

Contents

- [Using Clone\(\)](#)
 - [Caveat on cloned objects](#)
 - [Why would you use Clone?](#)
 - [Clone and lazy="true"](#)

Using Clone()

You are able to make a deep clone of every generated TransferObject, by calling 'clone()' on it.

For example:

```
<cfscript>
    user = getTransfer().get("user.User", 1);
    cloneUser = user.clone();
</cfscript>
```

This cloned object is outside of any caching that is currently in use within Transfer, which means that any changes that are done to this object, and are not saved, are only particular to the request that they are currently in.

It should be noted that [Transfer.save\(\)](#) and [Transfer.update\(\)](#) on a clone object will update the object currently in cache to the state of the saved object

Caveat on cloned objects

Cloned objects should not be added as composites to objects that are not clones. This can cause indeterminate results.

Why would you use Clone?

Cloning is very useful when you want to make changes to an object, but you don't want those changes reflected in the cache until you save the object back to the database.

The primary example of this is in validation of an object. Say for example, you get() a User from the database, and you are populating it with form data, you may not know if the data in the User object is valid or not, until you call User.validate(), but if you make changes to the User object that just came out of cache, it is possible that invalid data would be visible across your application.

To avoid this issue, you can create a clone() of your User, by calling:

```
<cfscript>
```

```
    user = getTransfer().get("user.User", 1);
    cloneUser = user.clone();
</cfscript>
```

This creates a new User which is a clone of the original. Changes can be made to the clone quite happily, without having to worry about them being visible across your cache. So that means you can call `.validate()` on the `cloneUser`, and check to see if the data is valid before saving it.

For example:


```
<cfscript>
    user = getTransfer().get("user.User", 1);
    cloneUser = user.clone();

    cloneUser.setName("");
    isValid = cloneUser.validate();
</cfscript>
```

When the `cloneUser` is saved, it's state is synchronised with the original User that came from the cache, thus ensuring that only valid data is visible across the cache.

Clone and lazy="true"

It should be noted that if you `clone()` an object, it will not automatically duplicate any relationships that are set to `lazy="true"`. Those are lazy loaded when requested on the clone. In this way you can ensure how many objects are duplicated when cloning a particular object.

 Categories:

- [Cache](#)
- [Clone](#)