# Core - Feature # 51731

| | | | |
|---|---|---|---|
| **Status:** | On Hold | **Priority:** | Should have |
| **Author:** | Thorsten Kahler | **Category:** | Performance |
| **Created:** | 2013-09-04 | **Assigned To:** | |
| **Updated:** | 2015-01-18 | **Due date:** | |
| **PHP Version:** | | | |
| **Complexity:** | nightmare | | |
| **Sprint Focus:** | | | |
| **Subject:** | Store sessions outside DB | | |

**Description**

On high traffic sites it's sometimes necessary to store session records outside the main database because of a high rate of update queries on the FE session tables. This issue becomes even more pressing when using DB replication or remote database servers.

I suggest to add a session storage service so it can be implemented to use different storage backends. These storage backend implementations can be based on caching backends or totally unrelated to the core.

**Related issues:**

| | | |
|---|---|---|
| related to Core - Bug # 16302: alt_doc.php does BE session verification twice | **Closed** | **2006-06-28** |
| related to Core - Bug # 53598: Select/Delete fe_sessions twice per request | **Resolved** | **2013-11-13** |

## Associated revisions

**Revision c50ee9f3 - 2013-09-06 20:24 - Thorsten Kahler**

[WIP][FEATURE] Classic session storage service

- !!! known bug: T3_SERVICES prohibits multiple instances of services so sessions in frontend are broken because FE and BE sessions are used in same request
- implementation of SessionStorageInterface based on traditional code from class AbstractUserAuthentication
- duplicated code could be removed from AbstractUserAuthentication later

Change-Id: Idad0fd2b3368a4c7ef5f163264becb1c9783c2b6
Resolves: #51731
Releases: 6.2

**Revision 97d077da - 2013-09-24 12:38 - Thorsten Kahler**

[WIP][FEATURE] Classic session storage service

- !!! known bug: T3_SERVICES prohibits multiple instances of services so sessions in frontend are broken because FE and BE sessions are used in same request
- implementation of SessionStorageInterface based on traditional code from class AbstractUserAuthentication
- duplicated code could be removed from AbstractUserAuthentication later

Change-Id: Idad0fd2b3368a4c7ef5f163264becb1c9783c2b6
Resolves: #51731
Releases: 6.2

## History

**#1 - 2013-09-04 17:44 - Thorsten Kahler**

*- % Done changed from 0 to 10*

Suggested implementation in sandbox sandbox/thorsten/session-api

**#2 - 2013-09-04 20:37 - Ernesto Baschny**

*- Status changed from New to Accepted*

Cool stuff, Thorsten, you already seem to have made some nice progress!

Are you planning to move the whole database storage as it is now into a service also? That would enable you to get rid of all those "if (sessionStorage)" you have currently in the code.

Looking at how Flow Session handling currently is implemented, it's a mix of session storage and session keeping (resume, handle the ID, the cookie etc). Separating the "Storage" in it's own concearn sounds like a good plan.

Maybe you could also take a look at Symfony for some inspiration on the interface, as they provide some neat additions like Bags and some other helper-methods.

At the end, to be able to test this for review it would be great if you could provide some implementation example as an extension.

**#3 - 2013-09-04 20:38 - Ernesto Baschny**

And the last question: Will you be able to tackle the first implementation before feature freeze (15th of October)? It would be helpful if you are able to join our Code Sprint in October so you can also discuss the API with the guys (and get some "live-reviews"): Enhances the chance of it being included in the release. ;)

**#4 - 2013-09-04 23:08 - Thorsten Kahler**

Ernesto Baschny wrote:

> *Are you planning to move the whole database storage as it is now into a service also? That would enable you to get rid of all those "if (sessionStorage)" you have currently in the code.*

I was thinking about using the current implementation as a DB session storage service. Didn't know if it's clever to remove the existing  and working code in 6.2 LTS. But if there's an agreement to declare the API stable moving the whole implementation to a storage service would be the next logical step.

> *Looking at how Flow Session handling currently is implemented, it's a mix of session storage and session keeping (resume, handle the ID, the cookie etc). Separating the "Storage" in it's own concearn sounds like a good plan.*

There's room for improvement of the whole session handling but IMO not before release of the LTS version. But the storage part should be stable enough to be used later on.

**#5 - 2013-09-04 23:14 - Thorsten Kahler**

*- File dkd_redis_sessions.zip added*

Ernesto Baschny wrote:

> *At the end, to be able to test this for review it would be great if you could provide some implementation example as an extension.*

I've developed a proof of concept implementation in parallel to the API, of course. It uses Redis via TYPO3s Redis caching backend as a session storage backend. The extension isn't finished yet but already works (see attachment:dkd_redis_sessions.zip)

**#6 - 2013-09-04 23:22 - Thorsten Kahler**

Ernesto Baschny wrote:

> *And the last question: Will you be able to tackle the first implementation before feature freeze (15th of October)? It would be helpful if you are able to join our Code Sprint in October so you can also discuss the API with the guys (and get some "live-reviews"): Enhances the chance of it being included in the release. ;)*

To facilitate reviews it's probably useful to have "classic" implementation (using the current code) and maybe an implementation using the DB caching backend. Let's see how far I can get with that.

