DESENVOLVIMENTO WEB – BACK-END

AULA 2

Prof.ª Luciane Yanase Hirabara Kanashiro

CONVERSA INICIAL

Nesta abordagem, vamos explorar o ecossistema do Spring, uma poderosa estrutura de desenvolvimento em Java que simplifica a construção de aplicativos empresariais robustos. O Spring framework oferece uma abordagem abrangente, abordando desde o controle de inversão de controle (IoC) até a injeção de dependência. Além disso, conheceremos o Spring Boot, uma extensão que simplifica significativamente o desenvolvimento, eliminando a necessidade de configurações detalhadas. Ao mergulharmos no Spring MVC, entenderemos como esse módulo facilita a criação de aplicativos web, seguindo o padrão Model-View-Controller. Exploraremos também as Anotações Spring, marcadores especiais que simplificam a configuração e o gerenciamento de componentes. Para começar, vamos configurar nosso ambiente de desenvolvimento, utilizando ferramentas como IntelliJ IDEA ou Eclipse e integrando Maven para facilitar o gerenciamento de dependências.

TEMA 1 – SPRING

Spring é um termo amplo que se refere ao ecossistema de projetos e tecnologias relacionados ao desenvolvimento de aplicações em Java. O ecossistema Spring inclui o Spring Framework, o Spring Boot e vários outros projetos e módulos, como Spring MVC, Spring Data, Spring Security, Spring Cloud, entre outros.

Fonte: spring.io.

1.1 Breve história do Spring

A ideia do Spring foi concebida por Rod Johnson, e sua origem está ligada ao livro *Expert One-on-One J2EE Design and Development*, escrito por Johnson e publicado em 2002. Nesse livro, são compartilhadas as experiências e ideias sobre o desenvolvimento de aplicações Java empresariais, com críticas às limitações e complexidades associadas ao modelo de programação Enterprise

JavaBeans (EJB) predominante na época. A seguir, estão resumidos os principais marcos e eventos na história do Spring.

- **Lançamento do Livro *Expert One-on-One J2EE Design and Development* (2002)** – Rod Johnson escreveu o livro mencionado, no qual compartilhou suas ideias e críticas ao modelo EJB e apresentou os princípios fundamentais que mais tarde se tornaram a base do Spring Framework.
- **Lançamento da Primeira Versão (2002)** – A primeira versão do Spring Framework foi lançada em 2002. Nessa versão inicial, o Spring introduziu conceitos como Inversão de Controle (IoC) e Injeção de Dependências (DI) para simplificar o desenvolvimento de aplicações Java empresariais.
- **Expansão e Crescimento (2003-2005)** – O Spring ganhou popularidade rapidamente devido à sua abordagem modular e flexível. A comunidade Spring começou a crescer, e muitos desenvolvedores adotaram o Spring como uma alternativa ao EJB.
- **Introdução de Novos Recursos e Projetos (2005 em diante)** – O Spring Framework continuou a evoluir e expandir seus recursos. Novos projetos e módulos foram introduzidos, como Spring AOP (Aspect-Oriented Programming), Spring MVC (Model-View-Controller) para desenvolvimento web, Spring Security para segurança, entre outros.
- **Aquisição pela VMware e Formação da Pivotal (2009)** – Em 2009, a SpringSource, empresa fundada por Rod Johnson, foi adquirida pela VMware. Posteriormente, a VMware formou a Pivotal Software para continuar o desenvolvimento do ecossistema Spring.
- **Lançamento do Spring Boot (2014)** – O Spring Boot foi introduzido como uma extensão do Spring Framework, projetado para simplificar o desenvolvimento de aplicações Spring, fornecendo configurações padrão sensatas e a capacidade de criar aplicativos autônomos com facilidade.
- **Retorno à VMware (2019)** – A VMware adquiriu novamente a Pivotal Software em 2019, reunindo o ecossistema Spring sob o guarda-chuva da VMware.

1.2 Spring Framework

É muito frequente confundir o Spring framework com o ecossistema Spring. O Spring Framework é o principal projeto dentro do ecossistema Spring. Podemos afirmar que é o framework Java mais popular do mundo e que sua utilização torna a programação Java mais rápida, fácil e segura para todos. O Spring Framework tem foco em velocidade, simplicidade e produtividade. Ele é projetado para ser altamente modular, promover a inversão de controle (IoC) e incentivar boas práticas de programação, como a programação orientada a aspectos (AOP). Sendo modular, permite que os desenvolvedores escolham os módulos específicos que desejam usar em suas aplicações. A configuração tradicional do Spring Framework pode envolver arquivos XML ou anotações Java.

Embora alguns considerem o Java EE e seu sucessor moderno Jakarta EE competindo com o Spring, eles são na verdade complementares. O modelo de programação Spring não abrange a especificação da plataforma Java/Jakarta EE, em vez disso, integra-se com especificações individuais cuidadosamente selecionadas do guarda-chuva tradicional de EE.

