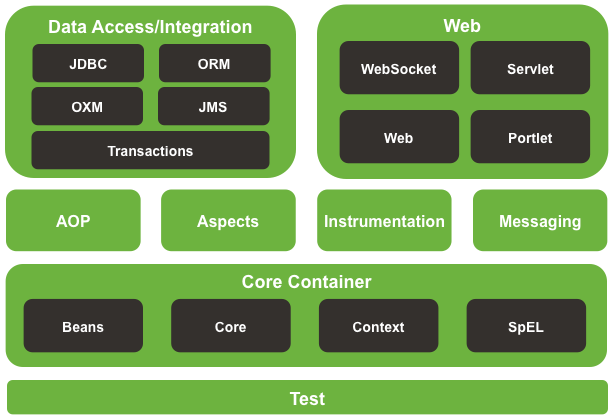
# Overview

What is Spring Framework? From Dependency Injection to Web MVC: <https://www.marcobehler.com/guides/spring-framework>



# Framework Artifacts

GroupId: org.springframework

| **ArtifactId** | **Description** |
| --- | --- |
| spring-aop | Proxy-based AOP support |
| spring-aspects | AspectJ based aspects |
| spring-beans | Beans support, including Groovy |
| spring-context | Application context runtime, including scheduling and remoting abstractions |
| spring-context-support | Support classes for integrating common third-party libraries into a Spring application context |
| spring-core | Core utilities, used by many other Spring modules |
| spring-expression | Spring Expression Language (SpEL) |
| spring-instrument | Instrumentation agent for JVM bootstrapping |
| spring-instrument-tomcat | Instrumentation agent for Tomcat |
| spring-jdbc | JDBC support package, including DataSource setup and JDBC access support |
| spring-jms | JMS support package, including helper classes to send and receive JMS messages |
| spring-messaging | Support for messaging architectures and protocols |
| spring-orm | Object/Relational Mapping, including JPA and Hibernate support |
| spring-oxm | Object/XML Mapping |
| spring-test | Support for unit testing and integration testing Spring components |
| spring-tx | Transaction infrastructure, including DAO support and JCA integration |
| spring-web | Web support packages, including client and web remoting |
| spring-webmvc | REST Web Services and model-view-controller implementation for web applications |
| spring-webmvc-portlet | MVC implementation to be used in a Portlet environment |
| spring-websocket | WebSocket and SockJS implementations, including STOMP support |

# Modules

## Core

IOP (DI)

## AOP

Function (business logic) und non functional (i.e. logging) trennen.

## WEB

MVC

## JDBC / Data Access

ORM: Hibernate, JPA, IBATIS, …

OXM: JAXB, Castor, JiBX, …

## SpEL

Spring Expression Language

# Module Core

## Dependency Injection

<https://www.vojtechruzicka.com/field-dependency-injection-considered-harmful/>

* Field Injection: use in tests if Mocking is not an option
* Setter Injection: use for optional dependencies
* Constructor Inject: use for mandatory dependencies
* Method Injection: rarely used

### Annotations for DI

* @Autowired,
* @Configurable
* @Qualifier
* @Resource

@Autowired on Setters

@Autowired on Properties

@Autowired on Constructors

### Application Context

constructor-arg: Wenn Zielklasse nicht autowirable ist

1. constructor-arg: Schachtelung

       <bean id=*"beanId1"* class=*"de.psicho.package1.className1"*>

             <constructor-arg>

                    <bean class=*"de.psicho.package2.className2"*>

                           <constructor-arg value=*"2"* />

                    </bean>

             </constructor-arg>

       </bean>

1. constructor-arg: Value

       <bean id=*"beanId2"* class=*" de.psicho.package3.className3"*>

             <constructor-arg value=*"${variable}"* />

       </bean>

1. constructor-arg: Reference

       <bean id=*"beanId3"* class=*" de.psicho.package4.className4"*>

             <constructor-arg ref=*"beanIdX"* />

       </bean>

1. annotation-config

Vorteil: Es können externe Beans konfiguriert werden, auf die kein component-scan möglich ist (z.B. java.lang.String)

  <context:annotation-config />

1. component-scan

Vorteil: Konfiguration komplett im Bean („@Component“)

  <context:component-scan

    base-package=*"de.psicho.\*,de.psicho.package.class"* />

### Annotations for component-scan vs. annotation-config

* <context:annotation-config> activates many different annotations in beans, whether they are defined in XML or through component scanning.
* <context:component-scan> is for defining beans without using XML.
* <context:component-scan> recognizes a superset of annotations recognized by <context:annotation-config>, namely:

*CLASSPATH component scanning:* Annotations discovered by <context:component-scan>:

* + @Component, @Service, @Repository, @Controller, @Endpoint

*Java @Configuration:* Annotation discovered by <context:component-scan> (**bold**: also discovered by <context:annotation-config>):

* + **Autowired**, Bean, **Classifier**, **Configurable**, Configuration, DependsOn, Import, ImportResource, Lazy, Order, Primary, Profile, **Qualifier**, **Required**, Scope, **Value**, **MessageEndpoint**

### Scope

Standardmäßig verwendet Spring den Scope “singleton”. Scopes sind:

|  |  |
| --- | --- |
| **Scope** | **Erklärung** |
| singleton | Singleton |
| prototype | Beliebig viele Instanzen |
| request | This scopes a bean definition to an HTTP request. Only valid in the context of a web-aware Spring ApplicationContext. |
| session | This scopes a bean definition to an HTTP session. Only valid in the context of a web-aware Spring ApplicationContext. |
| global\_session | This scopes a bean definition to a global HTTP session. Only valid in the context of a web-aware Spring ApplicationContext. |

# JSR-250 (annotations)

|  |  |  |
| --- | --- | --- |
| **Annotation name** | **description** | **Java EE** |
| Generated | Marks sources that have been generated | nein |
| Resource | Declares a reference to a resource, e.g. a database | nein |
| Resources | Container for multiple Resource annotations | nein |
| PostConstruct | Is used on methods that need to get executed after dependency injection is done to perform any initialization. | nein |
| PreDestroy | Is used on methods that are called before the instance is removed from the container | nein |
| Priority | Is used to indicate in what order the classes should be used. For, e.g., the Interceptors specification defines the use of priorities on interceptors to control the order in which interceptors are called. | ja |
| RunAs | Defines the role of the application during execution in a Java EE container | ja |
| RolesAllowed | Specifies the security roles permitted to access method(s) in an application. | ja |
| PermitAll | Specifies that all security roles are permitted to access the annotated method, or all methods in the annotated class. | ja |
| DenyAll | Specifies that no security roles are allowed to invoke the specified method(s). | ja |
| DeclareRoles | Used to specify the security roles by the application. | ja |
| DataSourceDefinition | Is used to define a container DataSource and be registered with JNDI. The DataSource may be configured by setting the annotation elements for commonly used DataSource properties. | ja |
| ManagedBean | Is used to declare a Managed Bean which are container managed objects that support a small set of basic services such as resource injection, lifecycle callbacks and interceptors. | ja |

# Frontend-lib

Autowiring ermöglicht es, dass der Controller zwar die GUI kennt, die GUI aber nicht den Controller.