TYPO3 Core - Bug #25657

Task Object State Is Not Saved

2010-04-29 16:07 - Jeff Segars

Status:	Rejected	**Start date:**	2010-04-29
Priority:	Should have	**Due date:**	
Assignee:	Francois Suter	**% Done:**	0%
Category:	scheduler	**Estimated time:**	0.00 hour
Target version:			
TYPO3 Version:		**Complexity:**	
PHP Version:		**Is Regression:**	
Tags:		**Sprint Focus:**	

Description

I've just started using the scheduler in 4.3 for the first time and overall its been a very pleasant experience. There is one area where I'm getting stuck, however.

In my task, I need to track a couple small variables that may change from one run to the next (ie. the last time an update was actually pulled from a remote server with changed data). I expected these would be saved when the task was serialized and available again on the next run, but instead I always have the default values.

This appears to be due to tx_scheduler->executeTask() which runs $task->save() prior to $task->execute. Is this the intended behavior? If so, I guess the best option is to just write my data somewhere else for saving rather than expecting my task to be persistent.

(issue imported from #M14251)

Related issues:

Related to TYPO3 Core - Task #27552: Update manual for TYPO3 4.6	**Closed**	**2011-06-19**

History

#1 - 2010-04-29 16:08 - Jeff Segars

Francois's response....

The task is saved prior to actual execution because that triggers the calculation of the next execution date. Looking at this in retrospect doesn't make it look very clever. We would probably need 2 different methods: one that just saves the task and one that calculates next execution time and saves. That's a bit too late to have such a change in 4.4, but you may want to open a bug tracker issue so that we don't forget.

#2 - 2010-07-29 03:07 - Christian Kuhn

I'm unsure if it's a good idea to store such values in the task object in the first place, the registry might be a better choice for such things.

Imagine you'd like to show / use / manipulate those values at some other point in TYPO3: You'd have to de-serialize the task again to get the current state. If you store it in the registry instead, it is available in a transparent way wherever you want.

The only reason to store the serialized task instance in db is (afaik) to realize multiple instances of one task class with different settings (execution time, additional fields, ...). Task code which depends on other logic (external magic, return value of a previous run, ...) do not really fit and belong to this and should be handled separately.

#3 - 2010-11-29 23:24 - Christian Kuhn

I've changed by mind in contrast to my previous post.

I think it **is** a good idea to store some values in a task instance itself, and to update and persist them if needed. #16360 is a perfect use case: It stores and updates it's current pointer to a database row where the next task run should start from. This is done by just calling $this->save() at the end of execute(), which serializes and persists the task instance again.

@Jeff / Francois Suter: Do you think this is sufficient? We could probably easily add a call to save() after execution of a task within the scheduler, but this will change behaviour and **might** introduce hard to debug issues to existing (third party) tasks. What do you think?

#4 - 2011-06-01 13:54 - Mr. Hudson

Patch set 1 of change I262584cdb4a896ea3b86be7c40fe184f2c55b724 has been pushed to the review server.
It is available at http://review.typo3.org/2479

#5 - 2011-06-01 14:02 - Andy Grunwald

- Target version deleted (0)

I`ve pushed a patch set to gerrit which changes the following:

- Adds a new member variable to tx_scheduler_Task: boolean $saveTaskObjectState
- Adds three new methods to tx_scheduler_Task: enableSavingTheTaskObjectState, disableSavingTheTaskObjectState, isSavingTheTaskObjectStateEnabled
- Refactored tx_scheduler::saveTask and adds a new parameter to controll the calculation of the next executiontime
- Adds a new condition after task execution which checks if the saving is enabled, if yes, the task object will be saved
- Adds a new example task 'tx_scheduler_SaveTaskObjectStateTask' to show the functionality

This Bugfix / Feature (in my opinion it is a feature, but i have committed it as bugfix, because it is a ticket in the bug tracker of scheduler) **must be enabled by the developer**. On default it is disabled.
The developer must take care of this functionality in his task.
With this there are no side effects on currently existing tasks.

This Bugfix (what is it? Bug? Feature?) could be easily backported to TYPO3 4.5 or 4.4, too.

#6 - 2011-06-01 14:10 - Ingo Renner

Christian Kuhn wrote:

> I'm unsure if it's a good idea to store such values in the task object in the first place, the registry might be a better choice for such things.
>
> Imagine you'd like to show / use / manipulate those values at some other point in TYPO3: You'd have to de-serialize the task again to get the current state. If you store it in the registry instead, it is available in a transparent way wherever you want.

I agree with Christian, just use the registry, it's the perfect place for such things (last time data was fetched and so on...). We also do this with some EXT:solr scheduler tasks...

#7 - 2011-06-24 22:54 - Francois Suter

- Status changed from New to Rejected

- Assignee set to Francois Suter

It happened that I discussed this issue with Xavier, because he stumbled on it too and the solution is really simple: just call

```
$this->save()
```

inside your task. It does no harm to the next execution date, because a new execution date is calculated only if the current one is in the past. Since the next execution has just been calculated, it will be in the future and thus not calculated again (unless you save your task at the end of the execution and it has run for longer than its frequency, but this probably means that you have a problem anyway).

I'll add a note about saving from within the task in the Scheduler's manual. IMO this issue is solved simply be improved documentation.

#8 - 2011-07-04 16:56 - Andy Grunwald

Okay. Finally you are right.
I`ve closed this patch and will wait for the updated ducomentation.

#9 - 2012-12-15 16:20 - Jonas Götze

This might save others some time:
If you save the task twice in your code you need to restore the scheduler reference after saving:
$this->save();
$this->setScheduler();
as this gets unset while saving, but is actually used when saving again and will lead to a Fatal Error if its not present.

Apart from that you should unset large properties of your Task class before saving - in my case I needed to unset $this->objectManager and $this->persistenceManager which I used. Without unsetting them the task object gets destroyed - because its too large for DB serialization I guess.

#10 - 2013-12-10 01:24 - Michael Stucki

- Category set to scheduler

#11 - 2013-12-10 01:27 - Michael Stucki

- Project changed from Scheduler to TYPO3 Core

- Category changed from scheduler to scheduler