



NabiCI

Continuous Integration Framework for PHP

User Guide

Software: Version 1.0

11/2009

written by Derya Oez

HWR Berlin School of Economics and Law

TABLE OF CONTENTS

1 INTRODUCTION	1
2 SYSTEM REQUIREMENTS	1
3 HOW TO GET THE SOFTWARE	2
4 QUICK START DEMONSTRATION	2
4.1 Test Files	3
4.2 Run 'test_loop'	3
5 CONFIGURATION OF A BUILD LOOP	8
5.1 Loop Section	9
5.2 Task Section	9
5.2.1 url	10
5.2.2 system.....	11
5.2.3 system_list.....	11
5.2.4 url_list	12
5.2.5 system_linewise	12
6 RUNNING OF THE BUILD LOOP	13
7 BUILD LOCK	13
7.1 build.lock.....	13
7.2 stop_ci.lock	14
8 VISUALIZATION OF THE CONTINUOUS INTEGRATION PROCESS .	14
9 ACCESS PROTECTION FOR RUN TASK	16
10 GENERAL NOTES	17

1 Introduction

Thank you for choosing NabiCI, the Continuous Integration framework for PHP.

According to software development Continuous Integration is an approach, where a software project is completely built and tested automatically in regular intervals. The target is to detect integration errors as soon as possible and to fix them ¹.

NabiCI is a software for automated execution of Continuous Integration loops.

A Continuous Integration loop is a series of commands which build and test a development project automatically. These commands are executed in regular intervals to make sure that the project has no integration errors. A loop usually contains syntax checks, unit tests, and automatic build and install routines. NabiCI visualizes the results of the Continuous Integration loop on web pages which can be monitored by the developers. Thus, each developer is always informed of the state of the project. As monitoring software, we recommend the freeware "siteUP" (<http://siteup.softonic.de/>).

This guide offers among other things a description for a quick start of the software to get the first impression using prepared test files. In addition to this there is a more detailed explanation for the configuration of your own Continuous Integration loop which is especially recommended to read for easy usage of NabiCI. Also the visualization and user options for users are described in it.

To learn more about NabiCI see the next chapters.

Have fun!

2 System requirements

NabiCI is a PHP script. You need PHP5 or greater to run NabiCI.

¹ Get more about Continuous Integration e.g. from
<http://www.martinfowler.com/articles/continuousIntegration.html>

3 How to get the software

A freeware release of NabiCI is offered for download from the web page

http://www.nabidoo.de/ci_framework.php .

After download you need to extract the files.

Note: keep the folder structure for correctly working

4 Quick Start Demonstration

This is just a simple demonstration of NabiCI using some prepared test files. The target is to get a quick insight before configuration for your own Continuous Integration loop.

Note: before configuration of your Loop it is strongly recommended to read [Configuration of a Build Loop](#)

4.1 Test Files

The subfolder '**Test_Files**' of '**NabiCI**' contains all files necessary for starting a test run. Here is a description of the files:

Test file	Explanation
SyntaxCheckResult.php	<ul style="list-style-type: none">• example of a result of a syntax check• content of file will be checked linewise
SyntaxCheckResultOK.php	<ul style="list-style-type: none">• equal to the SyntaxCheckResult.php
system_list.txt	<ul style="list-style-type: none">• a list of system commands• The file is called when running the loop. All commands in it will be executed one after other.
url_list.txt	<ul style="list-style-type: none">• a list of URL's• The file is called when running the loop. All commands in it will be executed one after other.
system_test.php	<ul style="list-style-type: none">• example of a result of an executed system command• content of file will be checked

For better understanding, open these files to see the content before running the 'test_loop'.

4.2 Run 'test_loop'

Please follow the instruction to run the 'test_loop':

1. Start a command line interface (e.g. shell) in the directory where you want to execute NabiCI and test whether php runs there from the command line.
2. Rename the '**ci.ini**' in the folder '**NabiCI**' into "**temp_ci.ini**".
More about '**ci.ini**' see [Configuration of a Build Loop](#)
3. Copy the 'test configuration file' called '**ci.ini**' from the subfolder '**Test_Files**' into '**NabiCI**'.

4. The copied test file 'ci.ini' contains settings which you now have to adapt to your system configuration.


Therefore open 'ci.ini' with a text editor. The task sections are not completed.


