Bean scope:

In Spring, scope can be defined using spring bean **@Scope** annotation.

There are type of bean scopes.

1. Singleton
2. Prototype
3. Request
4. Session

**Singleton:**

Single bean object instance is created for one spring IoC container.

singleton is default bean scope in spring container. It tells the container to create and manage only one instance of bean class, per container. This single instance is stored in a cache of such [singleton](https://howtodoinjava.com/design-patterns/creational/singleton-design-pattern-in-java/) beans, and all subsequent requests and references for that named bean return the cached instance.

**Prototype:**

Prototype is like it produces a new instance each, and every time a bean is requested.

prototype scope results in the creation of a new bean instance every time a request for the bean is made by application code.

**Request**:

A single instance will be created and available during complete lifecycle of an HTTP request.

In request scope, container creates a new instance for each and every HTTP request. So, if server is currently handling 50 requests, then container can have at most 50 individual instances of bean class. Any state change to one instance, will not be visible to other instances. These instances are destructed as soon as the request is completed.

Session:

A single instance will be created and available during complete lifecycle of an HTTP Session.

In session scope, container creates a new instance for each and every HTTP session. So, if server has 20 active sessions, then container can have at most 20 individual instances of bean class. All HTTP requests within single session lifetime will have access to same single bean instance in that session scope.

Any state change to one instance, will not be visible to other instances. These instances are destructed as soon as the session is destroyed/end on server.

* As a rule, you should prefer to use the prototype scope for all stateful beans and the singleton scope for stateless beans.
* Stateful: A stateful session bean maintains client-specific session information, or conversational state, across multiple method calls and transactions. ... Stateless: A stateless session bean does not maintain conversational state. Instances of a stateless session bean have no conversational state.

Example:

@Component

@Scope("prototype")

public class BeanClass {

}