1.2.2 Características

A seguir, estão listadas as principais características do Spring Framework.

- **Tecnologias principais** – injeção de dependência, eventos, recursos, i18n, validação, vinculação de dados, conversão de tipo, SpEL, AOP. Abaixo estão descritas sucintamente cada tecnologia:
    - **Inversão de Controle (IoC)** – No IoC, o controle do fluxo de execução é invertido, o que significa que o framework é responsável por gerenciar a criação e o ciclo de vida dos objetos (beans), permitindo a fácil configuração e interconexão desses objetos.
    - **Injeção de Dependências (DI)** – A IoC no Spring é implementada por meio da Injeção de Dependências. Isso permite que os beans declarem suas dependências, e o Spring as injeta durante a inicialização da aplicação. Isso promove a reutilização de componentes e torna as aplicações mais flexíveis e desacopladas.

- **Eventos** – O Spring suporta um modelo de eventos que permite a comunicação entre componentes da aplicação. Os eventos podem ser publicados e ouvidos, proporcionando uma maneira flexível de lidar com a comunicação dentro do sistema.
- **Recursos** – O Spring facilita o acesso a recursos, como arquivos e URLs, por meio de uma abstração simplificada. Isso é útil para lidar com configurações, imagens, arquivos de propriedades etc.
- **I18n (Internacionalização)** – O Spring oferece suporte à internacionalização, permitindo que os aplicativos sejam adaptados para diferentes idiomas e regiões.
- **Validação** – O Spring fornece um mecanismo de validação para objetos, permitindo a aplicação de regras de validação e o tratamento de erros de forma consistente.
- **Vinculação de Dados** – O Spring facilita a vinculação de dados entre objetos de domínio e a interface do usuário, especialmente em aplicativos web.
- **Conversão de Tipo** – O Spring suporta conversão de tipos automática, facilitando a manipulação de diferentes tipos de dados em aplicações.
- **SpEL (Spring Expression Language)** – O SpEL é uma linguagem de expressão poderosa usada em várias partes do Spring para avaliação de expressões, manipulação de dados e configuração.
- **AOP (Aspect-Oriented Programming)** – O AOP no Spring permite a separação de preocupações transversais, como logging e transações, tornando o código mais modular e fácil de manter.

- **Testes** – objetos simulados, estrutura TestContext, teste Spring MVC, WebTestClient. A seguir, estão descritos sucintamente cada componente de testabilidade:
  - **Objetos Simulados** – O Spring facilita a criação de objetos simulados (mocks) para testes unitários, permitindo isolar o código que está sendo testado.
  - **Estrutura TestContext** – TestContext é uma estrutura do Spring que fornece informações e suporte ao contexto para testes, facilitando a execução de testes de integração.

- **Teste Spring MVC** – O Spring fornece suporte para testes de aplicativos Spring MVC, permitindo testar controladores, fluxos de solicitações e respostas HTTP.
- **WebTestClient** – No contexto do Spring WebFlux, o WebTestClient é uma ferramenta de teste para aplicativos reativos, permitindo a execução de testes de integração em controladores reativos.

- **Acesso a dados** – transações, suporte DAO, JDBC, ORM, Marshalling XML, estruturas web Spring MVC e Spring WebFlux.
  - **Transações** – O Spring oferece suporte a transações declarativas, permitindo que os desenvolvedores definam transações usando anotações ou configurações XML.
  - **Suporte DAO (Data Access Object)** – O Spring facilita a implementação do padrão DAO, fornecendo suporte para acesso a dados de maneira coesa e modular.
  - **JDBC (Java Database Connectivity)** – O Spring simplifica o acesso a bancos de dados relacionais usando JDBC, oferecendo funcionalidades adicionais e tratando exceções de forma mais eficaz.
  - **ORM (Object-Relational Mapping)** – O Spring oferece integração com frameworks de mapeamento objeto-relacional, como Hibernate, simplificando o acesso a dados usando objetos de domínio.
  - **Marshalling XML** – O Spring facilita a serialização e desserialização de objetos Java para e a partir de XML usando diferentes tecnologias, como JAXB.
  - **Spring MVC (Model-View-Controller)** – O Spring MVC é um framework de desenvolvimento web baseado no padrão MVC, permitindo a construção de aplicativos web escaláveis e modulares.
  - **Spring WebFlux** – O Spring WebFlux é uma estrutura reativa que oferece suporte à programação reativa no desenvolvimento web, especialmente adequado para aplicativos orientados a eventos e com grande concorrência.
- **Integração** – comunicação remota, JMS, JCA, JMX, e-mail, tarefas, agendamento, cache e observabilidade.
  - **Comunicação Remota** – O Spring facilita a comunicação remota entre aplicativos, fornecendo suporte para RMI (Remote Method Invocation), HTTP remoto, entre outros.