More about task section see [Task Section](#) .

It is necessary to adapt these settings to your environment. Complete it as described in the table below.

Note: before writing system commands in the configuration file it is recommended always to be sure that the commands can be executed correctly. Therefore start the command line interface in the directory where NabiCI is installed and execute manually all commands.

Note: all backslashes in the configuration file must be quoted with a backslash e.g.

not ok -> c:\NabiCI\backslashslash 

ok → c:\\NabiCI\\quote\\backslashslash 

Task	Param. name	Parameter value description
		All required files are test files in the folder NabiCI/Test_Files
[task_system_test]	call =	php write your full path to the file system_test.php for execution for example: call = php c:\NabiCI\Test_Files\system_test.php
[syntax_check_linewise_a]	call =	php write your full path to the file SyntaxCheckResultOK.php for execution
[syntax_check_linewise_b]	call =	php write your full path to the file SyntaxCheckResult.php for execution

[systemlist]	call_list = <i>Write your full path to the file system_list.txt to call</i>
[url_list]	url_list = <i>Write your full path to the file url_list.txt to call</i>

5. Save and close '**ci.ini**' after completion.
6. Open the '**system_list.txt**' from the subfolder '**Test_Files**'.

The file **system_list.txt** is called when running the loop. All commands in it will be executed one after other. The same applies for **url_list.txt**.

There is already one system command in the first line of this file. This is just a sample which you can leave how it is.

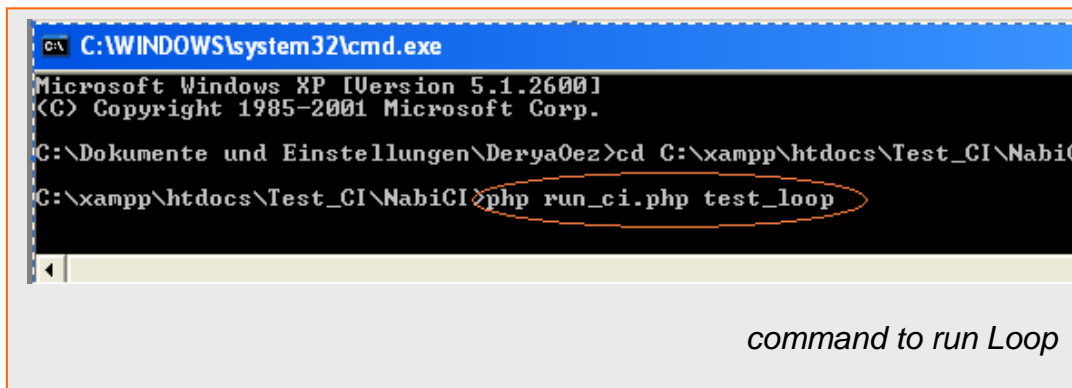
Write two more system commands in this file.

You get from the table below in line number two and three the description for the necessary command to write:

Line Nr.	system commands for ' system_list.txt '
1	php C:\\xampp\\htdocs\\Test_CI\\NabiCI\\Test_Files\\Demo.php
2	php write here the full path to SyntaxCheckResult.php
3	php write here the full path to SyntaxCheckResultOK.php

7. Save and close the file '**system_list.txt**' after finishing with writing the system commands.
8. All adaption is done now and test loop may be run.
9. Initiate the test loop with the following command in the command line interface.

Therefore do not forget to change to the directory '**NabiCI**' where the required file '**run_ci.php**' is.



```
C:\WINDOWS\system32\cmd.exe
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.
C:\Dokumente und Einstellungen\Derya0ez>cd C:\xampp\htdocs\Test_CI\NabiCI
C:\xampp\htdocs\Test_CI\NabiCI>php run_ci.php test_loop
```

command to run Loop

- For more general explanation relating to this command see [Running of the Build Loop](#) .
- In case of a build lock, you will be informed by the message „There is already a loop running” or “build.lock could not be deleted. Loops are still blocked” , therefore see [Build Lock](#) .

10. During the Continuous Integration processes the web pages are created.

All of them are saved in the subfolder of ‘NabiCI’ called ‘ci_web_pages’.

The folder where the files are produced must be accessible by your web server. Browse to the URL of e.g.

