Working with JCache in the “Real World”

Grant Little

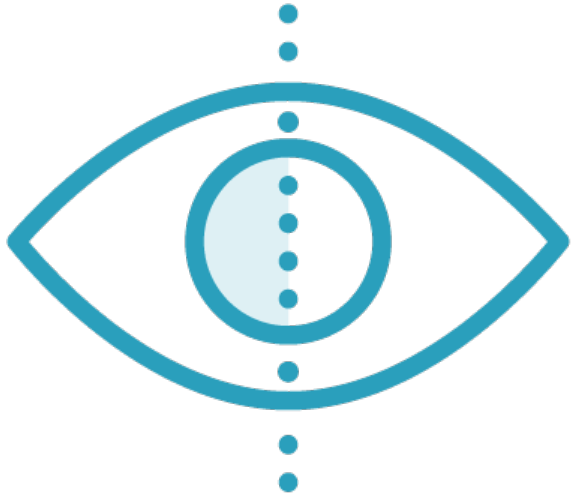
www.grantlittle.me

Overview

- Basic Spring Boot Application
- Basic Cache Operations
- Considering Consistency
- Changing to another JCache Provider

Demo Spring Boot Application

Pluralsight Spring Boot Courses



Creating Your First Spring Boot Application

Dan Bunker

Spring Boot: Efficient Development, Configuration and Deployment

Dustin Schultz

Basic Operations

```
Map<Integer, Customer> map = new HashMap<>();
map.put(1, customer1);
map.put(2, customer2);
cache.putAll(map);
```

Bulk Put – putAll()

```
Set<Integer> keys = new HashSet<>();
keys.add(1); keys.add(2); keys.add(3);
Map<Integer, Customer> map = cache.getAll(keys);
```

Bulk Retrieve – getAll()

```
boolean success = cache.remove(1);
```

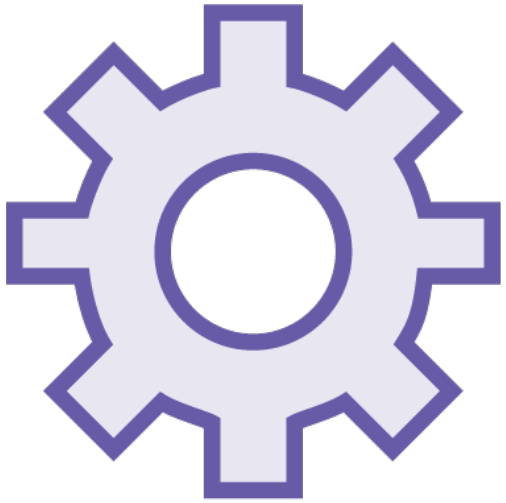
Remove

```
cache.clear();
```

Clear

Pessimistic Locking

Lock



Mutate

Unlock

```
V getAndReplace(K key, V value)

V getAndPut(K key, V value)

V getAndRemove(K key)
```

Lock Free

No guaranteed consistency

```
boolean putIfAbsent(K key, V value)
boolean remove(K key, V oldValue)
boolean replace(K key, V oldValue, V newValue)
boolean replace(K key, V newValue)
V getAndReplace(K key, V newValue)
```

Optimistic Locking

```
cache.close()
//or
cacheManager.close("cacheName")
```

Lifecycle Methods

Freeing Resources

```
// Does not mean data in cache is deleted
cache.close()

//Destroys cache and deletes the data
cacheManager.destroyCache(“cacheName”)
```

Changing Your JCache Implementation

Changing Your JCache Implementation

Users

JCache API

JCache Implementation - Hazelcast

Use JCache API where possible

Avoid implementation specific features

Review

Spring Boot Application

- Dependency Injection

Type Safety

- Runtime
- Compile Time

Cache Interface Operation

Lifecycle Operations

Swap JCache Implementation