- **JMS (Java Message Service)** – O Spring oferece suporte à integração com o JMS para mensagens assíncronas e comunicação entre sistemas distribuídos.
- **JCA (Java Connector Architecture)** – O Spring suporta a integração com adaptadores JCA para facilitar a integração com sistemas corporativos.
- **JMX (Java Management Extensions)** – O Spring oferece suporte ao JMX para gerenciar e monitorar aplicativos Java em tempo de execução.
- **E-mail** – O Spring facilita o envio de e-mails e integra-se com serviços de e-mail.
- **Tarefas e Agendamento** – O Spring fornece suporte para agendamento de tarefas, permitindo a execução de tarefas agendadas em intervalos específicos.
- **Cache** – O Spring oferece suporte a caching, permitindo melhorar o desempenho de operações frequentes ao armazenar resultados em cache.
- **Observabilidade** – O Spring oferece recursos para observabilidade, incluindo monitoramento de métricas, rastreamento de solicitações e registro.

- **Idiomas** – Kotlin, Groovy, linguagens dinâmicas. O Spring oferece suporte a diferentes linguagens, como Kotlin e Groovy, permitindo que os desenvolvedores escolham a linguagem que melhor se adapta às suas necessidades e preferências.

TEMA 2 – Spring Boot

Se o Spring Framework já trouxe rapidez no desenvolvimento de aplicações Java, podemos dizer que o surgimento do Spring Boot tornou o desenvolvimento Java ainda mais fácil. O Spring Boot faz parte do ecossistema Spring. A diferença principal entre Spring Framework e Spring boot é que, no Spring framework, as configurações são feitas de forma manual. O Spring Boot introduz o conceito de configuração automática, reduzindo a quantidade de configuração manual necessária. O Spring Boot fornece configurações predefinidas para muitas tecnologias, permitindo que o desenvolvedor inicie rapidamente sem a necessidade de muita configuração.

2.1 Característica

- **Criação de Aplicativos Spring Independentes** – O Spring Boot simplifica o processo de criação de aplicativos Spring autônomos. Isso significa que os aplicativos resultantes podem ser executados como JARs executáveis ou WARs tradicionais, sem a necessidade de configuração externa complexa.
- **Incorporação do Tomcat, Jetty ou Undertow Diretamente** – O Spring Boot permite a incorporação direta de servidores web, como Tomcat, Jetty ou Undertow, no próprio aplicativo. Isso elimina a necessidade de implantar arquivos WAR em servidores externos, tornando o processo de implantação mais simples.
- **Dependências 'Iniciais' Opinativas** – O Spring Boot oferece uma variedade de starters (dependências iniciais) para diferentes finalidades, como desenvolvimento web, acesso a dados, segurança, entre outros. Essas starters são pré-configuradas para simplificar o processo de compilação e configuração do aplicativo.
- **Configuração Automática de Bibliotecas Spring e de Terceiros** – O Spring Boot adota uma abordagem de "convenção sobre configuração", configurando automaticamente muitos aspectos do aplicativo com base nas bibliotecas presentes no classpath. Isso reduz a necessidade de configurações manuais extensas, permitindo que os desenvolvedores foquem mais na lógica de negócios.

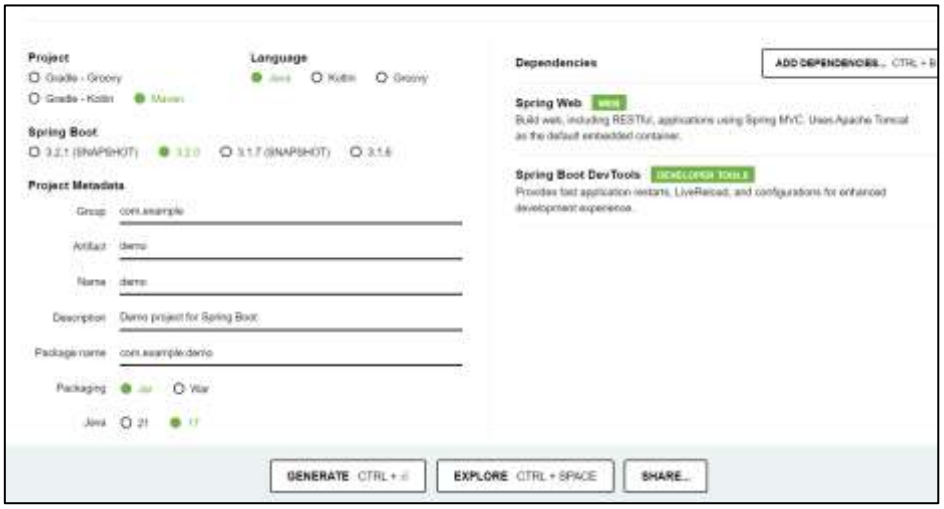
O Spring boot visa reduzir a sobrecarga de configuração manual e acelerar o desenvolvimento de aplicações Java, permitindo que o desenvolvedor se concentre na programação.

