**@Value**

• **org.springframework.beans.factory.annotation.Value** is

an annotation for external properties.

• It is used at field and parameter level.

• It is mostly used for expression-driven or property-driven dependency

injection.

• Expression-driven means using **Spring Expression Language (SpEL),**

• Property-driven means accessing the properties of other beans.

Expression driven olarak ya da property driven olarak da kullanabiliriz.

Diğer beanların propertylerini inject edebiliriz.

Expression-driven ise herhangi bir yerinden propertyi okuyarak inject edilmesini sağlayabiliriz.

• **Spring Expression Language (SpEL)** is an expression language that

provides querying and manipulating an object graph at runtime:

• Literal expressions

• Boolean and relational operators, assignment

• Regular expressions

• Class expressions and method invocation

• Accessing properties, arrays, lists, and maps, etc.

**SpEL** çok hızlı erisime izin veren, ve runtime manipulasyon sağlayan yapılardır.

• Its API is mainly in **org.springframework.expression** and its sub packages.

• It has its own parser:

**org.springframework.expression.ExpressionParser** which

has an implementation **SpelExpressionParser**.

• Every SpEL expression is represented by **Expression** interface.

**@Value Continue**

• @**Value** has a required attribute called **value** which designates the

value.

• Simple string values, properties of other beans and more complex

values using **SpEL** can be injected.

• All type conversions are handled automatically by Spring.

• A SpEL element defines the value of the property of a bean using

**#{expression}.**

• For @**Value** annotations, an expression resolver is preconfigured to

look for bean names when resolving expression text.

• getter methods are called when accessing bean properties.

• A **SpEL** element defines the value of a property specified in a properties

file through **${property-name}.**

propery file üzerinden ulaşacaksak **${property-name}. kullanırız.**

**Properties File**

• **Properties files are specified using <context:propertyplaceholder**

**location=""> in XML file or @PropertySource in source code.**

• If both are specified Spring combines them.

• If a property name collision occurs the last source overrides.

• If both are specified properties file specified in XML file is loaded first and

then properties file specified in annotation is loaded.

• Both use **classpath**: and **file**: for the path of the properties file.

• ${…​} placeholders can also be used in location of properties file when

specified in XML or annotation for the replacement of some values.

• In case of multiple properties file @**PropertiesResources** annotation

can be used.

• In XML multiple properties files can be provided using commaseparated paths.