[“http://localhost/NabiCI/ci_web_pages/Result/Result.php”](http://localhost/NabiCI/ci_web_pages/Result/Result.php)

The results are only visible correctly when you open the file through your web server, because the files contain PHP code.

The created result page is saved in:

NabiCI/ci_web_pages/Result/Result.php

created web page to the result page

The current loop page is saved in:

NabiCI/ci_web_pages/Current/Current.php

created web page to the current page

- For more Information see [Visualization of the Continuous Integration Process](#) .
11. When you have finished with the test run of NabiCI you are ready to start with the configuration of your loop. It is recommended to do as follows:
- Rename or delete the '**ci.ini**' you used for the test run.
 - Rename the "**temp_ci.ini**" back to '**ci.ini**' and keep it in the folder '**NabiCI**'.
 - Use this changed '**ci.ini**' for configuration, see therefore [Configuration of a Build Loop](#)

5 Configuration of a Build Loop

For running a build loop you need to configure it in the file '**ci.ini**'.

A sample file '**ci.ini**' already exists in the folder '**NabiCI**' where you adapt it to your requirements.

Here is a sample structure for a NabiCI configuration file. The name of any key word must be written exactly as below, here highlighted fat!

Sample structure:

```
[Loop_name in lower case letters]  
tasks = task_1 task2 task_and_so_on  
deadline = value
```

see [Loop Section](#)

```
[task_1]  
task_mode = value  
name depends on task_mode = value  
expected_result = value
```

```
[task_2]  
task_mode = value  
name depends on task_mode = value  
expected_result = value
```

see [Task Section](#)

```
[task_and_so_on]  
task_mode = value  
name depends on task_mode = value  
expected_result = value
```

sample structure of ci.ini

5.1 Loop Section

Section names are always enclosed in box brackets.

First you need a section in order to setup the loop name, all task names to be run and the deadline. It is possible to have more than one section for several loops with several tasks to be run and several deadlines. Later you use the loop you want.

Note: the section name must be the loop name, which will be used to initiate a Continuous Integration loop. More explanation to run Loop see [Running of the Build Loop](#) .

Note: all following fat highlighted names and values of parameters must be written exactly as in 'ci.ini'

For running a Continuous Integration loop the file '**ci.ini**' has to contain following parameters.

Obligatory parameters:

[write here the Loop name in lower case letters]

tasks = write here all task names,

separate with a blank e.g. task1 task2 task_and_so_on

deadline = a deadline in minutes e.g. 30 in order to warn in case of a

Loop is not running as expected and the deadline is passed

ci.ini

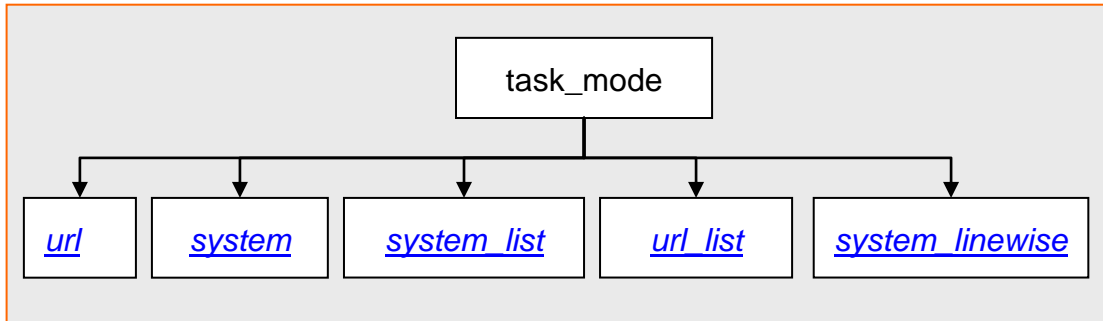
5.2 Task Section

The loop is composed of tasks. The tasks will call different kind of commands.

For each task to be run, which has been written in the [Loop Section](#) as part of the value of the "**tasks**" keyword, you need to create one section with the required parameters.

NabiCI provides five task modes. The execution of a command and checking of the result correctly depends on the choice of these options.