2.2 Spring initializr

O Spring Initializr é uma ferramenta on-line que simplifica significativamente o processo de inicialização e configuração de projetos Spring Boot. Ele permite que sejam rapidamente geradas a estrutura básica de um projeto Spring Boot com as dependências desejadas.

A seguir, temos a tela do Spring initializr com o projeto Maven selecionado, Linguagem Java, versão 3.2.0 do Spring Boot e Java 17

selecionado. Na figura, é possível visualizar duas dependências: Spring Web e Spring Boot Dev Tools. Podemos observar ainda que, dentre as opções para a versão do Spring Boot, algumas versões estão com a palavra *SNAPSHOT* entre parêntesis. Uma versão snapshot é uma versão que ainda está em desenvolvimento ativo e não foi lançada oficialmente como uma versão estável. Ao utilizar versões snapshot, os desenvolvedores precisam estar cientes de que essas versões podem ser instáveis ou sujeitas a mudanças frequentes, uma vez que estão em desenvolvimento ativo. Portanto, ao construir e implantar aplicações para ambientes de produção, é recomendável usar versões estáveis e testadas.



TEMA 3 – Spring MVC

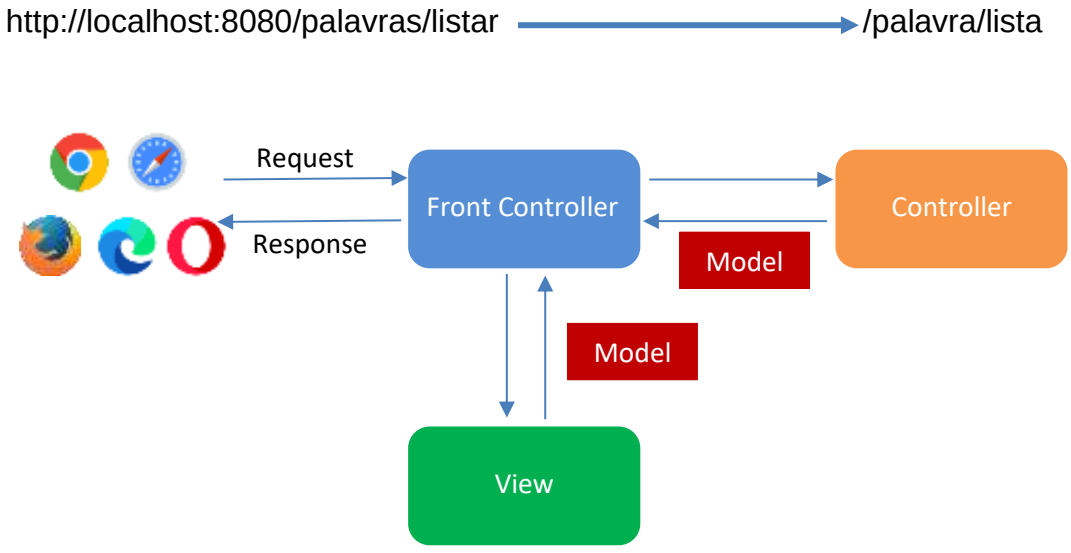
Spring Web MVC é um framework web e podemos dizer que foi incluído no Spring Framework desde o início de sua criação. O nome formal, *Spring Web MVC*, vem do nome de seu módulo de origem (spring-webmvc), mas é mais comumente conhecido como *Spring MVC*. O spring MVC facilita o desenvolvimento de aplicativos web baseados no padrão arquitetural MVC. Ele é uma parte do Spring framework que ajuda na manipulação de request e response HTTP.

Como vimos anteriormente, o padrão MVC divide uma aplicação em três componentes principais: Model, View e Controller. Já descrevemos exaustivamente essas camadas, mas aqui faz se necessário relembrar mais uma vez essa definição e descrever as camadas relacionando com o Spring MVC. A seguir é descrita cada camada do MVC no contexto do Spring MVC:

- **Model (Modelo)** – O Model representa a lógica de negócios e os dados da aplicação. Ele lida com a manipulação e a validação dos dados, além de conter a lógica de negócios específica da aplicação. No Spring MVC, o Model é frequentemente implementado como objetos Java (POJOs) ou beans gerenciados pelo Spring.
- **View (Visão)** – A View é responsável por exibir os dados ao usuário e interagir com ele. Ela apresenta os dados fornecidos pelo Model e pode ser uma página HTML, uma página JSP (JavaServer Pages), ou qualquer outra representação visual. O Spring MVC suporta vários tipos de Views, incluindo JSP, Thymeleaf, FreeMarker e outros.
- **Controller (Controlador)** – O Controller é responsável por receber as solicitações do usuário, interagir com o Model para obter ou modificar dados e selecionar a View apropriada para renderizar a resposta. Ele atua como o intermediário entre a parte visual (View) e a lógica de negócios (Model). No Spring MVC, os controllers são geralmente implementados como classes Java e são identificados por anotações, como @Controller.

