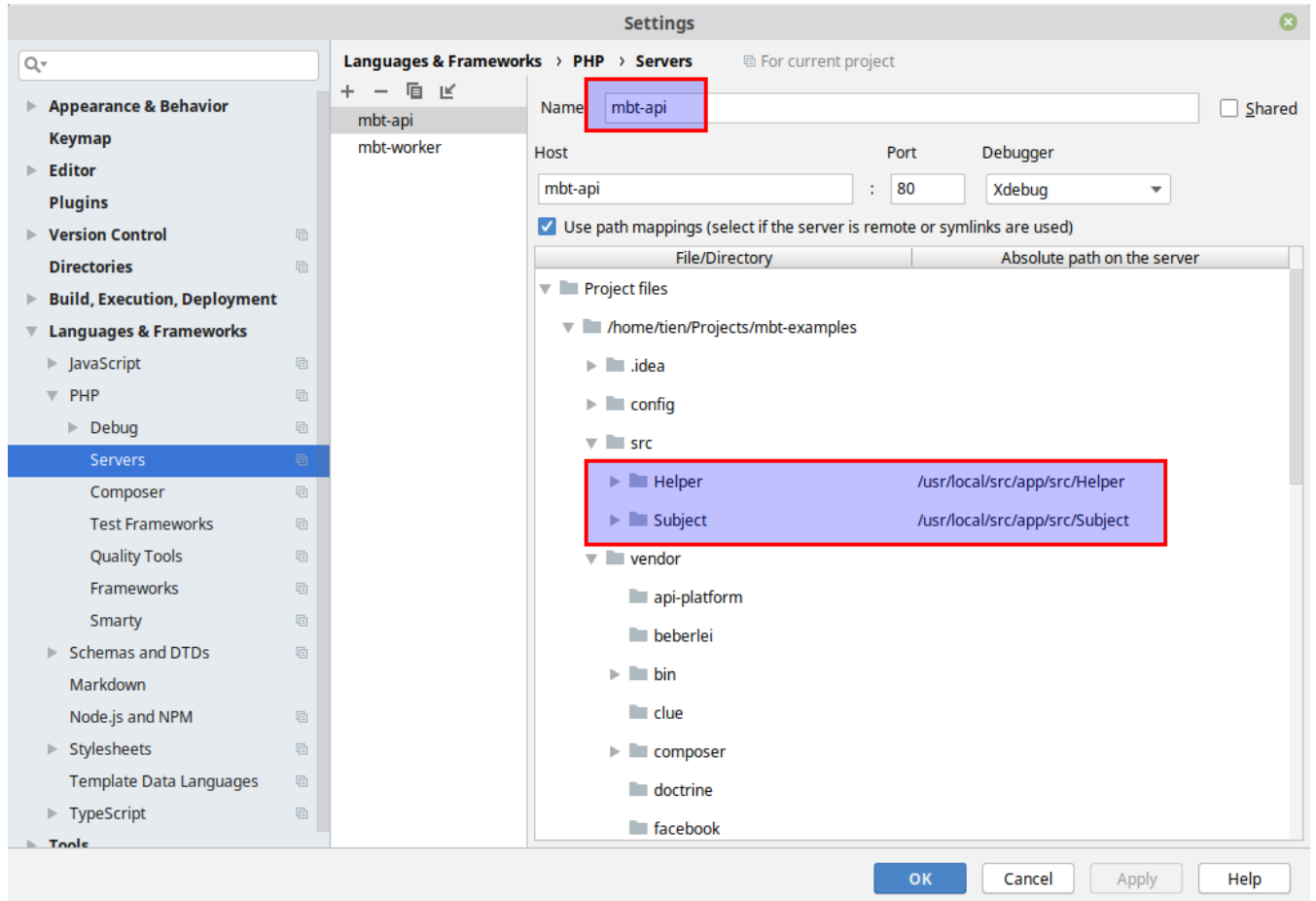
```
diff --git a/docker-compose.yml b/docker-compose.yml
index 6584338..8a43102 100644
--- a/docker-compose.yml
+++ b/docker-compose.yml
@@ -16,7 +16,7 @@ services:
   depends_on:
     - db
   api:
-    image: 'tienvx/mbt-api'
+    image: 'tienvx/mbt-api:debug'
   ports:
     - 80:80
   volumes:
@@ -33,7 +33,7 @@ services:
   USER_PASSWORD: '$$2y$$13$$gWuD7B7irvHWJo7kMx9w8eJpLEuSZiR5iMgK1p2foCeYPsjsAk0k.'
   ADMIN_PASSWORD: '$$2y$$13$$3rT10fXyh9o12mV$SaPnbmu92HX3anslUtXabjXu6zdYojpopeFVba'
   worker:
-    image: 'tienvx/mbt-e2e-worker'
+    image: 'tienvx/mbt-e2e-worker:debug'
   volumes:
     - './config/packages/dev/models:/usr/local/src/app/config/packages/dev/models'
     - './src/Subject:/usr/local/src/app/src/Subject'
@@ -46,6 +46,7 @@ services:
   environment:
     DATABASE_URL: 'mysql://root:root@db:3306/db'
     MESSENGER_TRANSPORT_DSN: 'amqp://guest:guest@queue:5672/%2f/messages'
+    PHP_IDE_CONFIG: "serverName=mbt-worker"
   admin:
     image: 'tienvx/mbt-admin'
     ports:
```

