

### Diskussionsseiten

- 1) <http://www.theserverside.de/singleton-pattern-in-java/>
- 2) <http://www.cs.umd.edu/~pugh/java/memoryModel/DoubleCheckedLocking.html>
- 3) <https://funthomas424242.wordpress.com/2012/02/19/cc-singleton-korrekt-implementieren/>

### Singleton - unsichere Initialisierung

Diese Implementierung hat folgende Besonderheiten:

[1] Theoretisch funktioniert sie nicht - in der Praxis taugt sie aber als Singleton (siehe dazu folgende Diskussion)

[2] Man darf sich hier bei Zugriff nicht auf ein korrekt initialisiertes Singleton verlassen, da im konkreten Fall auf Grund des Memory Modelles keine Annahmen darüber zugelassen sind wann die Felder des erzeugten Singleton wirklich alle korrekt befüllt und für jeden Thread konsistent lesbar sind.

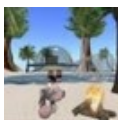
```
public class Resource{
    private static Resource singleton;
public static Resource create Instance(){
    if( singleton ==null){ //erster Test
        synchronized (Resource.class){
            if( singleton ==null){ //zweiter Test
                singleton=new Resource();
            } //endif
        } // end sync
    } // endif
    return singleton;
}
}
```

Details zum Problem unter: <http://www.cs.umd.edu/~pugh/java/memoryModel/DoubleCheckedLocking.html>

### Singleton - korrekte Initialisierung

```
private static volatile Resource resource;
private static volatile Resource singleton;
public static Resource create Instance(){
    if( resource ==null){
        synchronized (Resource.class){
            if( singleton ==null){
                singleton=new Resource();
            }
        }
        resource= singleton;
    }
    return resource;
}
```

Ähnliche Lösungen werden auch hier diskutiert: <http://www.angelikalanger.com/Articles/EffectiveJava/41.JMM-DoubleCheck/41.JMM-DoubleCheck.html>



By **Huluvu424242**  
(FunThomas424242)

Not published yet.  
Last updated 12th May, 2016.  
Page 1 of 1.

Sponsored by **ApolloPad.com**  
Everyone has a novel in them. Finish  
Yours!  
<https://apollopad.com>