Overview of the task modes:



Note: all following fat highlighted names and values of parameters must be written in 'ci.ini'

5.2.1 url

For calling an URL and checking its result, the task defined as follows:

Obligatory parameters:

[write here the task name]

task_mode = url

url = *here you write the url you want for execution*

expected_result = *here you write the expected result*

ci.ini

Note: As the commands and urls are visible for everybody, make sure that you do not publish user names or passwords in your commands and urls

An executed task returns a result. This actual result is checked whether it contains the expected result. Is that the case, the task is 'OK' otherwise 'NOT OK' as you will see on the created web page.

5.2.2 system

For execution of a system command and checking its result the task is defined as follows:

Obligatory parameters:

```
[write here the Task name]  
task_mode = system  
call = here you write the system command for execution  
expected_result = here you write the expected result
```

ci.ini

5.2.3 system_list

If there is more than one system command to execute where the same result will be expected, it is possible to create therefore one section in the INI file and NabiCI will treat this task through the mode “system_list”. All commands from the called list will be executed. The task result is “OK”, if all the commands return the expected result, otherwise it is “NOT OK”.

Note: Create a .txt file and write the commands into it. There must be one command per line.

The task is to define as follows:

Obligatory parameters:

```
[write here the Task name]  
task_mode = system_list  
system_list = here you write the path to the file which contains the  
system commands e.g. c:/nabiCI/system_list.txt  
expected_result = here you write the expected result
```

ci.ini

5.2.4 url_list

The description to “url_list” is equal to the mode “system_list” [see [system_list](#)]. You just create a .txt file and write the URL´s into it. Per line must be one URL.

The task is to define as follows:

Obligatory parameters:

```
[write here the Task name]
task_mode = url_list
url_list = here you write the path to the file which contains URL
           e.g. c:/nabiCI/url_list.txt
expected_result = here you write the expected result
ci.ini
```

5.2.5 system_linewise

It is also possible to check the result of a task linewise. That means each line of a result will be checked to control if it is as expected. If one of them does not have the expected result, the task is “NOT OK”, if all the lines contain the expected result, the task is “OK”.

Note: this setting is not possible for URL, just for system commands

The task is defined as follows:

Obligatory parameters:

```
[write here the Task name]
task_mode = system_linewise
call = here you write the system command for execution
expected_result = here you write the expected result
ci.ini
```

6 Running of the Build Loop

For an automated execution of a Continuous Integration loop it is recommended to use a cronjob, which starts a loop repeatedly in the desired interval.

After configuration of the loop, run it with following system command:

ENTER INTO COMMAND LINE INTERFACE:

```
php run_ci.php write here the name of the Loop to run
```

Note: See [Build Lock](#) to get the cases where there is a blockade for running a new Loop

7 Build Lock

There are three different reasons, why a build lock can take place.

The first reason for a build lock is that a new loop can only be run when the previous loop is finished. The second reason is that a user desired to lock manually. The third reason is that the build lock still exists because the program has been interrupted.

7.1 build.lock

Normally a file '**build.lock**' is created in the folder '**NabiCI**' automatically by the program every time a loop is run and is deleted automatically when the loop is finished. This explains why only one loop can be run one at a time.

In case of unexpected events such as a program crash it is possible that the lock file could not be deleted. Then it is necessary to manually remove this file to unblock.

If the build.lock has not been deleted and you are trying to start a new loop, you will get the message "There is already a Loop running".

7.2 stop_ci.lock

In the case of a loop is running and it is desired to stop the current build process for example because of fixing an error before starting a new loop, it is necessary manually to create a file **'stop_ci.lock'** in the folder **'NabiCI'**.

As long as this file exists, **'build.lock'** can not be removed by the software. This means, that it is not possible to run a new loop.

After finish with fixing errors, the user has to manually delete the lock files and then he is free to run a new loop.

If the stop_ci.lock has not been deleted and you are trying to start a new loop, you will get the message "build.lock could not be deleted. Loops are still blocked."

8 Visualization of the Continuous Integration Process

The result of a build is visualized on web pages.


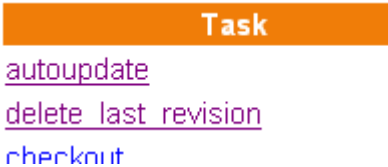
Here is a screenshot of the result from the 'nabidoo_loop':

No Error

[Current Loop](#) [made with Nabi](#)
[Refresh](#)

Last Completed Loop of Build revision 6674.

