# **Spring Core Framework**

### **Revisão Teórica: Core**

#### **Introdução**

O **Core** de um framework refere-se ao conjunto de funcionalidades essenciais que oferecem suporte ao desenvolvimento de aplicações. No contexto de frameworks como o **Spring**, o Core é responsável por fornecer a infraestrutura necessária para a implementação de conceitos como **Inversão de Controle (IoC)** e **Injeção de Dependência (DI)**. Além disso, ele oferece suporte ao gerenciamento de recursos, manipulação de strings, reflexão e gerenciamento de **classloaders**, que são componentes fundamentais para a criação de aplicações modulares e escaláveis.

#### **Funcionalidades do Core**

As principais funcionalidades do Core incluem:

1. **Contêiner IoC**O Core gerencia o ciclo de vida de objetos (*beans*), incluindo a instanciação, configuração e destruição. O contêiner IoC permite a configuração de dependências entre classes, promovendo um baixo acoplamento e alta coesão.
2. **Gerenciamento de Recursos**Inclui utilitários para acessar arquivos de configuração, mensagens de propriedades e outros recursos da aplicação de maneira centralizada e consistente.
3. **Reflexão**Permite que o framework manipule e interaja com classes em tempo de execução, possibilitando a detecção de anotações, invocação dinâmica de métodos e configuração de objetos.
4. **Manipulação de Strings**Proporciona métodos otimizados para formatar, validar e manipular strings, que são frequentemente usadas em operações como mensagens ao usuário, logs e expressões.
5. **Gerenciamento de ClassLoaders**Facilita o carregamento e a hierarquização de classes, especialmente em ambientes com múltiplos módulos, garantindo que dependências sejam resolvidas corretamente.

#### **Importância do Core**

O Core é o alicerce dos frameworks modernos, permitindo que desenvolvedores se concentrem na lógica de negócio ao abstrair a complexidade da infraestrutura subjacente. Suas principais contribuições são:

* **Redução da Complexidade**: Fornece ferramentas reutilizáveis e padronizadas que agilizam o desenvolvimento.
* **Modularidade**: Assegura que componentes possam ser adicionados ou substituídos sem impacto significativo.
* **Escalabilidade**: Gerencia os recursos e dependências de forma eficiente, facilitando o crescimento da aplicação.

## **Estrutura e Fundamentos**

O Spring Core funciona como um **contêiner IoC** que gerencia os objetos de uma aplicação. Ele utiliza o padrão IoC para controlar o ciclo de vida desses objetos, sendo responsável por instanciar, configurar e gerenciar os objetos (também chamados de *beans*).

A **injeção de dependência** é a técnica pela qual as dependências são fornecidas a esses objetos em tempo de execução, reduzindo o acoplamento entre as classes (FOWLER, 2004).

### **Formas de Configuração no Spring Core**

O Spring Core oferece diferentes formas de configuração de dependências:

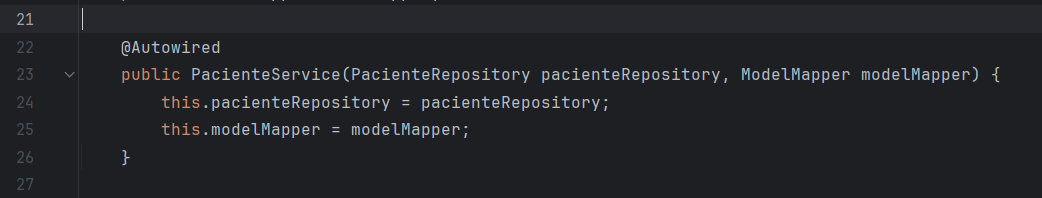
* **XML**: Configuração explícita em arquivos XML.
* **Anotações**: Uso de anotações como @Autowired e @Component para indicar dependências.
* **Configuração Java**: Definição de *beans* por meio de classes de configuração Java, utilizando a anotação @Configuration.

## **Aplicações Práticas**

### **Boas Práticas no Spring Core**

1. **Injeção via Construtor**: Promove imutabilidade e obriga a definição de dependências essenciais.
2. **Uso de Interfaces**: Facilita o desacoplamento e permite a substituição de implementações.
3. **Perfis de Ambiente**: Configurações diferenciadas para ambientes de desenvolvimento, teste e produção, utilizando a anotação @Profile.

## **Exemplo do uso do Spring Core na aplicação de clínica médica**



O código apresentado exemplifica como o Spring Core aplica seus conceitos fundamentais:

### **Injeção de Dependência (DI)**

A anotação @Autowired é uma funcionalidade do Spring Core utilizada para implementar DI. O contêiner IoC detecta as dependências, como PacienteRepository e ModelMapper, e as injeta automaticamente na classe PacienteService.

### **Gerenciamento de Beans**

Os objetos PacienteRepository e ModelMapper são registrados como *beans* no contêiner IoC do Spring. Isso pode ser configurado automaticamente ou via anotações como @Component, @Service ou @Repository.

## **Como o Spring Core Faz Isso?**

1. **Escaneamento de Componentes**: O Spring Core escaneia o código em busca de anotações como @Component, @Service, @Repository e @Configuration.
2. **Injeção Automática**: Ele detecta as dependências necessárias, baseadas no tipo ou no nome, e as injeta automaticamente nas classes que as declaram, utilizando @Autowired ou outras abordagens de DI.
3. **Ciclo de Vida de Beans**: O contêiner gerencia os *beans*, criando-os quando necessário e garantindo que todas as dependências estejam resolvidas antes de disponibilizá-los.

## **Contribuições e Impactos**

O Spring Core reduz a complexidade do desenvolvimento de aplicações Java ao introduzir uma arquitetura baseada em componentes reutilizáveis e altamente testáveis.

Estudos mostram que o padrão IoC adotado pelo Spring oferece maior flexibilidade em comparação ao modelo tradicional **EJB (Enterprise JavaBeans)**, por não requerer um contêiner pesado de aplicação. Isso é especialmente benéfico para aplicações leves e escaláveis (JOHNSON, 2002; FOWLER, 2004).

## **Conclusão**

A revisão teórica sobre o Spring Core destaca sua relevância para o desenvolvimento de software moderno. Conceitos como IoC e DI transformaram as práticas de programação em Java, promovendo modularidade, escalabilidade e manutenção simplificada.

Por meio de configurações flexíveis e boas práticas recomendadas, o Spring Core permanece como um pilar central no ecossistema Java.

## **Referências**

* FOWLER, M. *Inversion of Control Containers and the Dependency Injection Pattern*. Disponível em: https://martinfowler.com/articles/injection.html. Acesso em: 13 dez. 2024.
* JOHNSON, R. *Expert One-on-One J2EE Development without EJB*. Wrox Press, 2002.
* SPRING FRAMEWORK. *Core Container*. Disponível em:<https://spring.io>. Acesso em: 13 dez. 2024.