Pivotal

Adding Caching

Using @Cachable

Objectives

After completing this lesson, you should be able to

- Add caching to any Spring Bean using the `@Cachable` annotation

Agenda

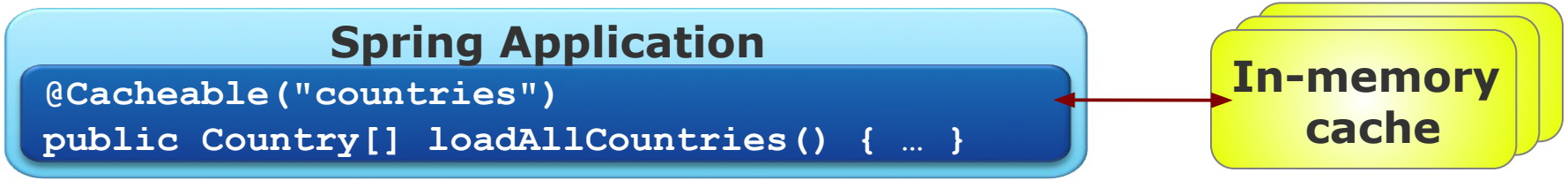
- **Caching with @Caching**

About Caching

- What is a cache?
  - In this context: a key-value store = Map
- Where do we use this caching?
  - Any method that always returns the same result for the same argument(s)
    - This method could do anything
      - Calculate data on the fly
      - Execute a database query
      - Request data via RMI, JMS, a web-service ...
  - A unique key must be generated from the arguments
    - That's the cache key

Caching Support

- Transparently applies caching to Spring beans (AOP)
  - Mark methods cacheable
    - Indicate caching key(s)
    - Name of cache to use (multiple caches supported)
  - Define one or more caches in Spring configuration



See: [Spring Framework Reference – Cache Abstraction](https://docs.spring.io/spring/docs/current/spring-framework-reference/integration.html#cache)
https://docs.spring.io/spring/docs/current/spring-framework-reference/integration.html#cache

Caching with @Cacheable

- `@Cacheable` marks a method for caching
  - its result is stored in a cache
  - subsequent invocations (with the *same arguments)*
    - fetch data from cache using key, method not executed
- `@Cacheable` attributes
  - *value*: name of cache to use
  - *key:* the key for each cached data-item
    - Uses SpEL and argument(s) of method

```
@Cacheable(value="topBooks", key="#refId.toUpperCase()")
public Book findBook(String refId) {...}
```

Caching via Annotations

```
@CacheConfig(cacheNames="topBooks")
public class BookService {

  @Cacheable(key="#title", condition="#title.length < 32")
  public Book findBook(String title, boolean checkWarehouse)  { ... }


  @Cacheable(key="#author.name")
  public Book findBook2(Author author, boolean checkWarehouse)  { ... }

  @Cacheable(key="T(example.KeyGen).hash(#author)")
  public Book findBook3(Author author, boolean checkWarehouse)  { ... }

  @CacheEvict(beforeInvocation=true)
  public void loadBooks() { ... }

  ...
```

All methods use *topBooks* cache

Only cache if condition is true

use object property

custom key generator

clear cache before method invoked

Enabling Caching Proxy

- Caching must be enabled ...

```
@Configuration
@EnableCaching
public class MyConfig {
    @Bean
    public BookService bookService() { … }
}
```

Setup Cache Manager

- Must specify a cache-manager
  - Some provided, or write your own
  - See `org.springframework.cache` package
- **`SimpleCacheManager`**
  - For each cache name, it creates a **`ConcurrentHashMap`**

```
@Bean
public CacheManager cacheManager() {
    SimpleCacheManager cacheManager =
        new  SimpleCacheManager("topAuthors", "topBooks");
    return cacheManager;
}
```

Keep *cacheManager* bean name

Concurrent Map Cache

Third-Party Cache Implementations

- Simple Cache is OK for testing
  - But has no cache control options (overflow, eviction)
- Third-party alternatives
  - Terracotta's EhCache
  - Pivotal's Gemfire
  - Google's Caffeine
  - Infinispan
  - Hazelcast
  - Redis
  - Couchbase

Third-Party Cache Manager – EHCache

EHCACHE

```
@Autowired
ApplicationContext context;

@Value("${ehcache.xml.location}")
String location;

@Bean
public CacheManager cacheManager() {
    Resource cacheConfig =  context.getResource(location);
    net.sf.ehcache.CacheManager cache =
            EhCacheManagerUtils.buildCacheManager(cacheConfig);
    return new EhCacheCacheManager(cache);
}
```

Must specify EhCache's XML configuration file

Location of `ehcache.xml`

Supports `file:` and `classpath:` prefixes

Third-Party Cache Managers – Gemfire

- Gemfire: A distributed, shared nothing data-grid
  - Can be used to setup a distributed cache
  - Caches (regions) replicated across multiple nodes
    - Consistent updates occur on all copies in parallel
    - No loss of data if a storage node fails
    - Automatic recovery and rebalancing

Also sets up `@EnableCaching`

```
@Configuration
@EnableGemfireCaching
@ClientCacheApplication(logLevel = "error", name = "CachingGemFireApplication")
public class GemfireCacheConfig {
}
```

Spring Gemfire Project

- GemFire configuration in Spring config files
  - Also enables configuration injection for environments
- Features
  - Exception translation
  - GemfireTemplate
  - Transaction management (*GemfireTransactionManager*)
  - Injection of transient dependencies during deserialization
  - *Gemfire Cache Manager class*

GEMFIRE®