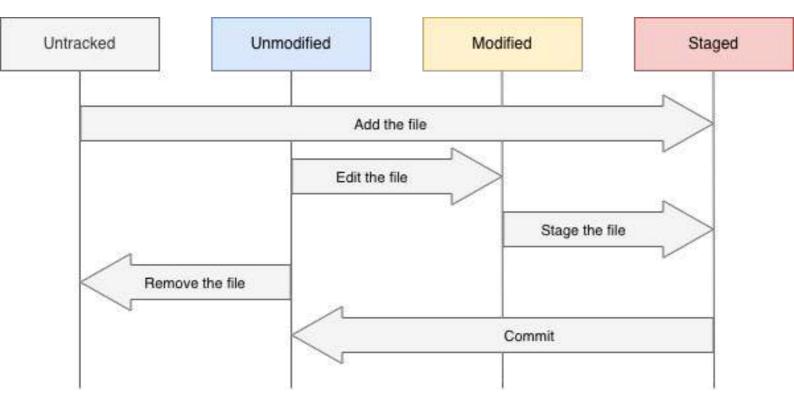
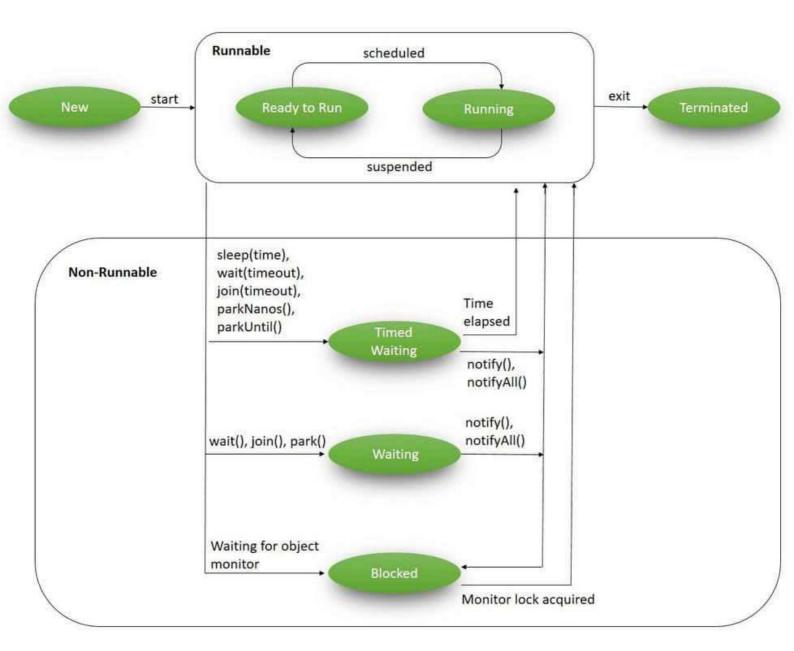
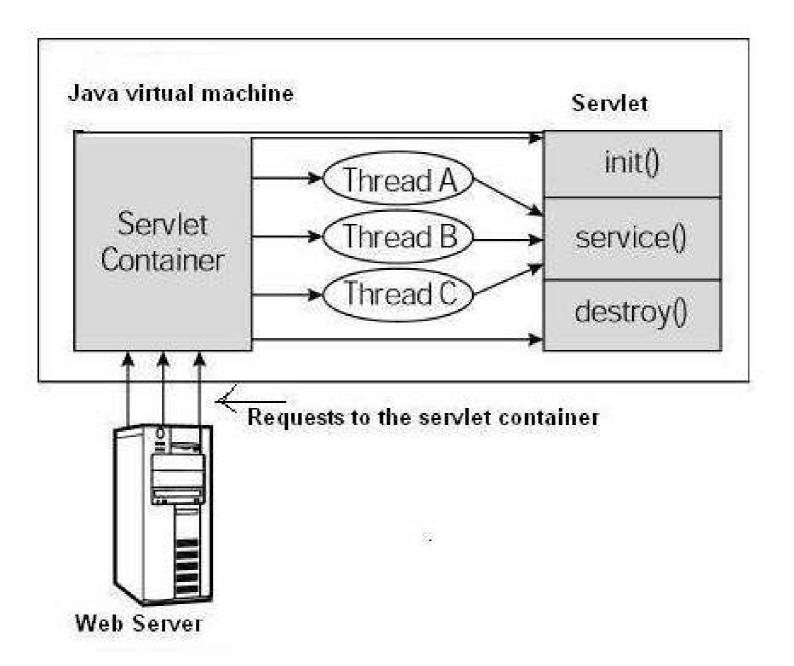