2. Update configuration

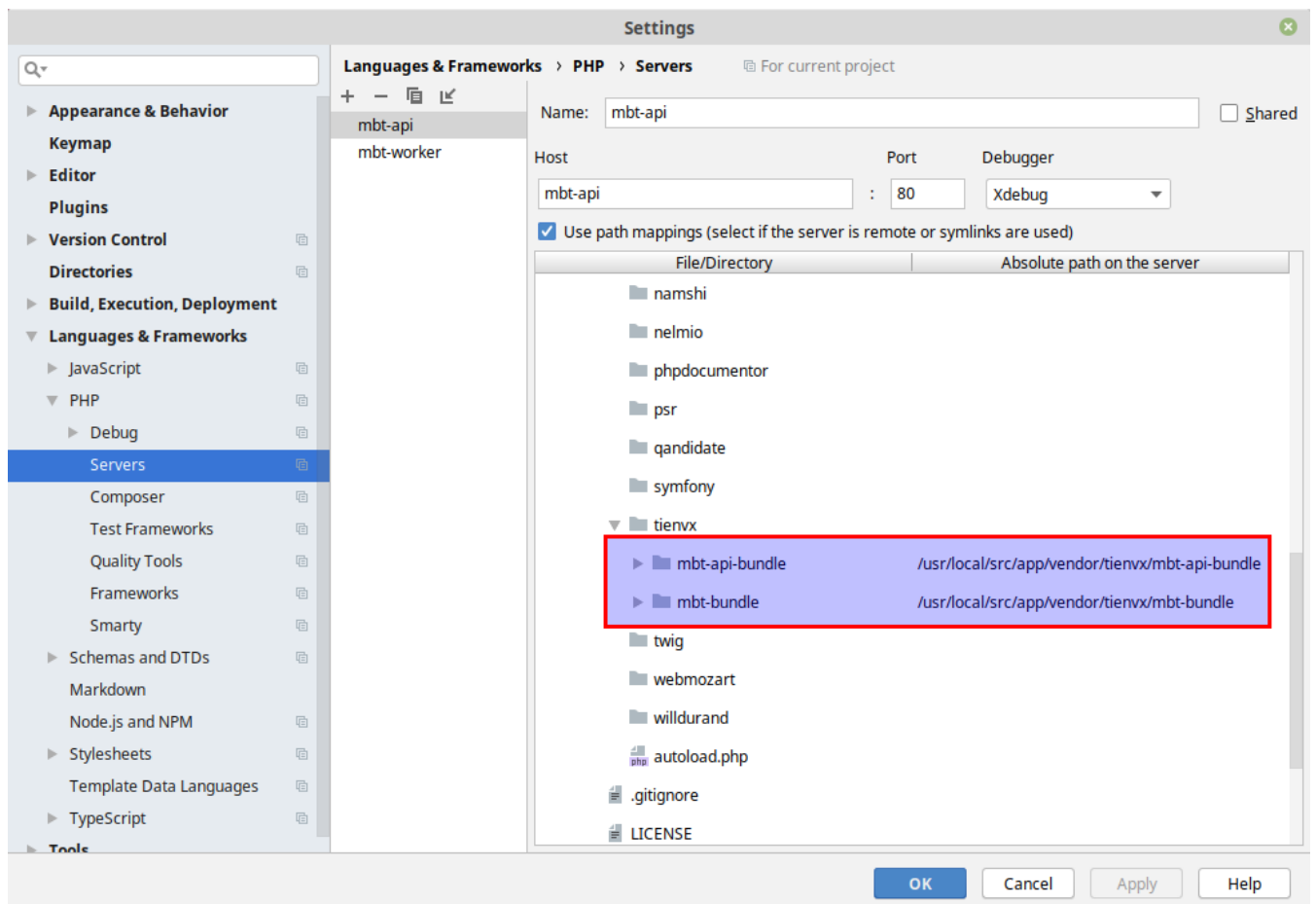


3. Set mapping

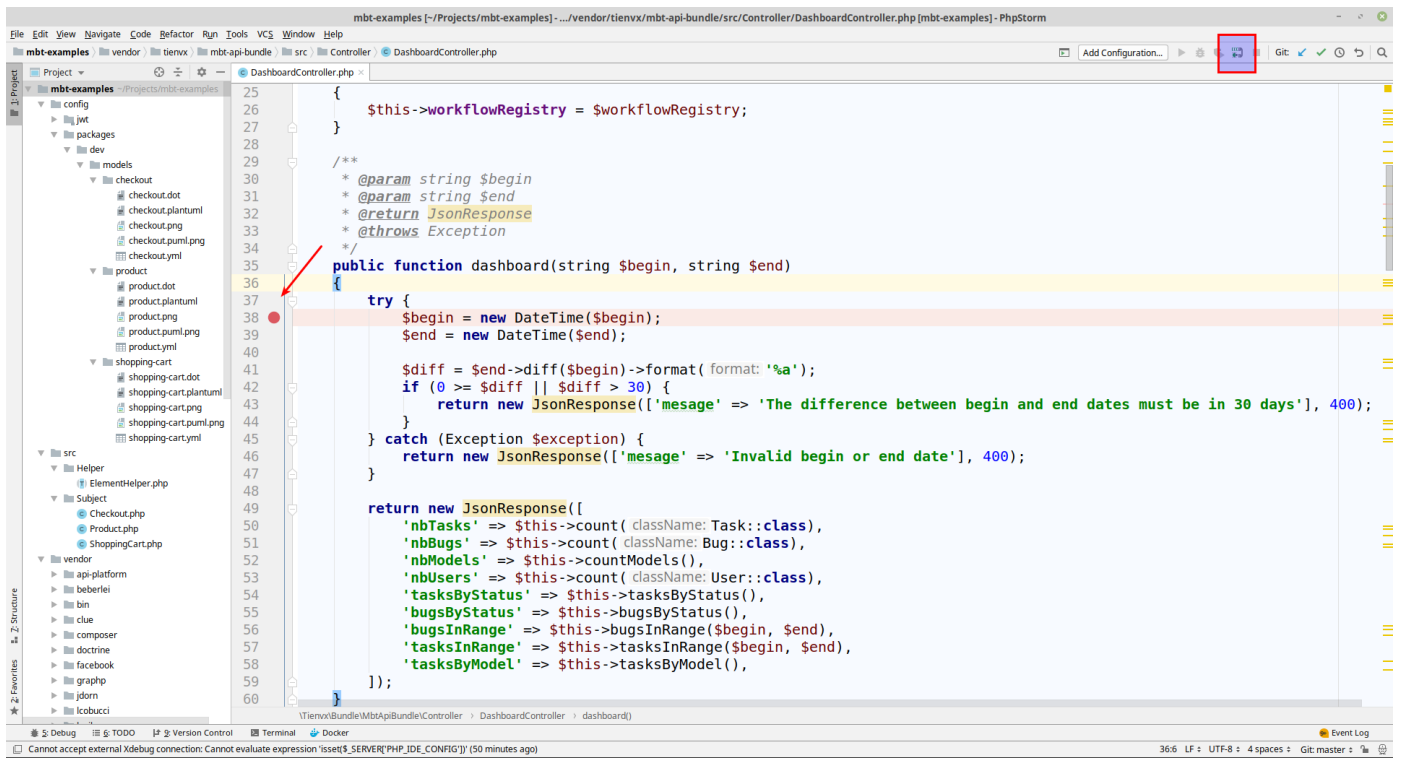
1. Subject



2. Vendor



4. Set breakpoints and start listening



Then you can able to debug the api.