	Task	Begin	End	Duration	Result	Run Task
Loop	nabidoo_loop					
Begin	autoupdate	1-Nov-2009, 21:50:05	1-Nov-2009, 21:50:07	2 s	OK	autoupdate
End	delete_last_revision	1-Nov-2009, 21:50:07	1-Nov-2009, 21:50:09	2 s	OK	delete_last_revision
Duration	checkout	1-Nov-2009, 21:50:09	1-Nov-2009, 21:50:30	21 s	OK	checkout
Expiration Time	syntax_check	1-Nov-2009, 21:50:30	1-Nov-2009, 21:53:25	2 min 55 s	OK	syntax_check
	set_writeable	1-Nov-2009, 21:53:25	1-Nov-2009, 21:53:25	0 s	OK	set_writeable
	install_after_checkout	1-Nov-2009, 21:53:25	1-Nov-2009, 21:53:26	1 s	OK	install_after_checkout
	generate_test_list_after_checkout	1-Nov-2009, 21:53:26	1-Nov-2009, 21:53:27	1 s	OK	generate_test_list_after_checkout
	tests_after_checkout_1	1-Nov-2009, 21:53:27	1-Nov-2009, 21:53:28	1 s	OK	tests_after_checkout_1
	tests_after_checkout_2	1-Nov-2009, 21:53:28	1-Nov-2009, 21:53:28	0 s	OK	tests_after_checkout_2
	tests_after_checkout_3	1-Nov-2009, 21:53:28	1-Nov-2009, 21:53:32	4 s	OK	tests_after_checkout_3
	tests_after_checkout_4	1-Nov-2009, 21:53:33	1-Nov-2009, 21:53:37	4 s	OK	tests_after_checkout_4
	tests_after_checkout_5	1-Nov-2009, 21:53:37	1-Nov-2009, 21:53:37	0 s	OK	tests_after_checkout_5
	tests_after_checkout_6	1-Nov-2009, 21:53:37	1-Nov-2009, 21:53:37	0 s	OK	tests_after_checkout_6
	tests_after_checkout_7	1-Nov-2009, 21:53:37	1-Nov-2009, 21:53:42	5 s	OK	tests_after_checkout_7
	tests_after_checkout_8	1-Nov-2009, 21:53:42	1-Nov-2009, 21:53:43	1 s	OK	tests_after_checkout_8

