DataSource Konfigürasyonu

DataSource Konfigürasyonu

- Spring Boot veritabanı bağlantılarını yönetmek için **javax.sql.DataSource** tipinde bir bean'ı otomatik olarak tanımlamaktadır
- Veritabanı bağlantı ayarları dataSource bean'ine özel property'ler ile yapılır

```
spring.datasource.url=jdbc:h2:tcp://localhost/~/test
spring.datasource.username=sa
spring.datasource.password=
spring.datasource.driver-class-name=org.h2.Driver
```

JNDI DataSource

- Spring Boot uygulaması bir uygulama sunucusuna deploy edilecek ise **JNDI** üzerinden bu **sunucudaki dataSource**'u da kullanması sağlanabilir

```
spring.datasource.jndi-name=java:comp/env/TestDB
```

Custom DataSource

- İstenirse tamamen **uygulamaya özel** bir **dataSource bean tanımı** da yapılabilir
- Bu durumda Spring Boot dataSource ve pooling konfigürasyonu da tamamen **devre dışı** kalacaktır

```
@Configuration
public class CustomDataSourceConfiguration {
    @ConfigurationProperties(prefix="app.datasource")
    @Bean
    public DataSource dataSource() {
        return new DriverManagerDataSource();
    }
}
```

```
app.datasource.url=jdbc:mysql://localhost/test
app.datasource.username=dbuser
app.datasource.password=dbpass
```

DataSourceBuilder

- classpath'de yer alan standart DataSource sınıflarını kullanarak **DataSource build** etmeyi sağlar

```
@Configuration
public class CustomDataSourceConfig {

    @ConfigurationProperties("app.datasource")
    @Bean
    public DataSource dataSource() {
        return DataSourceBuilder.create().build();
    }
}
```

Hangi pool kütüphanesini kullanacağına classpath'den karar verir, ayrıca Driver bilgisini de url property tanımından tespit edebilir

DataSourceBuilder

- İstenirse **spesifik pool kütüphanesi** de belirtilebilir

```
@Configuration
public class CustomDataSourceConfig {
    @ConfigurationProperties("app.datasource")
    @Bean
    public HikariDataSource dataSource() {
        return DataSourceBuilder.create()
                    .type(HikariDataSource.class).build();
    }
}
```

DataSourceBuilder

- İstenirse DataSourceBuilder'ın **spesifik bir DataSourceProperties ile initialize** edilmesi de mümkündür

```
@Configuration
public class CustomDataSourceConfig {
     @Bean
     @ConfigurationProperties("app.datasource.configuration")
     public HikariDataSource dataSource(DataSourceProperties properties) {
          return properties.initializeDataSourceBuilder()
                                   .type(HikariDataSource.class).build();
     }
     @Bean
     @Primary
     @ConfigurationProperties("app.datasource")
     public DataSourceProperties dataSourceProperties() {
          return new DataSourceProperties();
     }
}
```

```
app.datasource.url=jdbc:mysql://localhost/test
app.datasource.username=dbuser
app.datasource.password=dbpass
app.datasource.configuration.maximum-pool-
size=30
```

İletişim

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