Debug worker

TBD

Weight and Probability

[Home](#) / [Intermediate](#) / Weight-Probability

Weight

How much effort to do a transition

- For Random and Binary reducers (that use [Dijkstra](#) algorithm
- Min: 0
- Default: 1
- Not necessary integer
- The re-produce steps will contains transitions that have lower weight

Probability

How often a transition appear in the path

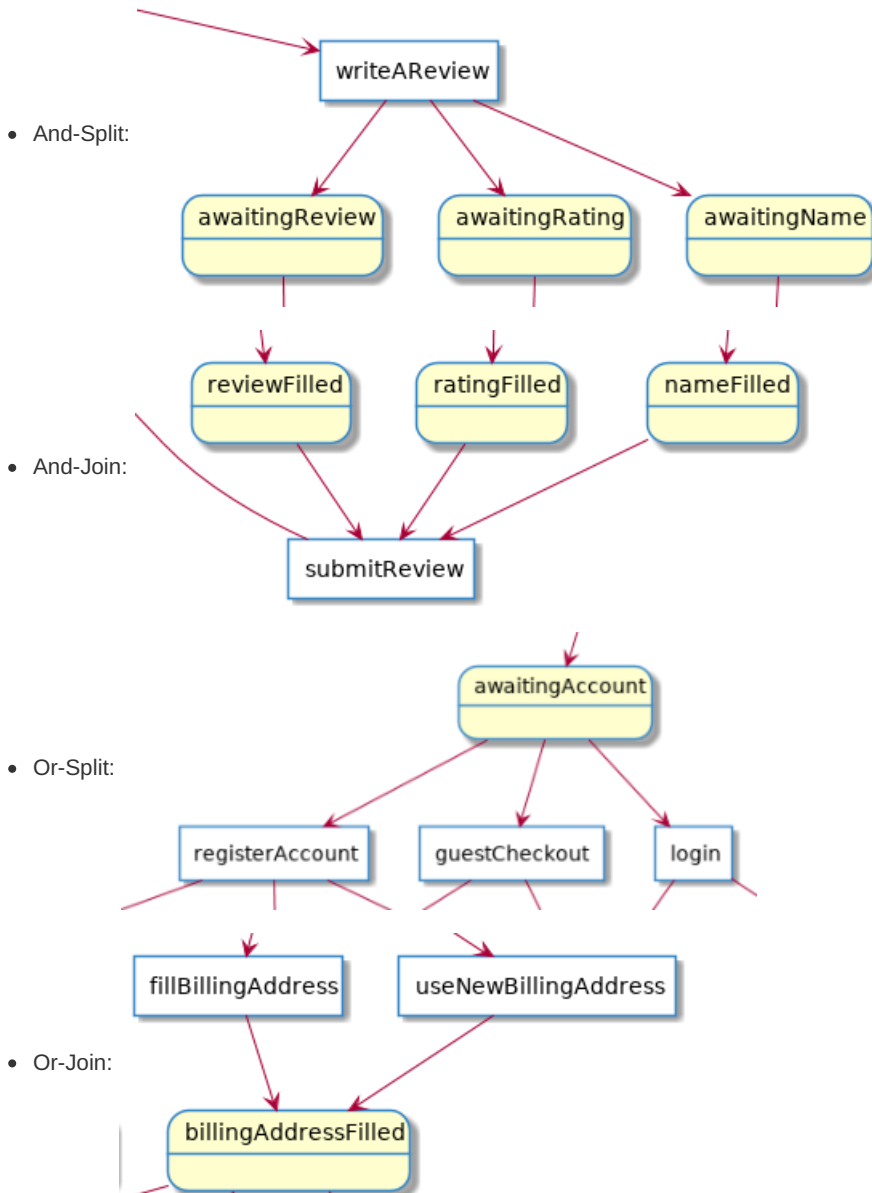
- For Probability generator
- Min: 1
- Default: 1
- Must be positive integer
- The number is relative to each other. Not necessary percentage
- More information [here](#)

Workflow and State Machine

[Home](#) / [Intermediate](#) / Workflow-State-Machine

Terminologies

- Workflow-Net: contains workflow and state machine, is a sub-class of Petri-Net



Workflow

- Can be in more than one place simultaneously
- Can be used to describe more complex logic than state machine
- Can contains And-Split, And-Join, Or-Split and Or-Join

State Machine

- Cannot be in more than one place simultaneously
- Can be used to describe simple logic
- Can only contains Or-Split and Or-Join

Default Users

[Home](#) / [Intermediate](#) / Default-Users

To get password for default users, run these command:

```
$ docker-compose exec api bin/console security:encode-password
$ # choose 1 (Tienvx\Bundle\MbtApiBundle\Entity\User)
$ # then type your password
$ # then replace '$' by '$$'
$ # then put it in docker-compose.yml
```

Generator

[Home](#) / [Intermediate](#) / Generator

A generator will help you generate test cases to test the system under test.