short Explanation	Part of NabiCI web page																
<p>The left side of the web page offers:</p> <ul style="list-style-type: none"> • a Link to the current loop page • Loop name • Loop start time • Loop end time • Duration of the loop • Expiration time, if passed, there will be an error because the web page is no more up to date. • overview of all run tasks 	 <table border="1" data-bbox="775 680 1168 833"> <tr><td>Loop</td><td>nabidoo_loop</td></tr> <tr><td>Begin</td><td>1-Nov-2009, 21:50:05</td></tr> <tr><td>End</td><td>1-Nov-2009, 22:47:37</td></tr> <tr><td>Duration</td><td>57 min 32 s</td></tr> <tr><td>Expiration Time</td><td>2-Nov-2009, 01:47:37</td></tr> </table> <p>Overview All Tasks</p> <ul style="list-style-type: none"> autoupdate delete_last_revision checkout syntax_check set_writeable install_after_checkout generate_test_list_after_check tests_after_checkout obfuscate generate_test_list_after_obfus tests_after_obfuscate 	Loop	nabidoo_loop	Begin	1-Nov-2009, 21:50:05	End	1-Nov-2009, 22:47:37	Duration	57 min 32 s	Expiration Time	2-Nov-2009, 01:47:37						
Loop	nabidoo_loop																
Begin	1-Nov-2009, 21:50:05																
End	1-Nov-2009, 22:47:37																
Duration	57 min 32 s																
Expiration Time	2-Nov-2009, 01:47:37																
<p>For each task there is also a separate page showing its details. There is a Link in the column 'task':</p>																	
<p>The web page shows you the details of a task, which will be opened after click :</p>	<h3>Details On Task syntax_check</h3> <table border="1" data-bbox="730 1357 1254 1644"> <thead> <tr> <th>Task Attributes</th> <th>Task Attribute Values</th> </tr> </thead> <tbody> <tr><td>Startzeit</td><td>1-Nov-2009, 23:50:24</td></tr> <tr><td>Endzeit</td><td>1-Nov-2009, 23:53:17</td></tr> <tr><td>Duration</td><td>2 min 53 s</td></tr> <tr><td>Task_Result</td><td>success</td></tr> <tr><td>task_mode</td><td>system_linewise</td></tr> <tr><td>kommando</td><td>find ../.nabidoo -name "*.php" -exec php -i -f {} \;</td></tr> <tr><td>expected_result</td><td>No syntax errors detected in</td></tr> </tbody> </table>	Task Attributes	Task Attribute Values	Startzeit	1-Nov-2009, 23:50:24	Endzeit	1-Nov-2009, 23:53:17	Duration	2 min 53 s	Task_Result	success	task_mode	system_linewise	kommando	find ../.nabidoo -name "*.php" -exec php -i -f {} \;	expected_result	No syntax errors detected in
Task Attributes	Task Attribute Values																
Startzeit	1-Nov-2009, 23:50:24																
Endzeit	1-Nov-2009, 23:53:17																
Duration	2 min 53 s																
Task_Result	success																
task_mode	system_linewise																
kommando	find ../.nabidoo -name "*.php" -exec php -i -f {} \;																
expected_result	No syntax errors detected in																
<p>For each task you can also get the start time, the end time and the duration:</p>	<table border="1" data-bbox="699 1693 1334 1845"> <thead> <tr> <th>Begin</th> <th>End</th> <th>Duration</th> </tr> </thead> <tbody> <tr> <td>1-Nov-2009, 21:50:05</td> <td>1-Nov-2009, 21:50:07</td> <td>2 s</td> </tr> <tr> <td>1-Nov-2009, 21:50:07</td> <td>1-Nov-2009, 21:50:09</td> <td>2 s</td> </tr> <tr> <td>1-Nov-2009, 21:50:09</td> <td>1-Nov-2009, 21:50:30</td> <td>21 s</td> </tr> </tbody> </table>	Begin	End	Duration	1-Nov-2009, 21:50:05	1-Nov-2009, 21:50:07	2 s	1-Nov-2009, 21:50:07	1-Nov-2009, 21:50:09	2 s	1-Nov-2009, 21:50:09	1-Nov-2009, 21:50:30	21 s				
Begin	End	Duration															
1-Nov-2009, 21:50:05	1-Nov-2009, 21:50:07	2 s															
1-Nov-2009, 21:50:07	1-Nov-2009, 21:50:09	2 s															
1-Nov-2009, 21:50:09	1-Nov-2009, 21:50:30	21 s															

The column 'Result' indicates if the result was achieved or not.

Click on the result of the desired task to see the actual result.

The column 'Run Task' gives you the possibility to run a task separately with a click on the desired link. This can take place both on the result page or the current page.

It is strongly recommended to protect this functionality with a password, for more see [Access Protection for Run](#)

Result	Run Task
OK	autoupdate
OK	delete last revision
OK	checkout
OK	syntax check
OK	set writeable
OK	install after checkout
OK	generate test list after
OK	tests after checkout 1

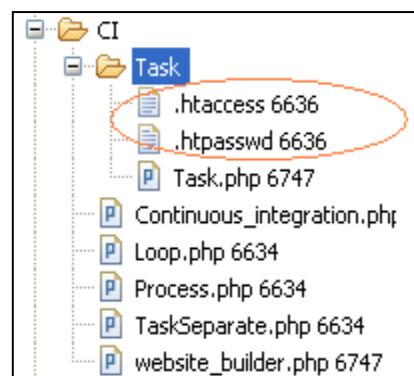
9 Access Protection for Run Task

It is possible to run a desired task separately from the web page.

Set up an access protection if required that such a task may be run exclusively by an authorized person.

Most common web servers offer the possibility to protect a web area with passwords. Therefore you prepare the both configuration file '**.htaccess**' and '**.htpasswd**' and put it in the folder 'NabiCi/CI/Task' as you can see on the screenshot.

After that, a task may only be run separately with entering the correct username and password.



10 General Notes

Webpage of 'Nabidoo':	http://www.nabidoo.de
Download 'NabiCI':	http://www.nabidoo.de/ci_framework.php
Contact:	axel.benz@hwr-berlin.de