SpringFactoriesLoader:

SpringFactoriesLoader loads and instantiates factories of a given type from "META-INF/spring.factories" files which may be present in multiple JAR files in the classpath. The spring.factories file must be in Properties format, where the key is the fully qualified name of the interface or abstract class, and the value is a comma-separated list of implementation class names.

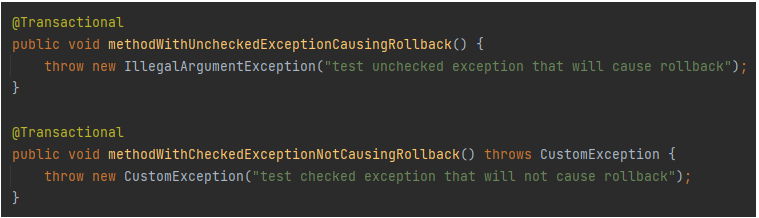
1) SpringFactoriesLoader will scan the META-INF/spring.factories file in the classpath.

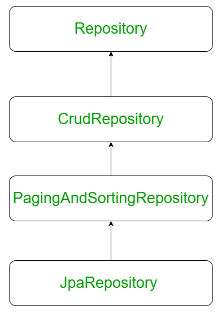
  2) SpringFactoriesLoader will load and instantiate the specified types in META-INF/spring.factories

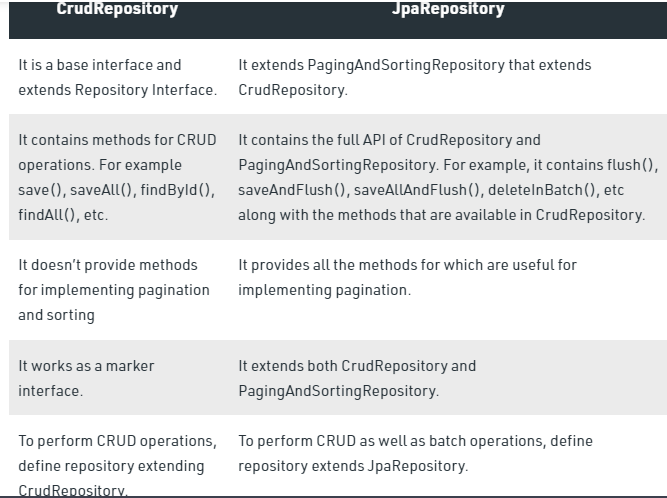
   3) The content of META-INF/spring.factories must be in the form of Key-Value of properties, with multiple values ​​separated by commas.

The **SecurityContext**and **SecurityContextHolder**are two fundamental classes of Spring Security. The SecurityContext is used to store the details of the currently authenticated user, also known as a principle. So, if you have to get the username or any other user details, you need to get this SecurityContext first. The SecurityContextHolder is a helper class, which provides access to the security context. By default, it uses a [ThreadLocal](http://javarevisited.blogspot.com/2012/05/how-to-use-threadlocal-in-java-benefits.html#axzz54hCQprEv) object to store security context, which means that the security context is always available to methods in the same thread of execution, even if you don't pass the SecurityContext object around  
  
Read more: <https://javarevisited.blogspot.com/2018/02/what-is-securitycontext-and-SecurityContextHolder-Spring-security.html#ixzz7Xl6e9BNo>

Default rollback policy in Spring Framework is set to automatic rollback, but only when unchecked(run time) exception is being thrown from the method annotated with @Transactional annotation. When checked exception is being thrown from the method, transaction is not being rolled back.







The Spring Expression Language (SpEL for short) is **a powerful expression language that supports querying and manipulating an object graph at runtime**

A basic benefit of dependency injection is **decreased coupling between classes and their dependencies**. By removing a client's knowledge of how its dependencies are implemented, programs become more reusable, testable and maintainable.

Spring offers fully typed advice - meaning that **you declare the parameters you need in the advice signature** (as we saw for the returning and throwing examples above) rather than work with Object[] arrays all the time.

**What is ContextStartedEvent event?**

**This event is published when the ApplicationContext is started using the start() method on the ConfigurableApplicationContext interface**.

**What is ContextClosedEvent event?**

This event is published when the ApplicationContext is closed using the close() method on the ConfigurableApplicationContext interface.