Currently there are 4 built-in generators:

1. Random: select the next transition randomly
2. Probability: select the next transition based on the probability of the transition
3. All Places: select the next transition in order to cover all places as fast as possible
4. All Transitions: select the next transition in order to cover all transitions as fast as possible

To create your own generator, run this command

```
$ tests/app/bin/console make:generator name ClassName
```

Path Reducer

[Home](#) / [Intermediate](#) / Path-Reducer

A path reducer will help reducing the reproduce steps of the bug.

Currently there are 3 built-in path reducers:

1. Binary: divide the reproduce steps into 2, 4, 8 parts and so on
2. Loop: look for the same places in the reproduce steps and remove all transitions between them
3. Random: select 2 random places in the reproduce steps, replace transitions between them by the shortest path

To create your own path reducer, run this command

```
$ tests/app/bin/console make:reducer name ClassName
```

Test Model

After creating a model, you may want to test it before writing subject. To test a model:


- 1. Go to [Models](#)
- 2. Click [Test Action](#)
- 3. Choose a model and generate with its meta data

Here is result:

Models *
Shopping Cart

Generator *
All Places

Step	Transition	Data	Places
1			["home"]
2	viewAnyCategoryFromHome	{"category":"24"}	["category"]
3	viewProductFromCategory	{"product":"28"}	["product"]
4	viewCartFromProduct	[]	["cart"]
5	checkoutFromCart	[]	["checkout"]
6	backToHomeFromCheckout	[]	["home"]

 SAVE

Report Bug

To report a bug to external service (e.g. Slack, Gmail), we have to do 2 things:

- 1. Configure services we want to report the bug. There are 3 places to do so:
 - 1. List services out in `reporters.yaml`. Here are some example configuration taken from `mbt-examples` project:

```
monolog:
  channels: ['mbt']
  handlers:
    slack:
      type: slack
      token: 'xxxx-xxxxxxxxxxxx-xxxxxxxx-xxxxxxxx-xxxxxx'
      channel: '#name-of-channel'
      channels: [mbt]
      level: error
    swift:
      type: swift_mailer
      from_email: 'mbt@example.com'
      to_email: 'error@example.com'
      # or list of recipients
      # to_email: ['dev1@example.com', 'dev2@example.com', ...]
      subject: 'A Bug Was Found!'
      formatter: mbt.bug_formatter
      content_type: text/html
      channels: [mbt]
      level: error
```

- 2. Depend on each service, we may need to add additional environment variables to `docker-compose.yml`, for example:

```
MAILER_URL: 'smtp://localhost:25?encryption=ssl&auth_mode=login&username=&password='
```

- 3. If service need additional code to work, we need to install the bundle in `Dockerfile`

```
RUN composer require symfony/swiftmailer-bundle
```

- 2. When creating new task, remember toggle 'Report Bug' on

Title *

Task 1

Models *

Generator *

Reducer *

Take Screenshots

Report Bug

SAVE

Enjoy:

