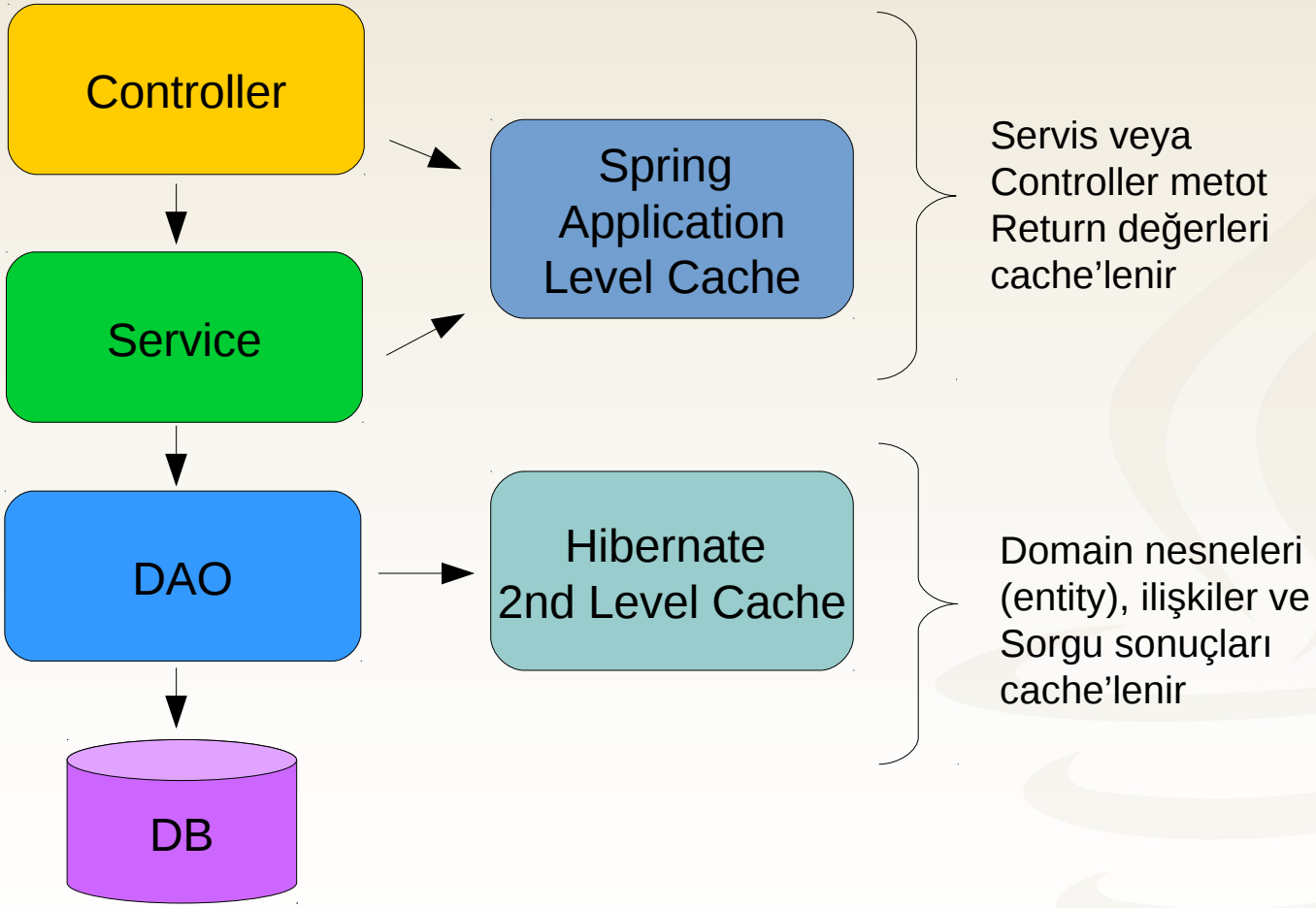
Metot Düzeyinde Cache Kabiliyeti

Spring ve Metot Düzeyinde Caching

- Spring cache **web veya remote metot çağrıları**ndan dönen sonuçların cache'lenmesi için düşünülmüştür
- **Controller** veya **servis** katmanında yer alan ve **cache ihtiyacı** olan bean'lerde kullanılır
- ORM 2nd level cache **persistent domain nesneleri**nin cache'lenmesi içindir
- Spring cache ise **metot return değerleri**ni cache'ler