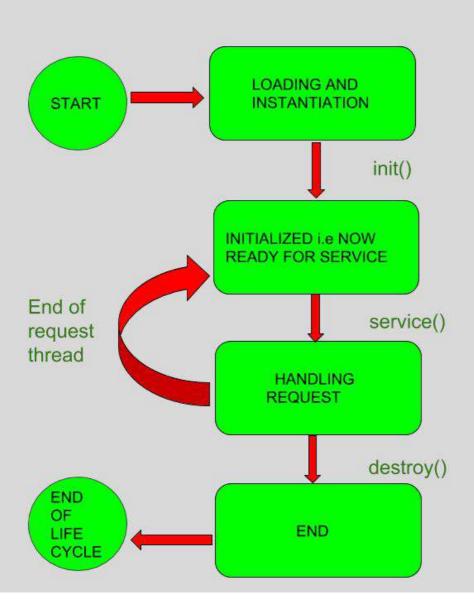
https://toolsqa.com/git/git-life-cycle/

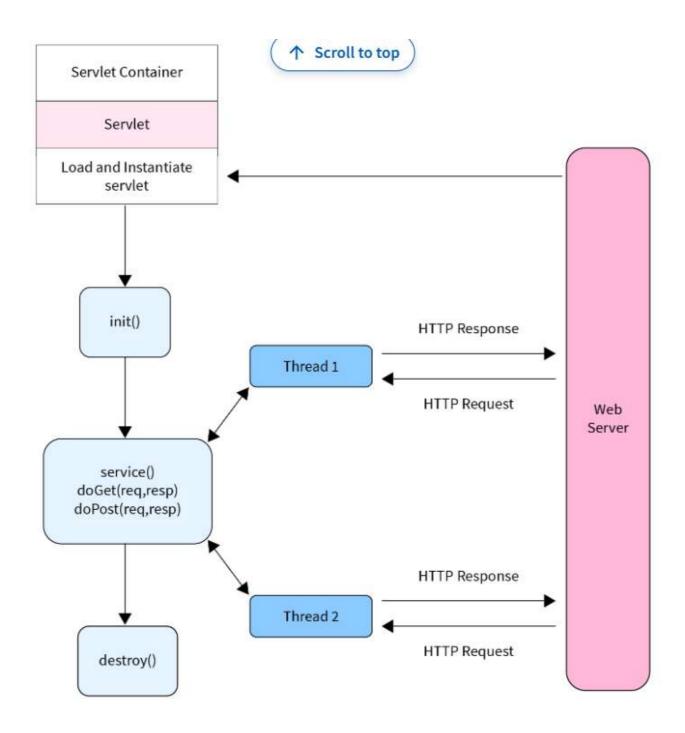


https://www.baeldung.com/java-thread-lifecycle

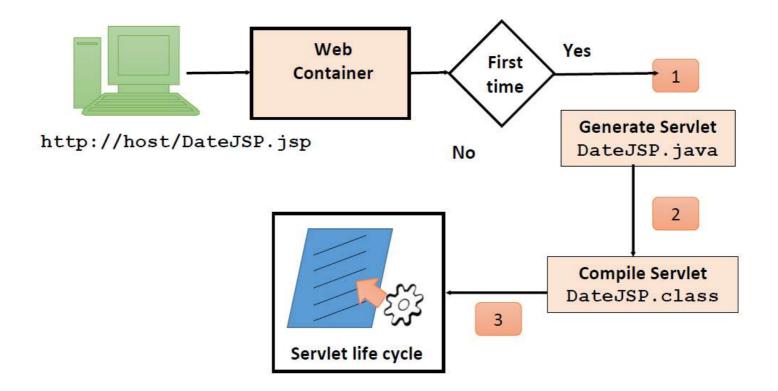








Ch8: JSP Life Cycle



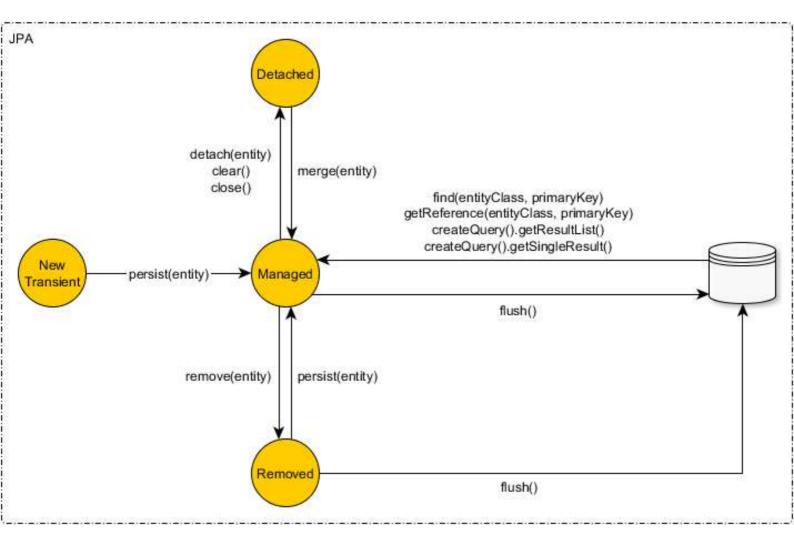
Ch8: JSP Life Cycle

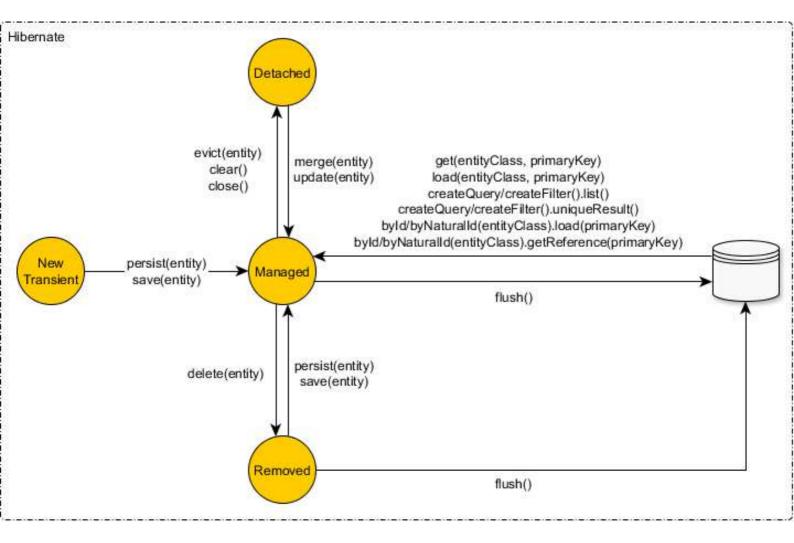
	JSP page translated into servlet	Servlet compiled	Servlet loaded into server's memory	jsplnit called	_jspService called	
Page first written						
Request1	Yes	Yes	Yes	Yes	Yes	
Request2	No	No	No	No	Yes	
Server restarted						
Request3	No	No	Yes	Yes	Yes	
Request4	No	No	No	No	Yes	
Page modified						
Request5	Yes	Yes	Yes	Yes	Yes	
Request6	No	No	No	No	Yes	

Clean Lifecyle
pre-clean
clean
post-clean

Default Lifecyle			
validate	test-compile		
initialize	process-test-classes		
generate-sources	test		
process-sources	prepare-package		
generate-resources	package		
process-resources	pre-integration-test		
compile	integration-test		
