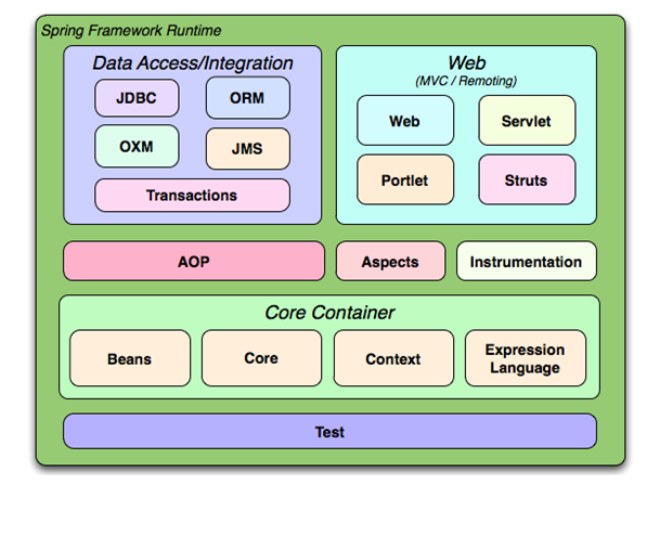
**Spring Framework**

**Entendendo o Core do Spring**

Como funciona

**Core e Beans**

* Estrutura
* IoC
* Injeção de Dependências
* BeanFactory

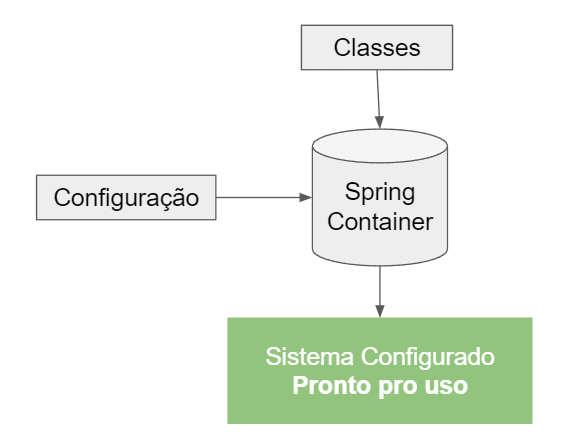
**Context**

* Core e Beans
* Internacionalização
* ApplicationContext

**Expression Language**

Define através de xml ou anotações em tempo de execução, valores e comportamentos dos beans

**Core Container**

****

**Spring IoC Container**

**O que é Inversão de Controle?**

É um princípio de design de programas de computadores onde a sequência (controle) de chamadas dos métodos é invertida em relação à programação tradicional, ou seja, ela não é determinada diretamente pelo programador.

**Application Context**

**Configuração**

* XML
* Codigo Java
* Anotações Java

**XML**

****

**Codigo Java**

****

**Anotações Java**

* @Component
* @Controller
* @Service
* @Repository
* @Autowired

**Entendendo o Spring Beans**

É qualquer objeto gerenciado pelo Spring Ioc.

Então lembra que eu falei que os modulos Bean e Core são as partes essenciais pro Spring funcionar? isso é pq eles são muito ligados, a definição de um depende do outro pra vc ter um bean vc precisa q ele seja monitorado pelo Spring IoC.

Então resumindo, Bean é simplesmente um dos varios objetos na nossa aplicação

**Stereotypes**

**Anotações Spring Bean**

Sim, esses estereótipos são as anotações que a gente ja ta usando no projeto.

E aqui a gente vai ver quais são essas anotações mais usadas e explicar pra q serve cada um deles. Começando então com a @Repository,é uma anotação usada em nivel de classes q representam a camada DAO, ou seja, qualquer classes que tenha alguma interação com banco de dados, e o uso dessa anotação ja faz a tradução de exceções de persistência, então qualquer erro de banco de dados ja vai ser traduzido pra um erro java.

A próxima agora é a @Service, onde geralmente vai ficar a logica da aplicação, as nossas regras de negocio . Essa agora é aquela mesma q a gnt viu no nosso Hello Word, a @Controller fica nas classes que representam os controladores no Spring MVC, ou seja, ela que vai controlar as chamadas das telas do nosso sistema. E a evolução da @Controller é a @RestController, que também é usada em classes controladoras mas não são usadas pra retornar nossas paginas e sim pra retornar dados em JSON.

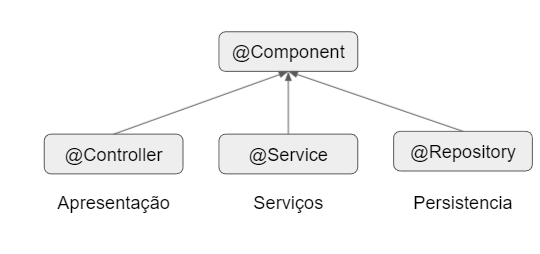
Essa Anotação agora indica uma classe que declara 1 ou mais metodos com @Bean como a gente ja viu na criação do nosso Bean.

Finalizando então pelo @Component, é uma anotação usada em classes pra gerar um bean generico, a gente usa ela quando a classe não se encaixa nas anteriores q a gente viu.

* @Repository
* @Service
* @Controller/@RestController
* @Configuration
* @Component

**@Component**

A @Component foi a ultima que a gente viu, pq ela é uma anotação de estereótipo mais genérica, e as @Controller, @Service e @Repository são especificações da @Component pra usos mais especificos, como apresentação de telas, regras de negocio e persistencia de dados.



**Escopos e ciclo de vida**

**Bean Scopes**

Nós temos no total 6 escopos de beans. O Escopo de um bean vai definir a visibilidade e seu ciclo de vida onde ele for usado.

começando com o escopo Singleton, significa que só vai existir uma única instância desse bean, por padrão quando a gente não fala explicitamente qual o escopo, o Spring ja utiliza o Singleton.

Agora com o escopo de prototipo, bem diferente do singleton que cria só uma instância, aqui uma instância diferente é criada cada vez que o bean é solicitado.

A proxima ja não usa a mesma anotação das 2 anteriores, a @RequestScope é uma anotação diferente, mas é uma especificação da @Scope e cria uma instancia de bean pra uma unida solicitação HTTP.

Do mesmo jeito da anterior, a @SessionScope é uma especificação de @Scope mas essa ja cria uma instancia de bean pra cada sessão HTTP.

A @ApplicationScope é tambem uma especialização da @Scope pra indicar um bean com ciclo de vida vinculado com um aplicativo web.

E pra finalizar o escopo de webSocket utiliza a mesma @Scope das primeias, mas com a diferença de falar o atributo proxyMode, pq quando o contexto do app web é instanciado, não tem solicitação ativa, então o spring vai criar um proxy pra ser injetado como dependencia e vai instanciar o nosso bean quando precisar em uma solicitação.

* Singleton
* Prototype
* Requestb
* Session
* Application
* WebSocket

**Injeção de dependências**

**O que é Injeção de Dependência?**

É uma forma de aplicar inversão de controle em uma classe que utiliza funcionalidades de outras classes.

É uma forma de aplicar inversão de controle, q a gente viu na primeira aula, em uma classe que utiliza funcionalidades de outras classes, tirando a responsabilidade dela de instanciar ou de buscar objetos dessas outras classes q ela depende.

**Autowired**

* Propriedades
* Construtores
* Métodos

**CDI**

O CDI é a especificação Java que define a injeção de dependência e inversão de controle para aplicações JavaEE.