Cache Katmanları



Spring ve Metot Düzeyinde Caching

- Sınıf veya metot düzeyinde **@Cacheable** anotasyonu ile tanımlanır
- @Cacheable
  - Sınıf düzeyinde bütün metotlarda caching'i devreye alır
  - Metot düzeyinde ise metot spesifik tanımlar yapılabilir
- Ayrıca **@CacheEvict** ve **@CachePut** anotasyonları ile cache içerisindeki veri yönetilebilir

@Cacheable Kullanımı

```
public class FooService {

    @Cacheable("fooWithNameRegion")
    public Foo findFoo(String name) {
        //...
    }

    @Cacheable(
        value="defaultRegion",key="#date.time")
    public Foo findFoo(Date date) {
        //...
    }


    @Cacheable(
        value="defaultRegion",condition="#i>10")
    public Foo findFoo(int i) {
        //...
    }
}
```

Cache bölümünün ismidir, Cache'lenen bilgi fiziksel cache içerisinde bu bölümde tutulacaktır

Cache **key** default durumda **metot parametreleri**nden elde edilir, **SpEL expression** yardımı ile de elde edilebilir

Cache işleminin ne zaman devreye gireceğini belirlemek için SpEL yardımı ile "**condition**" da tanımlanabilir

@CacheEvict ve @CachePut Kullanımı

```
public class FooService {

   @CacheEvict("fooWithNameRegion")
   public void updateFoo(String name) {
      //...
   }

   @CachePut("fooWithNameRegion")
   public Foo insertFoo(String name) {
      //...
   }

}
```

Key değerine karşılık gelen entry cache'den çıkarılır

Metot return değeri cache'e entry olarak eklenir
Default durumda metot input parametreleri ile key değeri belirlenir

Spring ve Metot Düzeyinde Caching Konfigürasyonu

- `<cache:annotation-driven/>` elemanı ile devreye girer
- Container'da tanımlı **CacheManager** arayüzünü implement eden bir bean'a ihtiyaç duyar
- Built-in **EhCache** ve **ConcurrentHashMap** tabanlı implemantasyonları mevcuttur
- Diğer cache provider'lar da **entegre** edilebilir

Spring ve Metot Düzeyinde Caching Konfigürasyonu

```
<cache:annotation-driven/>

<bean id="cacheManager"
class="org.springframework.cache.support.SimpleCacheManager">
    <property name="caches">
        <set>
            <bean
class="org.springframework.cache.concurrent.ConcurrentMapCacheFact
oryBean">
                <property name="name" value="defaultRegion" />
            </bean>
            <bean
class="org.springframework.cache.concurrent.ConcurrentMapCacheFact
oryBean">
                <property name="name" value="fooWithNameRegion" />
            </bean>
        </set>
    </property>
</bean>
```