Monolog APP 2:59 PM

Message

You need to specify options for this product! Can not add product

Level



ERROR

Today at 2:59 PM

ERROR	
Bug ID:	3
Bug Title:	New bug found
Task Title:	Bulk Tasks #1 (1)
Bug Message:	Can not upload file!
Steps:	<div> <div>Step:</div> <div>1</div> <div>Transition:</div> <div>Data:</div> <div>Places:</div> <div>product</div> </div> <div> <div>Step:</div> <div>2</div> <div>Transition:</div> <div>selectOptions</div> <div>Data:</div> <div>[]</div> <div>Places:</div> <div>awaitingRadio awaitingCheckbox awaitingText awaitingSelect awaitingTextarea awaitingFile awaitingDate awaitingTime awaitingDateTime</div> <div>Screenshot:</div> <div>Screenshot</div> </div> <div> <div>Step:</div> <div>3</div> <div>Transition:</div> <div>selectDate</div> <div>Data:</div> <div>[]</div> <div>Places:</div> <div>awaitingRadio awaitingCheckbox awaitingText awaitingSelect awaitingTextarea awaitingFile awaitingTime awaitingDateTime dateSelected</div> <div>Screenshot:</div> <div>Screenshot</div> </div> <div> <div>Step:</div> <div>4</div> <div>Transition:</div> <div>fillTextarea</div> <div>Data:</div> <div>[]</div> <div>Places:</div> <div>awaitingRadio awaitingCheckbox awaitingText awaitingSelect awaitingFile awaitingTime awaitingDateTime dateSelected textareaFilled</div> <div>Screenshot:</div> <div>Screenshot</div> </div> <div> <div>Step:</div> <div>5</div> <div>Transition:</div> <div>selectTime</div> <div>Data:</div> <div>[]</div> </div>

Notes:

1. If you don't know how to get smtp information for gmail, check it out [here](#)
2. If you don't know how to get slack api token, check it out [here](#)
3. If you want to verify the bug before report it:
 1. Don't toggle 'Report Bug' on while creating tasks
 2. Once you done verify the bug, use report bulk action

<div><div></div>2 items selected</div>										
ID	Title	Status	Path	Step	Transition	Data	Places	Screenshot	Length	Task
<input checked="" type="checkbox"/>	/api/bugs/1	New bug found	reported	1			[home]		2	/api/task/1
				2	addFromHome	{"product": "30"}	[home]			You need to specify options for this product! Can not add product.
<input checked="" type="checkbox"/>	/api/bugs/2	New bug found	reported	1			[home]		2	/api/task/2
				2	addFromHome	{"product": "30"}	[home]			You need to specify options for this product! Can not add product.

Advanced

[Home](#) / Advanced

For people who want to deploy to production

- [Deploy](#)
- [CI Integration](#)
- [Unit Testing](#)

Deploy

[Home](#) / [Advanced](#) / Deploy

Docker Swarm

On manager node

```
$ docker swarm init
$ # docker swarm init --advertise-addr vboxnet0
$ docker-machine create worker-node
$ docker swarm join-token worker
$ env MBT_VERSION=v1.6.0 docker stack deploy -c docker-compose.yml mbt
$ docker service scale mbt_worker=4
```

On worker node

```
$ docker-machine ssh worker-node
$ docker swarm join --token TOKEN IP:PORT
```

Note: The stack is not working as expected because of invalid mount config and links in worker-node.

These links may help to fix invalid mount config:

- <https://blog.dahanne.net/2017/11/20/docker-swarm-and-nfs-volumes/>
- https://www.reddit.com/r/docker/comments/7p069n/docker_swarm_remote_volumes_best_practices/

Useful commands:

```
$ docker stack ls
$ docker stack ps mbt
$ docker stack services mbt
$ docker service logs mbt_api
$ docker-machine ssh worker-node
$ docker stack rm mbt
$ docker swarm leave
$ docker-machine ls
$ docker-machine rm worker-node
$ docker swarm leave --force
```

Kubernetes

```
$ env MBT_VERSION=v1.6.0 docker stack deploy --orchestrator=kubernetes -c docker-compose.yml mbt
```

Note: We need to install compose-on-kubernetes first <http://collabnix.com/a-first-look-at-compose-on-kubernetes-for-minikube/>

Useful commands:

```
$ minikube start
$ minikube status
$ minikube delete
$ kubectl get nodes
$ kubectl create namespace compose
$ kubectl -n kube-system create serviceaccount tiller
$ kubectl -n kube-system create clusterrolebinding tiller --clusterrole cluster-admin --serviceaccount kube-system:tiller
$ helm init --service-account tiller
$ helm version
$ helm install --name etcd-operator stable/etcd-operator --namespace compose
$ kubectl api-versions| grep compose
$ minikube service list
$ docker stack ls --orchestrator=kubernetes
$ kubectl describe pod podname
$ kubectl get deployment
$ kubectl get pods
$ kubectl get services
$ docker stack rm mbt --orchestrator=kubernetes
```