O Spring MVC é uma implementação específica do padrão arquitetural MVC, temos assim que o modelo MVC do Spring apresenta algumas diferenças quanto ao modelo MVC genérico. A figura a seguir ilustra o funcionamento do Spring MVC. A seguir é apresentada a explicação de como esse modelo funciona.

Modelo MVC do Spring MVC



Fonte: Luciane Yanase Hirabara Kanashiro

O Front Controller funciona como dispatcher servlet, ou seja, recebe as requisições que vem do navegador. O dispatcher servlet serve como um ponto de entrada único para todas as solicitações que chegam à aplicação web, sendo responsável por receber todas as solicitações HTTP e encaminhá-las para os controladores adequados.

No Spring MVC, a camada de controle é composta pelos controladores (Controllers), a gente pode ter várias classes de controle. Os controladores são responsáveis por receber solicitações do DispatcherServlet, processar a lógica de negócios associada e interagir com a camada de modelo (Model). Cada controlador é responsável por tratar um conjunto específico de solicitações e coordenar o fluxo correspondente da aplicação.

A Model (camada de modelo) é responsável por gerenciar o estado dos objetos de negócios e fornecer dados para a camada de visualização (View) por meio dos controladores. Responsável também pelas regras de negócios e persistência de dados.

A View template faz a resolução das páginas (html,jsp). Tem a responsabilidade de representar os dados fornecidos pela camada de modelo (Model) de uma forma que seja adequada para a apresentação ao usuário final. Isso pode incluir a geração de HTML, XML, JSON, ou qualquer outro formato de resposta que seja necessário para a interação com o cliente. A camada de visualização interage com os controladores (Controllers) para receber dados da camada de modelo. Ela utiliza esses dados para gerar as respostas apropriadas, mantendo uma separação clara entre a lógica de controle e a apresentação.

TEMA 4 – ANOTAÇÕES DO SPRING

Os desenvolvedores Java conhecem bem o conceito de anotações. As anotações adicionam metadados no código-fonte. Elas funcionam como marcadores especiais que podem ser aplicados a pacotes, tipos, métodos, construtores, parâmetros e outros elementos de código para fornecer informações adicionais sobre seu comportamento ou propósito.

Os Metadados referem-se a dados que fornecem informações sobre outros dados. No contexto de Java, as anotações (como @Override, @Deprecated, @Entity etc.) são metadados que fornecem informações adicionais sobre classes, métodos, campos e outros elementos do código-fonte.

No código a seguir, é mostrado um exemplo da utilização da anotação @Override na sobrescrita do método toString.

```
@Override
	public String toString() {
		return "Pessoa [nome=" + nome + ", idade=" + idade +
"]";
	}
```

A definição para anotações no Spring não é diferente. As anotações Spring são marcadores especiais usados para fornecer metadados e instruções ao contêiner do Spring. Elas são usadas para simplificar a configuração, facilitar a injeção de dependência e expressar várias configurações em um formato mais conciso e declarativo.

O Spring Framework faz amplo uso de anotações para simplificar a configuração e o desenvolvimento de aplicações. Nas subseções a seguir, são apresentadas algumas das principais anotações do Spring.

4.1 Anotações

4.1.1 Anotações para Configuração

As anotações para configuração do Spring são utilizadas para simplificar e organizar a configuração de aplicações Spring Framework. Elas são usadas para marcar classes, métodos e campos com metadados que o Spring usa para configurar o comportamento da aplicação. Essas anotações ajudam a reduzir a quantidade de código de configuração XML que era comum em versões mais antigas do Spring.

- @Configuration – Indica que uma classe serve como uma fonte de definições de bean para o contexto de aplicação.
- @Bean – Utilizada em métodos dentro de classes @Configuration para indicar que o método produz um bean gerenciado pelo Spring.
- @ComponentScan – Define o escopo dos pacotes a serem escaneados em busca de componentes anotados, como @Component, @Service, @Repository etc.
- @Import – Importa uma ou mais classes de configuração.

4.1.2 Anotações para Gerenciamento de Componentes

As anotações para gerenciamento de componentes no Spring são usadas para facilitar a configuração e o uso de componentes dentro de uma aplicação Spring. Elas são utilizadas para marcar classes como componentes gerenciados pelo container do Spring, permitindo que essas classes sejam automaticamente detectadas e configuradas pelo Spring.