**Advice is the actual action to be taken either before or after the method execution.**

Before advice: **Advice that executes before a join point, but which does not have the ability to prevent execution flow proceeding to the join point** (unless it throws an exception).Run Advice before method execution

After returning advice: **Advice to be executed after a join point completes normally**: for example, if a method returns without throwing an exception. After throwing advice: Advice to be executed if a method exits by throwing an exception.

After-throwing advice runs after http request is processed and an exception occurred.

**Can we inject value and ref both together in a bean? True**

**How to use ref keyword in beans?**

**Using Setter methods ans constructor arguments**

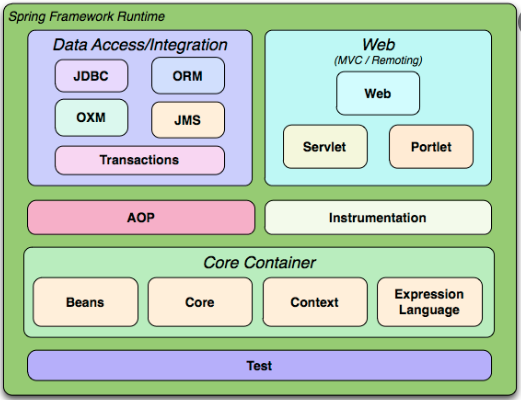
**How do you turn on annotation wiring?**

**To enable @Autowired, you have to register 'AutowiredAnnotationBeanPostProcessor', and you can do it in two ways :**

1. Include <context:annotation-config /> Add Spring context and <context:annotation-config /> in bean configuration file. ...
2. Include AutowiredAnnotationBeanPostProcessor.

**Which are the modules of core container?**

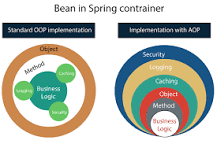
Beans, Core, Context, SpEL are the modules in core container.



ApplicationContext class acts as IoC Container.

AOP (aspect-oriented programming) is a programming style that can be adopted **to define certain policies that in turn are used to define and manage the cross-cutting concerns in an application.**

What are aspects in Spring boot?

[[](https://www.google.com/search?sxsrf=ALiCzsZ0lAB5XH43-lhSAlbd14B5beQu4w:1657080262523&q=What+are+aspects+in+Spring+boot?&tbm=isch&source=iu&ictx=1&vet=1&fir=iDCUI4HIoSU9_M%252CFhY6PI-wxn9k5M%252C_&usg=AI4_-kR_DTvj-CU-tR51umWhm7Fy008g1A&sa=X&ved=2ahUKEwjWyKWDseP4AhXom9gFHQHRAuIQ9QF6BAgUEAE#imgrc=iDCUI4HIoSU9_M)](https://www.google.com/search?sxsrf=ALiCzsZ0lAB5XH43-lhSAlbd14B5beQu4w:1657080262523&q=What+are+aspects+in+Spring+boot?&tbm=isch&source=iu&ictx=1&vet=1&fir=iDCUI4HIoSU9_M%252CFhY6PI-wxn9k5M%252C_&usg=AI4_-kR_DTvj-CU-tR51umWhm7Fy008g1A&sa=X&ved=2ahUKEwjWyKWDseP4AhXom9gFHQHRAuIQ9QF6BAgUEAE" \l "imgrc=iDCUI4HIoSU9_M)

Aspect: **A code unit that encapsulates pointcuts, advices, and attributes**. Class: A code unit that encapsulates methods and attributes. Pointcut: It defines the set of entry points in which advice is executed. Method signature: It defines the entry points for the execution of method bodies.

The @Autowired annotation **provides more fine-grained control over where and how autowiring should be accomplished**. The @Autowired annotation can be used to autowire bean on the setter method just like @Required annotation, constructor, a property or methods with arbitrary names and/or multiple arguments.

JAVA:

It is sometimes good practice to throw an AssertionError explicitly.

1. AssertionErrors should never be handled.( It is never appropriate to write code to handle failure of an assert statement.)
2. Assertion checking is typically enabled during program development and testing
3. Assertion checking can be selectively enabled or disabled on a per-package basis. Note that the package default assertion status determines the assertion status for classes initialized in the future that belong to the named package or any of its "subpackages".
4. The assertion status can be set for a named top-level class and any nested classes contained therein. This setting takes precedence over the class loader's default assertion status, and over any applicable per-package default. If the named class is not a top-level class, the change of status will have no effect on the actual assertion status of any class.

Online tests

https://www.indiabix.com/online-test/java-programming-test/