# Spring Boot ve Cache

# Ön Bellek (Caching) İşlemleri

- Spring Boot **cache starter** ve Configuration sınıfının üzerinde **@EnableCaching** anotasyonu mevcut ise metot düzeyinde cache kabiliyetini devreye alır

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-cache</artifactId>
</dependency>
```

```
@SpringBootApplication
@EnableCaching
public class PetClinicApplication {
    ...
}
```

# Ön Bellek (Caching) İşlemleri

- @**Cacheable** anotasyonuna sahip bean metotlarının return değerleri cache'lenerek müteakip erişimde return değeri cache'den dönülür

```
public class PersonController {
    @Cacheable("personByEmail")
    public Person findByEmail(String email) {
    ...
    }
}
```

# Cache Provider Konfigürasyonu: Simple

- Spring fiziksel olarak herhangi bir cache gerçekleştirimi sunmaz, sadece **Cache** ve **CacheManager** arayüzlerine sahiptir
- Bu arayüzler arkasında **herhangi bir cache provider kütüphanesi** kullanılabilir
- Spring Boot default olarak ApplicationContext'de **Cache** tipinde **en az bir bean mevcut ise** bu bean'ları wrap eden bir CacheManager bean'i oluşturarak onun üzerinden çalışır

# Cache Provider Konfigürasyonu: Simple

- Herhangi bir cache provider mevcut değil ise **ConcurrentHashMap** tipinden cache konfigürasyonu devreye girer
- Cache'ler ilk erişimde otomatik yaratılırlar
- İstenirse application.properties'deki tanım ile **sadece belirtilen** cache'lerin başlangıçta yaratılması da mümkündür

```
spring.cache.cache-names=foo,bar
```

- Bu durumda ismi **belirtilmeyen cache'in kullanılması hata** ile sonuçlanır

# Cache Provider Konfigürasyonu: JCache

- Eğer classpath'de **JSR-107 JCache** kütüphanesi mevcut ise bu devreye alınır
- EhCache3, Hazelcast, Infinispan JCache gerçekleştirimlerinden bazılarıdır

```
<dependency>
    <groupId>javax.cache</groupId>
    <artifactId>cache-api</artifactId>
</dependency>
<dependency>
    <groupId>org.ehcache</groupId>
    <artifactId>ehcache</artifactId>
</dependency>
```

- Spring metotlar üzerinde **JCache anotasyonları**nı da desteklemektedir

# Cache Provider Konfigürasyonu: JCache

- Aynı anda birden fazla JCache kütüphanesi varsa hangisinin kullanılacağı **application.properties**'de belirtilmelidir

```
spring.cache.jcache.provider=org.ehcache.jsr107.EhcacheCachingProvider
spring.cache.jcache.config=classpath:ehcache3.xml
```

- Bootstrap sırasında application.properties'deki tanımla ilave cache'ler de tanımlanabilir

```
spring.cache.cache-names=foo,bar
```

# Cache Provider Konfigürasyonu: EhCache2

- Eğer classpath'de **EhCache 2** kütüphanesi ve **ehcache.xml** dosyası mevcut ise bu kullanılır

```
<dependency>
    <groupId>net.sf.ehcache</groupId>
    <artifactId>ehcache</artifactId>
</dependency>
```

- Farklı bir ehcache konfigürasyonu da yüklenebilir

```
spring.cache.ehcache.config=classpath:config/ehcache2-config.xml
```

# Cache Provider Konfigürasyonu: Hazelcast

- Spring Boot Hazelcast kütüphanesi mevcut ise ve konfigürasyonu da tanımlı ise otomatik olarak **HazelcastInstance** yaratır

```
<dependency>
     <groupId>com.hazelcast</groupId>
     <artifactId>hazelcast</artifactId>
</dependency>
<dependency>
     <groupId>com.hazelcast</groupId>
     <artifactId>hazelcast-spring</artifactId>
</dependency>
```

```
spring.hazelcast.config=classpath:hazelcast.xml
```

- Eğer **HazelcastInstance** otomatik olarak konfigüre edilmiş ise caching için de bu instance kullanılır

# Cache Provider Konfigürasyonu: Infinispan

- Eğer classpath'de **Infinispan** kütüphanesi mevcut ise bu kullanılır

```
<dependency>
     <groupId>org.infinispan</groupId>
     <artifactId>infinispan-spring4-embedded</artifactId>
</dependency>
<dependency>
     <groupId>org.infinispan</groupId>
     <artifactId>infinispan-jcache</artifactId>
</dependency>
```

- Konfigürasyon dosyası **explicit** biçimde belirtilmelidir

```
spring.cache.infinispan.config=infinispan.xml
```

- Infinispan desteği **sadece gömülü sunucu**da geçerlidir

# Cache Provider Konfigürasyonu: Couchbase

- Eğer classpath'de couchbase client ve spring cache kütüphaneleri mevcut ise devreye girer

```
<dependency>
     <groupId>com.couchbase.client</groupId>
     <artifactId>java-client</artifactId>
</dependency>
<dependency>
     <groupId>com.couchbase.client</groupId>
     <artifactId>couchbase-spring-cache</artifactId>
</dependency>
```

- Bootstrap sırasında application.properties'deki tanımla ilave cache'ler de tanımlanabilir

```
spring.cache.cache-names=foo,bar
```

# Cache Provider Konfigürasyonu: Redis

- Eğer data-redis starter'ı mevcut ise devreye girer

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-data-redis</artifactId>
</dependency>
```

- Bootstrap sırasında application.properties'deki tanımla ilave cache'ler de tanımlanabilir

```
spring.cache.cache-names=foo,bar
```

# Cache Provider Konfigürasyonu: Caffeine

- Eğer classpath'de caffeine kütüphanesi mevcut ise devreye girer

```
<dependency>
    <groupId>com.github.ben-manes.caffeine</groupId>
    <artifactId>caffeine</artifactId>
</dependency>
```

- Bootstrap sırasında application.properties'deki tanımla ilave cache'ler de tanımlanabilir

```
spring.cache.cache-names=foo,bar
```

# Cache Provider Konfigürasyon Özelleştirmeleri

- Birden fazla cache provider'ın classpath'de mevcut olması durumunda istenirse Spring Boot'un **sıralamasının dışında** belirtilen cache provider devreye alınabilir

```
spring.cache.type=redis
```

- Bazı ortamlarda ise, örneğin test, **cache** konfigürasyonunun **tamamen devre dışı** bırakılması gerekebilir

```
spring.cache.type=none
```

# İletişim

- Harezmi Bilişim Çözümleri
- Kurumsal Java Eğitimleri
- http://www.java-egitimleri.com
- info@java-egitimleri.com