- @Component – Indica que uma classe é um componente e deve ser gerenciada pelo contêiner Spring.
- @Service – Especialização de @Component para indicar que a classe é um serviço.
- @Repository – Especialização de @Component para indicar que a classe é um repositório, geralmente usados para acesso a dados.
- @Controller – Indica que uma classe é um controlador em uma aplicação Spring MVC.

4.1.3 Anotações para Injeção de Dependência

As anotações para injeção de dependência no Spring são usadas para facilitar e automatizar a injeção de dependências em classes. A injeção de dependência é um padrão de projeto importante no desenvolvimento de software, que consiste em fornecer as dependências necessárias a um objeto de forma externa, em vez de criá-las internamente. Isso torna o código mais modular, testável e fácil de manter.

- @Autowired – Indica que um campo, método setter ou construtor deve ser injetado automaticamente pelo Spring.
- @Qualifier – Especifica o nome qualificador quando há várias implementações possíveis de uma interface.
- @Value – Injeta valores diretamente em campos ou métodos.

4.1.4 Anotações para Ciclo de Vida do Bean

As anotações para ciclo de vida do bean no Spring são usadas para controlar o ciclo de vida dos objetos gerenciados pelo container do Spring, ou seja, os beans. O ciclo de vida de um bean no Spring inclui sua inicialização, uso e possível destruição. As anotações permitem que você execute métodos específicos em determinados pontos do ciclo de vida do bean, como antes ou depois da inicialização, antes ou depois da destruição, entre outros.

- @PostConstruct – Método anotado com @PostConstruct é executado após a criação do bean.
- @PreDestroy – Método anotado com @PreDestroy é executado antes da destruição do bean.
- @Lazy – Esta anotação é usada para indicar que um bean deve ser inicializado de forma preguiçosa, ou seja, só será criado quando for requisitado pela primeira vez.

4.1.5 Anotações para Web

As anotações para web do Spring são usadas para simplificar o desenvolvimento de aplicações web utilizando o Spring Framework. Elas são usadas para configurar controladores, mapear URLs para métodos específicos, lidar com requisições HTTP, entre outras funcionalidades relacionadas à camada web da aplicação.

- @RequestMapping – Mapeia solicitações HTTP para métodos de manipulação em controladores Spring MVC.
- @PathVariable – Indica que um parâmetro do método deve ser vinculado a uma variável de modelo de caminho.
- @RequestParam – Vincula parâmetros de solicitação a parâmetros de método em métodos de controlador.
- @ResponseBody – Indica que o valor de retorno do método deve ser serializado diretamente na resposta HTTP.
- @ResponseStatus – Define o código de status HTTP e, opcionalmente, a razão a ser retornada para a resposta.
- @ModelAttribute – Vincula um método ou parâmetro de método a um atributo de modelo.

4.1.6 Anotações para Tratamento de Exceções

As anotações para tratamento de exceções no Spring são utilizadas para lidar com exceções de forma mais estruturada e elegante em aplicações Spring. Elas permitem que você capture exceções e tome ações específicas, como retornar uma mensagem de erro personalizada, redirecionar para uma página de erro, ou realizar alguma outra ação de tratamento de exceção adequada.

- @ControllerAdvice – Agrupa anotações @ExceptionHandler, @InitBinder e @ModelAttribute em uma classe para ser compartilhada entre todos os controladores.
- @ExceptionHandler – Define um método para manipular exceções lançadas por métodos de controlador.

4.1.7 Anotações para Transações

As anotações para transações no Spring são utilizadas para controlar transações em métodos ou classes que realizam operações em bancos de dados. Elas permitem definir de forma declarativa o comportamento transacional de métodos específicos, garantindo que as operações sejam executadas de forma consistente e segura, seguindo os princípios ACID (Atomicidade, Consistência, Isolamento e Durabilidade).

- @Transactional – Indica que um método deve ser envolto em uma transação.

4.1.8 Anotações para Testes

As anotações para testes no Spring são utilizadas para facilitar a escrita e execução de testes automatizados em aplicações Spring. Elas permitem configurar o contexto de teste, injetar dependências, executar código antes e depois dos testes, entre outras funcionalidades relacionadas a testes.

- @RunWith(SpringRunner.class) – Utilizada em classes de teste para integração com o Spring durante a execução de testes.
- @SpringBootTest – Indica que a classe de teste é uma classe de teste do Spring Boot.

Essas são apenas algumas das principais anotações do Spring. O framework oferece uma variedade de anotações que abrangem diferentes aspectos do desenvolvimento de aplicações, desde a configuração até a manipulação de solicitações web, injeção de dependência, tratamento de exceções, entre outros. A seguir, temos um exemplo de utilização das anotações em uma classe de serviço.

```
@Service @Transactional(readOnly=false)

public class PalavraServiceImpl implements PalavraService {

	@Autowired
	private PalavraDao dao;

	//restante do código

}
```

TEMA 5 – AMBIENTE DE DESENVOLVIMENTO

Agora que conhecemos um pouco sobre o ecossitema Spring, a diferença entre Spring framework e Spring Boot, vamos ver sobre o ambiente de desenvolvimento para se implementar um sistema utilizando Spring Boot.