process-classes	post-integration-test		
generate-test-sources	verify		
process-test-sources	install		
generate-test-resources	deploy		
processs-test-resources			

Site Lifecyle
pre-site
site
post-site
site-deploy

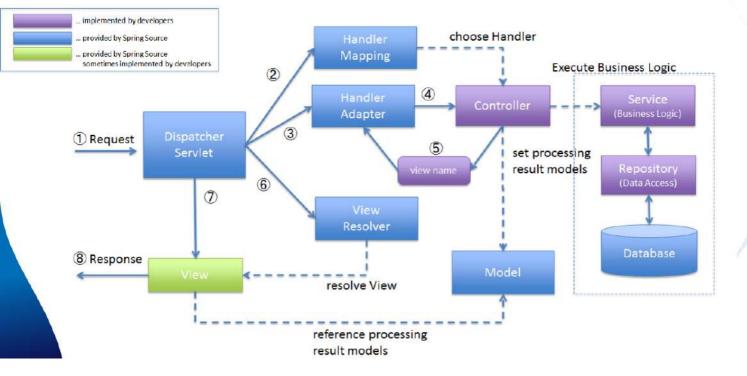






Request Life Cycle





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

Bean Definitions are loaded

Bean Definitions are procesed

Beans are instantiated

Dependencies are injected (setters are called)

Beans are Post Processed (before initialisation)

Beans are initialised

Beans are Post Processed (after initialisation)

Beans live in Application Context

Bans are destroyed

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

Bean scopes: Singletone (default) Prototype Request Application WebSocket Session Custom Scopes

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.

Injection annotations @Autowired

@Qualifier (to specify name when autowiring)

@Primary (primary bean of the same type to inject)