Here is how to install kubectl, minikube, helm, compose-on-kubernetes:

```
$ sudo snap install kubectl --classic
$ sudo snap install kubeadm --classic
$ sudo snap install kubelet --classic

$ curl -Lo minikube https://storage.googleapis.com/minikube/releases/latest/minikube-linux-amd64 ; chmod +x minikube ; sudo cp minikube /usr/local/bin ; rm minikube
$ minikube start

$ sudo snap install helm --classic

$ wget https://github.com/docker/compose-on-kubernetes/releases/download/v0.4.21/installer-linux
$ chmod a+x installer-linux
$ ./installer-linux -namespace=compose -uninstall
$ ./installer-linux -namespace=compose -etcd-servers=http://compose-etcd-client:2379 -tag=v0.4.21
```

CI Integration

[Home](#) / [Advanced](#) / CI

We support 3 endpoints so that we can integrate it with CI:

1. Create bulk tasks for all models

```
curl 'http://localhost/mbt-api/bulk-create-all' -H 'Content-Type: application/json' -H 'Authorization: Bearer [TOKEN]' --data '{"title":"Bulk Tasks For All Models","generator":"random","metaData":{"maxPathLength":300,"transitionCoverage":100,"placeCoverage":100},"reducer":"loop","takeScreenshots":true,"reportBug":true}'
```

2. Create bulk tasks by model tag(s)

```
curl 'http://localhost/mbt-api/bulk-create-tags' -H 'Content-Type: application/json' -H 'Authorization: Bearer [TOKEN]' --data '{"title":"Bulk Tasks By Model's Tags","tags":["shopping cart","checkout","product"],"generator":"random","metaData":{"maxPathLength":300,"transitionCoverage":100,"placeCoverage":100},"reducer":"loop","takeScreenshots":true,"reportBug":true}'
```

3. Create bulk tasks by models

```
curl 'http://localhost/mbt-api/bulk-create' -H 'Content-Type: application/json' -H 'Authorization: Bearer [TOKEN]' --data '{"title":"Bulk Tasks By Models","models":["/mbt-api/models/article","/mbt-api/models/pull_request"],"generator":"random","metaData":{"maxPathLength":300,"transitionCoverage":100,"placeCoverage":100},"reducer":"loop","takeScreenshots":true,"reportBug":true}'
```

Note:

- Login to get token

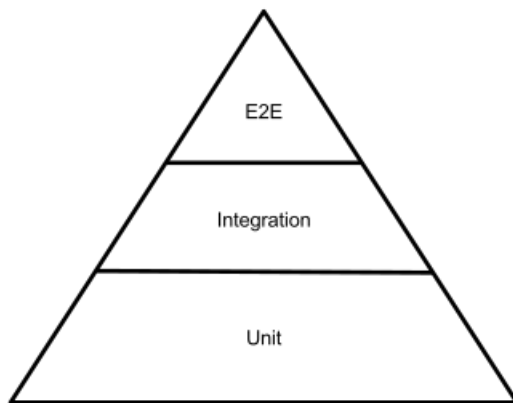
```
curl 'http://localhost/mbt-api/login_check' -H 'Content-Type: application/json' --data '{"username":"admin","password":"admin"}
```

- By default, token last for 1 hour

Unit Testing

[Home](#) / [Advanced](#) / Unit-Testing

Set up



In testing pyramid, MBT-Bundle work best with E2E and Integration testing. What if I want to apply MBT-Bundle to unit testing? Although we do not recommend you to do so, but there is a way to do it:

1. Put these [services](#) into your CI config file
2. In your CI config file, create a server with your latest code from development branch
3. From your subject (PHP code), request to that server to test
4. Finally, continue edit your CI config file, create tasks to start testing your service:

```
curl 'http://localhost/mbt-api/bulk-create-tags' -H 'Content-Type: application/json' -H 'Authorization: Bearer [TOKEN]' --data '{"title":"Bulk Tasks By Model's Tags","tags":["shop ping cart","checkout","product"],"generator":"random","metaData":{"maxPathLength":300,"transitionCoverage":100,"placeCoverage":100},"reducer":"loop","takeScreenshots":true,"reportBug":true}'
```

Notes

1. The docker image `tienvx/mbt-unit` only support slack reporter, you may want to extend it, add more libraries to allow to send bug reports to where you want
2. The runner only have 1 worker, so this work best if your service have small number of models. If your unit test cases run very slow, try split your code with related models into a new service.