Para desenvolver uma aplicação utilizando o Spring Boot, você precisará configurar um ambiente de desenvolvimento adequado. Veremos nas subseções a seguir as ferramentas necessárias para Desenvolvimento com Spring Boot.

5.1 Ferramentas

5.1.1 JDK

O Java Development Kit (JDK) entra como uma ferramenta básica para se desenvolver qualquer aplicativo em Java. Sendo assim, para programar em Java, é necessário ter o JDK instalado. O JDK inclui :

- JRE: para se ter a máquina virtual e bibliotecas necessárias para executar um programa java é necessário ter o JRE.
- Javac: o Javac é o compilador da linguagem Java.
- Javadoc: Javadoc é ferramenta de documentação do Java.
- Jdb *(Java Debugger)*: Jdb é a ferramenta de depuração.

- APIs: conjunto de serviços da linguagem Java.

5.1.2 IDE (Ambiente de Desenvolvimento Integrado) - Eclipse

Outra ferramenta básica e imprescindível para desenvolvimento de software é a IDE. Existem muitas opções populares que incluem: IntelliJ IDEA, Eclipse e Spring Tool Suite (STS). Veremos aqui sobre o Eclipse por ser uma ferramenta popular entre os desenvolvedores Java.

Fonte: eclipse.org.

O Eclipse é uma IDE (Integrated Development Environment ou Ambiente de Desenvolvimento Integrado) de código aberto que fornece um conjunto de ferramentas e recursos para o desenvolvimento de software em várias linguagens de programação. Amplamente utilizado para o desenvolvimento Java, oferece suporte a projetos Java, Maven, entre outros. Além disso, ele se integra ao compilador Java, fornecendo compilação e execução de aplicativos Java diretamente na IDE.

Embora seja mais conhecido pelo suporte ao desenvolvimento Java, o Eclipse oferece suporte a várias linguagens de programação como: C/C++, Python, PHP e Ruby. Pode se instalar plug-ins específicos para cada linguagem.

A Interface Gráfica (GUI) é amigável, de modo a organizar ferramentas e recursos em perspectivas e visualizações, fornecendo uma experiência integrada para codificação, depuração, teste e gerenciamento de projetos.

Quanto ao editor de código-fonte, este oferece recursos avançados, como realce de sintaxe, formatação automática, dicas de código, navegação rápida entre classes e métodos e suporte a refatoração.

O Eclipse é uma escolha popular entre desenvolvedores pela sua flexibilidade, sendo altamente extensível por meio de plugins, oferecendo ainda suporte para o desenvolvimento de aplicativos web com plug-ins para frameworks como Spring, JavaServer Faces (JSF), e outros. Há também suporte a HTML, CSS e JavaScript.

5.1.3 Configuração do Projeto: Spring initializr

Fonte: spring.io.

Já mencionamos o Spring initializr anteriormente. Ele é uma ferramenta on-line que facilita a inicialização e configuração de projetos Spring Boot. Sendo assim, para citar um projeto Spring Boot com as dependências desejadas, basta acessar o Spring Initializr em <https://start.spring.io/> e selecionar as dependências com base nas necessidades do seu projeto, como por exemplo: Spring Web, Spring Data JPA, Thymeleaf (para templates HTML), etc.

5.1.4 Desenvolvimento e Execução: Maven

A codificação da aplicação pode ser facilmente Iniciada utilizando o Spring initializr. Como vimos anteriormente, ele já gera rapidamente um arquivo de inicialização e configuração. Depois, pode começar a escrever seu código com classes anotadas com @Controller para criar controladores web, @Service para serviços, e assim por diante. Para esse rápido gerenciamento e execução do Projeto, o Maven é utilizado.

Fonte: maven.apache.org.

Apache Maven é uma ferramenta de gerenciamento de construção e automação de projetos utilizada principalmente para projetos Java. Com base no conceito de modelo de objeto de projeto (POM), o Maven pode gerenciar a construção, os relatórios e a documentação de um projeto a partir de uma informação central.

O chamado *POM* é um arquivo xml para descrever as configurações do projeto. O POM inclui informações sobre dependências, plugins, configurações de compilação, e outros detalhes do projeto. As dependências do projeto são

gerenciadas automaticamente. Os artefatos (bibliotecas, JARs) necessários para o projeto são baixados automaticamente de repositórios remotos, como o Maven Central. O Maven Central é o repositório público mais utilizado, mas você também pode configurar repositórios privados ou locais. Maven segue AINDA o princípio de "Convenção sobre Configuração" (Convention over Configuration). Isso significa que, em muitos casos, as configurações padrão são suficientes, mas também é possível personalizar quando necessário.

