Bean Visibility

Spring Bean Scopes Type

SCOPE	DESCRIPTION
singleton (default)	Single bean object instance per spring IoC container
prototype	Opposite to singleton, it produces a new instance each and every time a bean is requested.
request	A single instance will be created and available during complete lifecycle of an HTTP request. Only valid in web-aware Spring *ApplicationContext*.
session	A single instance will be created and available during complete lifecycle of an HTTP Session. Only valid in web-aware Spring *ApplicationContext*.
application	A single instance will be created and available during complete lifecycle of *ServletContext*. Only valid in web-aware Spring *ApplicationContext*.
websocket	A single instance will be created and available during complete lifecycle of *WebSocket*. Only valid in web-aware Spring *ApplicationContext*.

singleton and prototype scopes

```java
@Component
//This statement is redundant
- singleton is default scope
@Scope("singleton")
public class BeanClass {
}
```

```java
@Component
@Scope("prototype")
public class BeanClass {
}
```

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 beans, and all subsequent requests and references for that named bean return the cached instance.

prototype scope results in the creation of a new bean instance every time a request for the bean is made by application code.
You should know that destruction bean lifecycle methods are not called **prototype** scoped beans, only initialization callback methods are called.

request and session scopes

```
@Component
@Scope("request")
public class BeanClass {
}
//or
@Component
@RequestScope
public class BeanClass {
}
```

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.

```
@Component
@Scope("session")
public class BeanClass {
}
//or
@Component
@SessionScope
public class BeanClass {
}
```

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.

application and websocket scopes

```java
@Component
@Scope("application")
public class BeanClass {
}
//or
@Component
@ApplicationScope
public class BeanClass {
}
```

In **application** scope, container creates one instance per web application runtime. It is almost similar to singleton scope, with only two differences:

- **application** scoped bean is **singleton** per *ServletContext*, whereas singleton scoped bean is singleton per *ApplicationContext*.
- **application** scoped bean is visible as a *ServletContext* attribute.

```java
@Component
@Scope("websocket")
public class BeanClass {
}
```

The WebSocket Protocol enables two-way communication between a client and a remote host that has opted-in to communication with client. In this type of web applications, HTTP is used only for the initial handshake. If the handshake succeeds, the TCP socket remains open and both client and server can use it to send messages to each other.

Note that **websocket** scoped beans are typically singletons and live longer than any individual WebSocket session.