

Spring ve Entegrasyon Testleri



Spring TestContext Framework'ün Özellikleri

- Entegrasyon testlerinin **standalone ortamda yazımı ve çalıştırılmasını** mümkün kılar
- ApplicationContext'i oluşturan **bean tanımlarının doğru yapıp yapılmadığının** kontrolü sağlanır
- JDBC ve ORM araçları ile **veri erişiminin testi** yapılır
 - SQL, HQL sorgularının kontrolü yapılır
 - ORM mapping'lerin düzgün yapıp yapılmadığı test edilmiş olur

Spring TestContext Framework'ün Özellikleri

- Test sınıflarına **dependency injection** yapılabilir
- TestCase içerisinde **ApplicationContext'e erişmek** de mümkündür
- Test metodlarının **transactional context** içerisinde çalıştırılması mümkündür
- Otomatik **ApplicationContext yönetimi** yapar
- Spring container'ı test sınıfları ve test metotları arasında yeniden oluşturmamak için **cache desteği** sağlar

Spring TestContext Framework Konfigürasyonu

ApplicationContext yüklenmesi
Dependency injection
Transactional test desteği aktive olur

```
@RunWith(SpringJUnit4ClassRunner.class)
@ContextConfiguration
public class FooTests {
    // ...
}
```


Default: classpath:/com/example/FooTests-context.xml
locations attribute ile farklı dosyalar belirtilebilir

```
@RunWith(SpringJUnit4ClassRunner.class)
@ContextConfiguration("/appContext.xml")
public class BarTests {
    // ...
}
```

Entegrasyon Testleri ve ApplicationContext Yönetimi

```
@RunWith(SpringJUnit4ClassRunner.class)
@ContextConfiguration("/base-context.xml")
public class BaseTest {
    // ...
}
```

```
@ContextConfiguration("/extended-context.xml")
public class ExtendedTest extends BaseTest {
    // ...
}
```



ExtendedTest sınıfı BaseTest sınıfından türediği için bu sınıftaki test metotları için yaratılacak olan ApplicationContext base-context.xml ve extended-context.xml dosyaları yüklenerek oluşturulacaktır

ApplicationContext Yönetimi ve ÖnBellekleme

- ApplicationContext test sınıfı bazında sadece **bir kereliğine** yüklenir
- ApplicationContextAware arayüzünü implement ederek veya **@Autowired** ile enjekte ederek **test sınıfı içerisinde ApplicationContext'e erişilebilir**
- Herhangi bir test sonrası ApplicationContext'in yeniden yüklenmesi istenirse, **test sınıfı veya test metodu düzeyinde @DirtiesContext** kullanılabilir

ApplicationContext Yönetimi ve ÖnBellekleme

```
@RunWith(SpringJUnit4ClassRunner.class)
@ContextConfiguration("/appcontext/beans-*.xml")
public class SpringTests {
```

```
    @Autowired
    private ApplicationContext applicationContext;
```

```
    @Test
    @DirtiesContext
    public void testMethod1() {
    }
```

Bu test metodu çalıştıktan sonra ApplicationContext bir sonraki test metodu çalıştırılmadan önce yeniden yaratılacaktır

```
    @Test
    public void testMethod2() {
    }
```

```
}
```

Entegrasyon Testleri ve Dependency Injection

- **@Autowired** veya JSR-250 **@Resource** anotasyonları ile test sınıflarına bağımlılıklar enjekte edilebilir
- **Field** veya **setter** metot düzeyinde injection yapılabilir
- **@Autowired** default **byType** ile injection yapar
- **@Resource** ile **byName** injection yapılabilir

Entegrasyon Testleri ve Dependency Injection

```
@RunWith(SpringJUnit4ClassRunner.class)
@ContextConfiguration
public class SpringTests {

    @Autowired
    private FooService fooService;

    ...
}
```

Web Uygulamalarının Test Edilmesi

- Spring standalone testler içerisinde **WebApplicationContext**'in oluşturulmasını sağlar
- Böylece **Controller katmanı** da entegrasyon testlerine dahil edilebilir

```
@RunWith(SpringJUnit4ClassRunner.class)
@WebAppConfiguration("webapp")
@ContextConfiguration("/appcontext/beans-*.xml")
public class SpringWebAppTests {
    ...
}
```

ContextRoot dizinini belirler
Default **src/main/webapp**'dir

Web Uygulamalarının Test Edilmesi

```
@RunWith(SpringJUnit4ClassRunner.class)
@WebAppConfiguration("webapp")
@ContextConfiguration("/appcontext/beans-*.xml")
public class SpringWebAppTests {

    @Autowired
    private WebApplicationContext wac; // cached

    @Autowired
    private MockServletContext servletContext; // cached

    @Autowired
    private MockHttpSession session;

    @Autowired
    private MockHttpServletRequest request;

    @Autowired
    private MockHttpServletResponse response;

}
```

Testler için oluşturulan
WebApplicationContext
nesnesi

Test metotları içerisinde
erişilebilecek built-in
mock nesnelerdir

Spring'in Servlet API
Mock sınıflarıdır

Testleri Ortama Göre Çalıştırmak

- **@IfProfileValue** anotasyonu ile belirli bir ortam veya sistem değişkeninin değerine göre testler enable/disable edilebilir
- Sınıf veya metot düzeyinde kullanılabilir

```
@IfProfileValue(name="targetPlatform", value="test")  
@Test  
public void testMethod() {  
    // ...  
}
```

Test metodu eğer ifade true olarak evaluate ediyor ise çalıştırılır. Sınıf düzeyinde kullanılırsa o sınıftaki hiçbir test metodu çalıştırılmayacaktır

Testler ve Java Tabanlı Container Konfigürasyonu

```
@RunWith(SpringJUnit4ClassRunner.class)
@ContextConfiguration(classes={AConfig.class,BConfig.class})
public class SpringTests {
    @Autowired
    private Foo foo;

    @Test
    public void testFoo() {
        Assert.assertNotNull(foo.getBar());
    }
}
```

XML konfigürasyon dosyaları
yerine konfigürasyon
sınıfları listelenir

```
@RunWith(SpringJUnit4ClassRunner.class)
@ContextConfiguration
public class SpringTests {
    @Configuration
    static class TestContextConfiguration {
        @Bean
        public Foo foo() {
            return new Foo();
        }
    }
}
```

Test'e özel bean konfigürasyon
sınıfı inner class olarak
tanımlanabilir

İletişim

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

