Dokumentation AOP Beispiel

# Cache.java

|  |
| --- |
| **package** ch.swaechter.eaf.cache;  **import** java.util.ArrayList; **import** java.util.HashMap; **import** java.util.List; **import** java.util.Map;  **public class** Cache {   **private** Map<Long, Object> **elements**;   **public** Cache() {  **this**.**elements** = **new** HashMap();  }   **public void** addObjectToCache(Object object, **long** key) {  **this**.**elements**.put(key, object);  }   **public** Object getObjectFromCache(**long** key) {  **return this**.**elements**.get(key);  }   **public** List<Object> getAll() {  **return new** ArrayList<>(**elements**.values());  }   **public void** removeObject(Long id) {  **this**.**elements**.remove(id);  }   **public void** removeAll() {  **this**.**elements**.clear();  }   **public int** getNumberOfObjects() {  **return elements**.size();  } } |

# CacheManager.java

|  |
| --- |
| **package** ch.swaechter.eaf.cache;  **import** org.springframework.stereotype.Service;  **import** java.util.HashMap; **import** java.util.Map;  @Service **public class** CacheManager {   **private** Map<Class, Cache> **elements**;   **public** CacheManager() {  **this**.**elements** = **new** HashMap<>();  }   **public** Cache createCacheForClass(Class clazz) {  **if** (**elements**.get(clazz) == **null**) {  Cache cache = **new** Cache();  **elements**.put(clazz, cache);  **return** cache;  } **else** {  **return elements**.get(clazz);  }  }   **public** Cache getCacheByClass(Class clazz) {  **return elements**.get(clazz);  }   **public int** getNumberOfCaches() {  **return elements**.size();  }   **public int** getTotalNumberOfCachedObjects() {  **int** result = 0;  **for** (Map.Entry entry : **elements**.entrySet()) {  Cache cache = (Cache) entry.getValue();  result += cache.getNumberOfObjects();  }  **return** result;  } } |

# AopApplication.java

|  |
| --- |
| **package** ch.swaechter.eaf;  **import** ch.swaechter.eaf.user.User; **import** ch.swaechter.eaf.user.UserRepository; **import** org.springframework.beans.factory.annotation.Autowired; **import** org.springframework.boot.CommandLineRunner; **import** org.springframework.boot.SpringApplication; **import** org.springframework.boot.autoconfigure.SpringBootApplication; **import** org.springframework.stereotype.Service;  @SpringBootApplication **public class** AopApplication {   **public static void** main(String[] args) {  SpringApplication.*run*(AopApplication.**class**, args);  }   @Service  **public class** UserRunner **implements** CommandLineRunner {   @Autowired  **private** UserRepository **userRepository**;   @Override  **public void** run(String... args) {  **for** (**int** i = 0; i < 10; i++) {  **userRepository**.save(**new** User(**"User "** + i));  }  }  } } |

# CacheAspect.java (Ein Aspekt pro Methode)

|  |
| --- |
| **package** ch.swaechter.eaf.aop;  **import** ch.swaechter.eaf.cache.Cache; **import** ch.swaechter.eaf.cache.CacheManager; **import** ch.swaechter.eaf.user.User; **import** org.aspectj.lang.ProceedingJoinPoint; **import** org.aspectj.lang.annotation.Around; **import** org.aspectj.lang.annotation.Aspect; **import** org.springframework.beans.factory.annotation.Autowired; **import** org.springframework.stereotype.Component;  **import** java.util.List; **import** java.util.Optional;  @Aspect @Component **public class** CacheAspect {   @Autowired  **private** CacheManager **cacheManager**;   @Around(**"execution(\* \*..eaf.user.UserService.getUsers())"**)  **public** Object interceptGetUsers(ProceedingJoinPoint point) **throws** Throwable {  Cache cache = getOrCreateCache(User.**class**);  **if** (cache.getAll().isEmpty()) {  List<Object> objects = (List<Object>) point.proceed();  **for** (Object object : objects) {  User user = (User) object;  cache.addObjectToCache(user, user.getId());  }  }  **return** cache.getAll();  }   @Around(**"execution(\* \*..eaf.user.UserService.getUser(..)) && args(id)"**)  **public** Object interceptGetUser(ProceedingJoinPoint point, Long id) **throws** Throwable {  Cache cache = getOrCreateCache(User.**class**);  **if** (cache.getObjectFromCache(id) == **null**) {  Optional<User> user = (Optional<User>) point.proceed();  **if** (user.isPresent()) {  cache.addObjectToCache(user.get(), id);  }  }  **return** Optional.*of*(cache.getObjectFromCache(id));  }   @Around(**"execution(\* \*..eaf.user.UserService.saveUser(..)) || execution(\* \*..eaf.user.UserService.updateUser(..))"**)  **public** Object interceptUpdateUser(ProceedingJoinPoint point) **throws** Throwable {  Cache cache = getOrCreateCache(User.**class**);  User updatedUser = (User) point.proceed();  cache.addObjectToCache(updatedUser, updatedUser.getId());  **return** updatedUser;  }   @Around(**"execution(\* \*..eaf.user.UserService.deleteUser(..)) && args(id)"**)  **public void** interceptDeleteUser(ProceedingJoinPoint point, Long id) **throws** Throwable {  Cache cache = getOrCreateCache(User.**class**);  point.proceed();  cache.removeObject(id);  }   **private** Cache getOrCreateCache(Class targetClass) {  Cache cache = **cacheManager**.getCacheByClass(targetClass);  **return** cache != **null** ? cache : **cacheManager**.createCacheForClass(targetClass);  } } |

# OldClassCacheAspect.java (Ein Aspekt mit Methodenunterscheidung)

|  |
| --- |
| **package** ch.swaechter.eaf.aop;  **import** ch.swaechter.eaf.cache.Cache; **import** ch.swaechter.eaf.cache.CacheManager; **import** ch.swaechter.eaf.user.User; **import** org.aspectj.lang.ProceedingJoinPoint; **import** org.aspectj.lang.annotation.Aspect; **import** org.aspectj.lang.reflect.MethodSignature; **import** org.springframework.beans.factory.annotation.Autowired; **import** org.springframework.stereotype.Component;  **import** java.lang.reflect.Method; **import** java.util.List;  @Aspect @Component **public class** OldClassCacheAspect {   @Autowired  **private** CacheManager **cacheManager**;   *//@Around("execution(\* \*..eaf.user.UserController..\*(..))")* **public** Object interceptMethods(ProceedingJoinPoint point) **throws** Throwable {  MethodSignature signature = (MethodSignature) point.getSignature();  Method method = signature.getMethod();  System.***out***.println(**"Method called: "** + point.getTarget().getClass() + **"."** + method.getName());  **switch** (method.getName()) {  **case "getUsers"**:  **return** getUsers(point);  **case "getUser"**:  **return** point.proceed();  *//* ***TODO: For all other methods* default**:  System.***out***.println(**"Mr. Luthiger...we didn't expect that :("**);  **throw new** IllegalStateException(**"Thanks for reusing old MSP exams!"**);  }  }   **private** Object getUsers(ProceedingJoinPoint point) **throws** Throwable {  Class targetClass = point.getTarget().getClass();  Cache cache = **cacheManager**.getCacheByClass(targetClass);  **if** (cache == **null**) {  cache = **cacheManager**.createCacheForClass(targetClass);  }   **if** (cache.getAll().isEmpty()) {  List<Object> objects = (List<Object>) point.proceed();  **for** (Object object : objects) {  User user = (User) object;  cache.addObjectToCache(user, user.getId());  }  }  **return** cache.getAll();  } } |