JUG SAXONY CAMP 2017

# Caching-Verfahren für Java Enterprise Anwendungen

Erik Rohkohl
HTWK Leipzig & itemis AG

# Motivation

```
public class CacheDemo {
    public int someExpensiveCalculation(int argOne, String argTwo) {
        return new Random().nextInt();
  }
}
```

# Motivation

```
public class CacheDemo {

    private Map<Integer, Integer> cache;

    public CacheDemo() {
        cache = new HashMap<Integer, Integer>();
    }

    public int someExpensiveCalculation(int argOne, String argTwo) {
        int cacheValue;
        int cacheKey = argOne + argTwo.hashCode();

        if (cache.containsKey(cacheKey)) {
            cacheValue = cache.get(cacheKey);
        } else {
            cacheValue = new Random().nextInt();
            cache.put(cacheKey, cacheValue);
        }
        return cacheValue;
    }
}
```

# JCache API

- JSR-107
- Standardisierte Lösung für JEE- und SE-Anwendungen Einträge in einem Cache zu verwalten
- Setzt Caching-Verfahren mittels Interceptoren aus dem *Context and Dependency Injection* Standard um
- Steuert Verhalten des Caches zur Laufzeit mithilfe von Annotationen (@CacheResult)
- Unterstützt Speicherstrategien (store-by-value u. -reference) und Ersetzungsstrategien (LRU, FIFO, …)
- Kein Teil des Java-Enterprise-Stacks
- WildFly-Implementierung: *Infinispan*

# Caching-Verfahren mit JCache API und Infinispan-Interceptor

```
public class Service {

    @CacheResult
    public int execute(int a, int b){
        return (int)(Math.random()*1000);
    }
}
```

# Caching-Verfahren mit JCache API und Infinispan-Interceptor

```
<beans bean-discovery-mode="all">
    <interceptors>
        <class>org.infinispan.jcache.annotation.CacheResultInterceptor</class>
    </interceptors>
</beans>
```

# Caching-Verfahren mit JCache API und Infinispan-Interceptor

http://localhost:8080//jug/saxony/camp/jcache/one/Servlet

http://localhost:8080//jug/saxony/camp/jcache/two/Servlet

# Caching-Verfahren mit JCache API und Infinispan-Interceptor

- Probleme:
  - Kein Datenaustausch zwischen Java Enterprise Anwendungen
  - Cache nicht stabil gegenüber einem Undeployment der Anwendung
  - Keine Cache-Konfiguration möglich
  - Speicherbereich liegt nicht unter Kontrolle des Application Servers
- Lösung:
  - Caching-Verfahren mit benutzerdefiniertem Interceptor für @CacheResult-Annotation

# Caching-Verfahren mit JCache API und benutzerdefiniertem Interceptor

- eigene Implementierung eines *CacheResultInterceptors* mithilfe des CDI-Standards
- Caching-Verfahren kompatibel mit JCache API
- Infinispan als eingebettetes Caching-Framework nutzen
- Caches deklarativ konfigurieren innerhalb des Application Servers

# Caching-Verfahren mit JCache API und benutzerdefiniertem Interceptor

```
<subsystem xmlns="urn:jboss:domain:infinispan:4.0">
    <cache-container name="myCache" default-cache="default">
        <local-cache name="default"
            jndi-name="java:jboss/infinispan/container/myCache/default"
            statistics-enabled="true">
            <eviction strategy="LRU" max-entries="100" />
            <expiration interval="1000" lifespan="300000" max-idle="300000" />
        </local-cache>
    </cache-container>
</subsystem>
```

# Caching-Verfahren mit JCache API und benutzerdefiniertem Interceptor

```
public class Service {

    @CacheResult(cacheName="myCache")
    public int execute(int a, int b){
        return (int)(Math.random()*1000);
    }
}
```

# Caching-Verfahren mit JCache API und benutzerdefiniertem Interceptor

```
@Interceptor
@Dependent
@CacheResult
public class CacheResultInterceptorCustom {

    private EmbeddedCacheManager cacheContainer;
    private org.infinispan.Cache<Integer, Object> cache;

    @AroundInvoke
    public Object managedTransaction(InvocationContext ctx) throws Exception {

        return null;
    }
}
```

# Caching-Verfahren mit JCache API und benutzerdefiniertem Interceptor

```
@AroundInvoke
public Object managedTransaction(InvocationContext ctx) throws Exception {

    CacheResult annotation = ctx.getMethod().getAnnotation(CacheResult.class);
    String cacheName = annotation.cacheName();
    cacheContainer = (EmbeddedCacheManager) new InitialContext()
            .lookup("java:jboss/infinispan/container/" + cacheName);

    cache = cacheContainer.getCache();
    int cacheKey = Objects.hash(ctx.getParameters());
    Object result = cache.get(cacheKey);

    if (result == null || annotation.skipGet()) {
        result = ctx.proceed();
        cache.put(cacheKey, result);
    }
    return result;
}
```

# Caching-Verfahren mit JCache API und benutzerdefiniertem Interceptor

```
<beans bean-discovery-mode="all">
    <interceptors>
        <class>
            jug.saxony.camp.infinispan.embedded.jar.interceptor.CacheResultInterceptorCustom
        </class>
    </interceptors>
</beans>
```

# Caching-Verfahren mit JCache API und benutzerdefiniertem Interceptor

```
<application>
    <resource-env-ref>
        <lookup-name>java:jboss/infinispan/container/myCache/default</lookup-name>
        <resource-env-ref-name>myCache/default</resource-env-ref-name>
        <resource-env-ref-type>org.infinispan.Cache</resource-env-ref-type>
    </resource-env-ref>
</application>
```

# Caching-Verfahren mit JCache API und benutzerdefiniertem Interceptor

```
@MetaInfServices
public class BeanDiscoveryObserver implements Extension{

    public void registerCacheResultAnnotation(@Observes BeforeBeanDiscovery event) {
          event.addInterceptorBinding(CacheResult.class);
          System.out.println("CacheResult Annotation registered.");
    }
}
```

# Caching-Verfahren mit JCache API und benutzerdefiniertem Interceptor

http://localhost:8080//jug/saxony/camp/infinispan/embedded/one/Servlet

http://localhost:8080//jug/saxony/camp/infinispan/embedded/two/Servlet

# GitLab

- Alle Code-Beispiele als Maven-Projekte
- WildFly Application Server
- Folien
- Bachelorarbeit zum Thema:

*Caching-Verfahren für Java-Enterprise Anwendungen auf Grundlage des JSR-107*

https://gitlab.imn.htwk-leipzig.de/erohkohl/JUG_SAXONY_CAMP_2017