

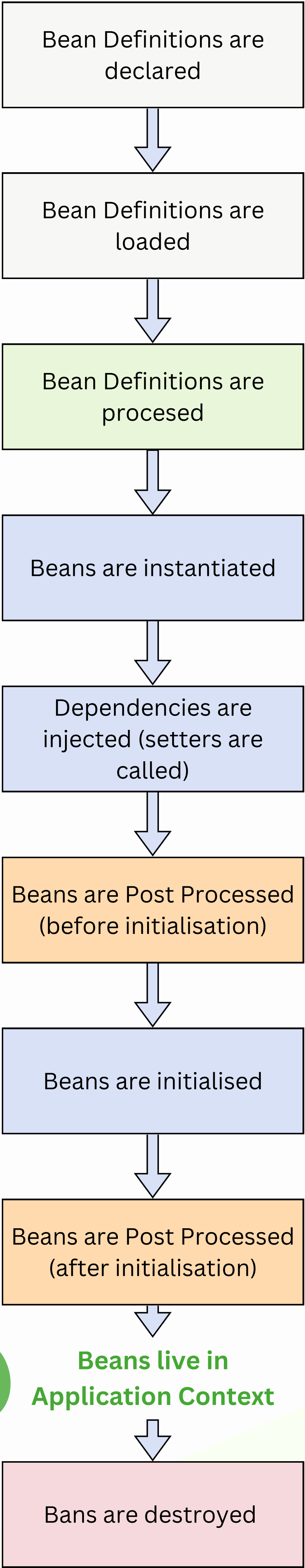
Spring Bean Lifecycle CheatSheet



Spring Bean - POJO (Plain Old Java Object)
managed by the IoC container (Spring)

Bean annotations:

@Bean **@Component** **@Service**
@Controller **@RestController** **@Repository**



Bean Definitions are declared in XML | annotations with package scanning | annotations with a @Configuration-annotated class | Groovy configuration

BeanDefinitionReader

parses the configuration and **creates** BeanDefinition objects.

BeanFactoryPostProcessor

tweaks BeanDefinitions before actual bean creation. Implementing *BeanFactoryPostProcessor* interface grants access to the created BeanDefinitions and allows modifications.

BeanFactory

invokes the constructor of each bean. If needed, it delegates this to custom FactoryBean instances.

If dependencies are injected in the constructor, dependent beans are created first, followed by those that depend on them.

Avoid cyclic dependencies: use constructor Injection | @Lazy | Refactor code

BeanPostProcessor

adjusts beans in the first round (after the object is fully created but before initialisation). Called **before the init-method** (if defined) and returns the bean.

The **@PostConstruct** init method of the bean is triggered, and the bean gets initialised

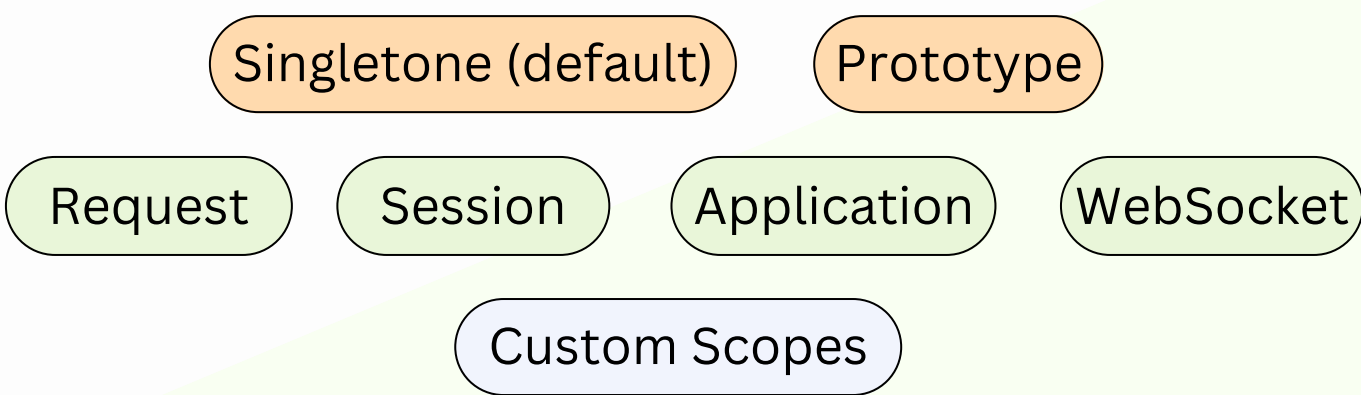
BeanPostProcessor

makes their **second pass** on the bean, usually needed if you need to wrap a proxy around the bean, or in case of circular dependencies

Fully created bean is stored in the context and accessible

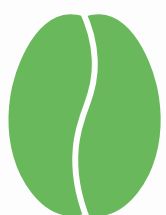
When the context is being destroyed or bean is disposed, the **@PreDestroy** destroy method of the bean is called.

Bean scopes:



Injection annotations

@Autowired
@Setter
@Qualifier (to specify name when autowiring)
@Primary (primary bean of the same type to inject)



Beans live in Application Context