**#7 - 2013-09-06 20:30 - Thorsten Kahler**
*- Status changed from Accepted to Resolved*
*- % Done changed from 10 to 100*

Applied in changeset commit:c50ee9f32186eb71439cad2e80b6198b03d9a461.

[Edit: It seems Gerrit is a bit too avid ;-)]

**#8 - 2013-09-06 20:50 - Thorsten Kahler**
*- Status changed from Resolved to Accepted*
*- % Done changed from 100 to 10*

TYPO3s services don't work too well as a way to select and instantiate a storage backend because GeneralUtitility::makeInstanceService() implements a pseudo-singleton. That's not a problem in the backend, but in the frontend BE user sessions are checked besides the FE user sessions. But due to the singleton characteristic of the session storage service init() is only called once and you can't switch between FE and BE session storages without an enormous effort.

In short: it's possible to instantiate *different* instances of the storage service for FE and BE, but not *multiple* which is required in FE.
If we want to stick with the services approach I could imagine different ways:
1. Implement different classes for BE and FE session storage; these can be stubs which inherit all functionality from a common parent class and only serve to satisfy the pseudo-singleton behaviour

   This seems to be the leanest approach, yet it still means some overhead in terms of empty classes:

```
abstract class ClassicStorage extends \TYPO3\CMS\Core\Service\AbstractService implements StorageInterface {
// implementation here
}
```

```
class ClassicStorageFrontend extends ClassicStorage {
// no code here or method init() at most
}
class ClassicStorageBackend extends ClassicStorage {
// no code here or method init() at most
}
```

2. Make the service a factory that only creates a storage object of the requested type (FE, BE)
   Maybe the factory class could be generic and such part of the core. Then this wouldn't mean much of a change to the current design of the *implementations*. Concerning the architecture this is just one pattern too much to solve the underlying issue properly.

```
class AbstractUserAuthentication {
public function start() {
 $storageFactory = GeneralUtility::makeInstanceService('sessionStorage');
 $this->sessionStorage = $storageFactory->getInstance($this->session_table == 'fe_sessions' ? 'frontend' : 'backend');
 // ...
}
```

3. Change GeneralUtitility::makeInstanceService() to use $serviceType and serviceSubtype besides className to identify service instances
   This will break some service implementations for sure so IMHO it's not an option. Nevertheless the current behaviour could be achieved by marking the service class as singleton.

```
class AuthenticationService extends \TYPO3\CMS\Sv\AbstractAuthenticationService implements \TYPO3\CMS\Core\SingletonInterface
{
// ...
}
```

4. Tell the storage which kind of session is meant
   This is as ugly as it gets, regardless of the implementation:

```
class AbstractUserAuthentication {
// ...
public function fetchUserSession() {
 $session = $this->sessionStorage->get($this->id, $this->session_table);
 // ...
}
// ...
}
class SessionStorage {
 public function get($identifier, $type) {
   if ($type === 'fe_sessions'} {
     // ...
   } else {
     // ...
   }
 }
// ...
}
```

```
class AbstractUserAuthentication {
// ...
public function fetchUserSession() {
 $this->sessionStorage->injectAuthentication($this);
 $session = $this->sessionStorage->get($this->id);
 // ...
}
// ...
}
class SessionStorage {
  public function injectAuthentication(AbstractUserAuthentication $auth) {
    $this->authentication = $authentication;
  }
  public function get($identifier) {
    if ($this->authentication->session_table === 'fe_sessions'} {
      // ...
    } else {
      // ...
    }
  }
// ...
}
```

Any other opinions or ideas?

**#9 - 2013-09-24 13:30 - Thorsten Kahler**

*- Status changed from Accepted to Resolved*

*- % Done changed from 10 to 100*

Applied in changeset commit:97d077daafaa83d38d01eb1b263bb6e3f3661d82.

**#10 - 2013-09-24 13:55 - Christian Kuhn**

*- Status changed from Resolved to New*

Merge was to sandbox only, patch is not in master, yet, reopened.

**#11 - 2013-09-24 14:00 - Thorsten Kahler**

Christian Kuhn wrote:

> *Merge was to sandbox only, patch is not in master, yet, reopened.*

Thanks :)

**#12 - 2013-10-02 11:47 - Thorsten Kahler**

*- Status changed from New to Accepted*

*- Assigned To set to Thorsten Kahler*

**#13 - 2013-10-14 22:44 - Thorsten Kahler**

*- Status changed from Accepted to On Hold*

*- Target version deleted (6.2.0)*

*- % Done changed from 100 to 10*

*- Complexity changed from hard to nightmare*

To get this cleanly done needs much more work and some major changes in AbstractUserAuthentication and its subclasses. Thus this feature is delayed to the next release after 6.2.

**#14 - 2015-01-18 23:05 - Mathias Schreiber**

*- Assigned To deleted (Thorsten Kahler)*

*- Target version set to 8*

## Files

| | | | |
|---|---|---|---|
| dkd_redis_sessions.zip | 3.4 kB | 2013-09-04 | Thorsten Kahler |