**Java’s Standard DI Annotations**

• **Spring** supports Java’s standard injection methods:

• As part of JSR-250 **@Resource**

• As part of JSR-330 standard DI annotations such as **@Inject** and **@Named**

**Support for JSR-250**

• Spring 2.5 added support for JSR-250 annotations:

• **@Resource** will be introduced here.

• **@PostConstruct** and **@PreDestroy** will be introduced later in lifecycle management.

• These are **in javax.annotation package** which is part of Java EE and was also part of **java.xml.ws** module of JDK.

• Starting version 11, this module is not part of the JDK anymore so its artifacts should be added to the project separately

**@Resource**

• **javax.annotation.Resource** is an annotation used on fields and property setter methods for injection.

• **@Resource** takes several attributes one of which is name.

• **Spring** takes the value of **name** attribute as the bean name to be injected.

• If no name is specified, the default name is derived from the field name or the property name if a setter method is annotated.

• All of the Spring’s qualification mechanisms work well with @Resource.

• The main use case to use **@Resource** with Spring might be having a

piece of Java code that had already used **@Resource** and reusing it in a new project where Spring is utilized.

• Other than that use case there is no need to use **@Resource** in a project that uses Spring.

**JSR-330**

• JSR-330 is a specification for Dependency Injection in Java EE.

**• More generally it is called Context and Dependency Injection (CDI).**

• It has been led by Rod Johnson of SpringSource which was the name

of the company for Spring framework at 2009) and Bob Lee of Google.

• Its main annotations **@Inject and @Named** **in javax.inject**

**package.**

• CDI 2.0 is part of Java EE 8 and 3.0 will be part of Jakarta EE 9.0.

• Weld (**https://weld.cdi-spec.org/)** is the reference implementation of DI for Java EE platform.

• There are some other implementations such as Apache Commons Inject

**(https://commons.apache.org/sandbox/commons-inject/index.html).**

• As of now the DI spec is implemented as Weld 3.1.5.

• 3.0 is being implemented as Weld 4.

• Spring 3.0 added support for JSR-330 annotations.

**JSR-299**

• **JSR-299 is another specification for Context and Dependency**

**Injection (CDI) for Java**.

• It has been led by Gavin King of RedHat.

• JSR-299 is built on the top of JSR-330 and adds some advancedfeatures.

**Support For JSR-330**

**@Inject**

**• @Inject can be used instead of @Autowired.**

**• It has no attribute and it can be used at field, constructor and method**

**level.**

**• @Inject injects any Java object which is a POJOs.**

• There is no need to mark POJOs to be injected **by @Inject**.

• Only configuration needed for DI to work is a beans.xml file in METAINF

folder in the root of the packages.

**@Named**

• **@Named** does the same functionality **of @Component**.

• It has an attribute called value of type String which designates the string-**based qualifier.**

• It is also used to qualify beans for injection.

• In this usage it has the same functionality of **Qualifier of Spring**.

• In **fact javax.inject.Qualifier** can be used to build custom qualifiers.

@Named @Component’in karsılığı

@Inject ise @Autowired’in karsılığı.

• The main use case to use JSR-330’s injection mechanism with Spring

might be as with @**Resource**, having a piece of Java code that had

already used JSR-330’s injection annotations and reusing it in a new

project where Spring is utilized.

• Other than that use case there is no need to use JSR-330’s injection

mechanisms in a project that uses Spring.

• That’s because Spring provides all kinds of DI structures.