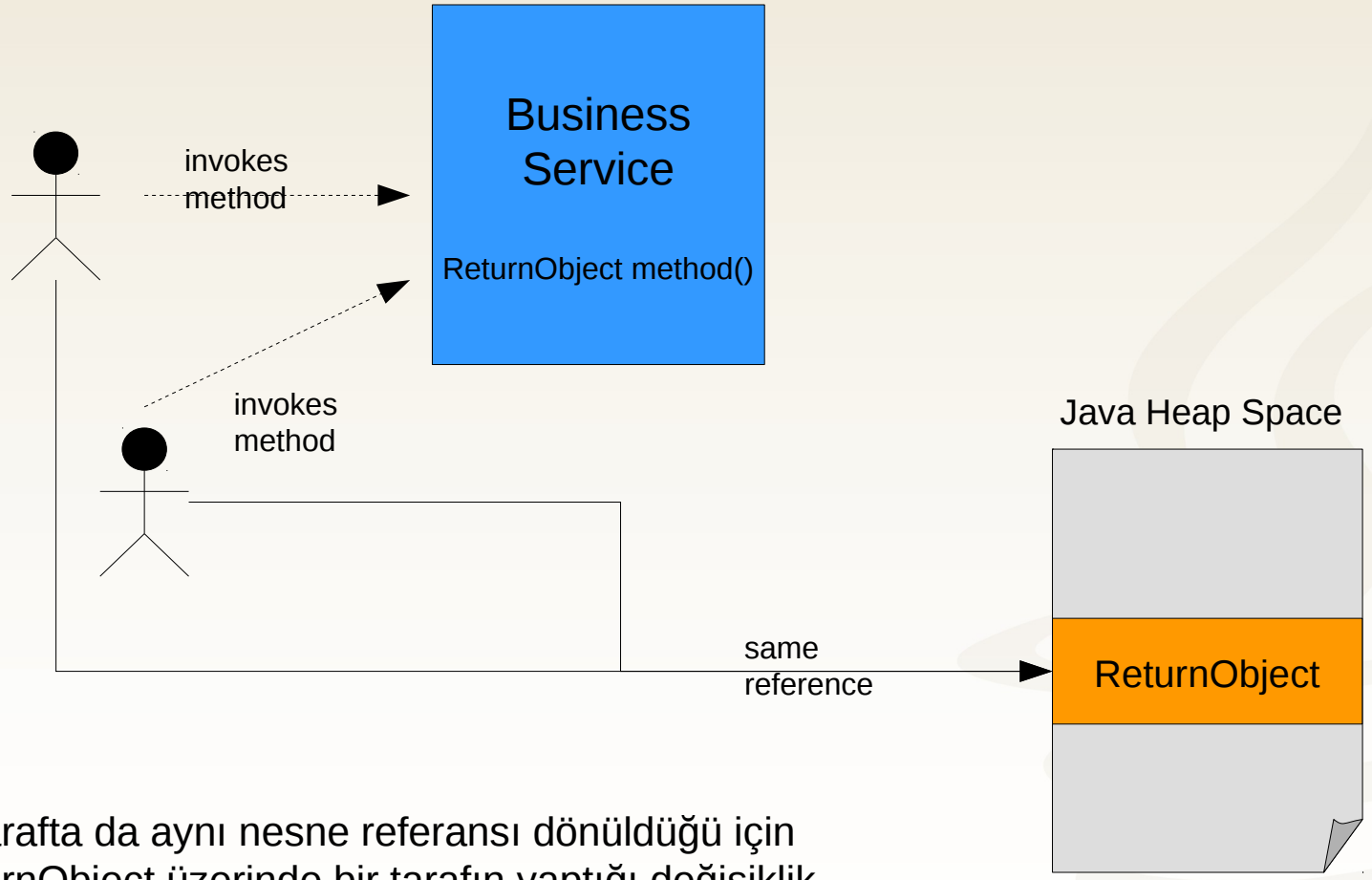
Test ortamları için uygun basit bir CacheManager implemantasyonudur

Her bir **cache region** için bir bean tanımlanır. Nesneler bu cache bölümleri içerisinde **key'e göre gruplanarak** saklanır

Metot Düzeyinde Caching Kullanırken Dikkat Edilecekler

- Metot **return değerleri** cache'lendiği takdirde bu değerlerin **immutable** (read-only) olması önemlidir
- Aksi takdirde cache'den dönen aynı nesne referansı üzerinden hareket edildiği için **istenmeyen yan etkiler** ortaya çıkabilir
- Bu nedenle caching kullanımı **controller katmanı** için daha uygundur

Metot Düzeyinde Caching Kullanırken Dikkat Edilecekler



İki tarafta da aynı nesne referansı dönüldüğü için
ReturnObject üzerinde bir tarafın yaptığı değişiklik
Diğer tarafı da etkileyecektir

İletişim

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