5.1.5 Desenvolvimento e Execução: Apache Tomcat

Fonte: tomcat.apache.org.

O Apache Tomcat é popularmente conhecido pelos desenvolvedores apenas como *Tomcat,* e segundo a página da própria Apache Software Foundation, o Tomcat é uma implementação de código aberto das especificações Jakarta Servlet, Jakarta Server Pages, Jakarta Expression Language, Jakarta WebSocket, Jakarta Annotations e Jakarta Authentication. Estas especificações fazem parte da plataforma Jakarta EE.

Simplificando, ele é um servidor web de código aberto e contêiner de servlets, projetado para ser compatível com os padrões Java EE (agora Jakarta EE) proporcionando um ambiente de execução leve e eficiente para aplicativos web.

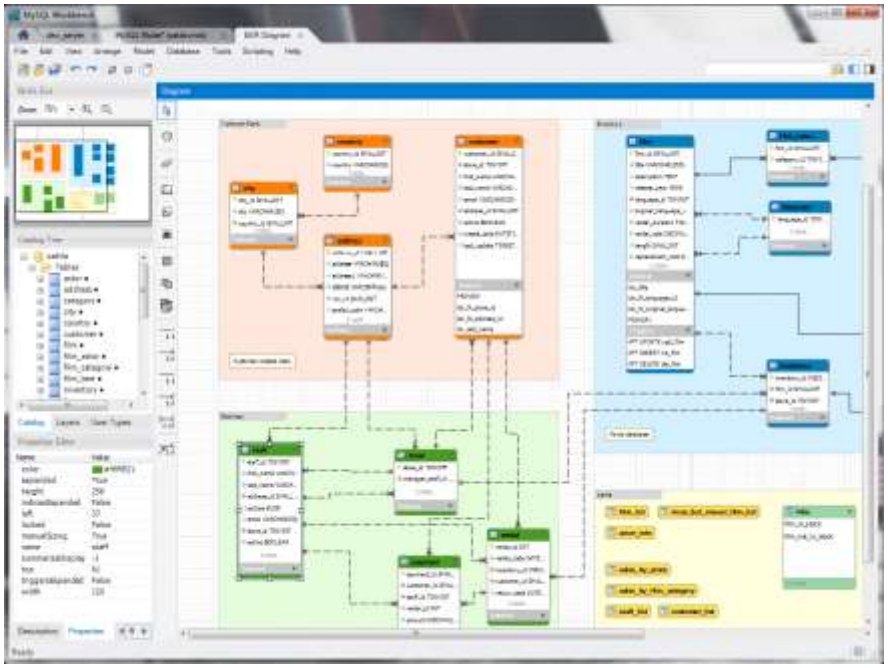
A execução do projeto pode ser feita a partir da IDE ou usando ferramentas como o Maven ou Gradle. A aplicação Spring Boot incorpora um servidor embutido (como Tomcat) para facilitar o desenvolvimento e teste.

5.1.6 Banco de Dados: MySQL Workbench

Quando um aplicativo precisar de persistência de dados, deve ser configurada uma conexão com um banco de dados. Spring Boot oferece suporte a uma variedade de bancos de dados, como MySQL, PostgreSQL, H2 (para desenvolvimento), etc. Nessa seção, abordaremos as características do Mysql Workbench.

Fonte: mysql.com.

MySQL Workbench é uma ferramenta visual unificada para o sistema de gerenciamento de banco de dados MySQL. Foi desenvolvido pela Oracle e está disponível para Windows, Linux e Mac OS X. Oferecendo uma interface gráfica que simplifica o processo de criação, modelagem, manutenção e administração de bancos de dados MySQL, o Workbench permite ainda que os desenvolvedores visualizem e projetem esquemas de banco de dados usando um editor gráfico. Isso inclui a criação de tabelas, definição de relacionamentos, e modelagem de dados. A engenharia Reversa também está presente nessa ferramenta, sendo possível gerar diagramas de classe a partir de um banco de dados existente. A figura a seguir ilustra a janela de design do Banco de Dados.



Fonte: mysql.com.

5.1.7 Banco de Dados: Spring Data JPA

O Spring Data JPA pertence à família Spring e facilita a implementação de repositórios baseados em JPA (Java Persistence API). O Spring Data JPA

torna mais fácil construir aplicativos com Spring que usam tecnologias de acesso a dados. Ele é utilizado para interagir com o banco de dados de maneira simplificada, permitindo a rápida configuração das entidades e os repositórios para realizar operações de CRUD. A seguir, temos um código com exemplo e anotação @Repository.

```
@Repository
public class PalavraDaoImpl extends AbstractDao<Palavra, Long>
implements PalavraDao {


}
```

5.1.8 Testes Unitários e de Integração: JUnit

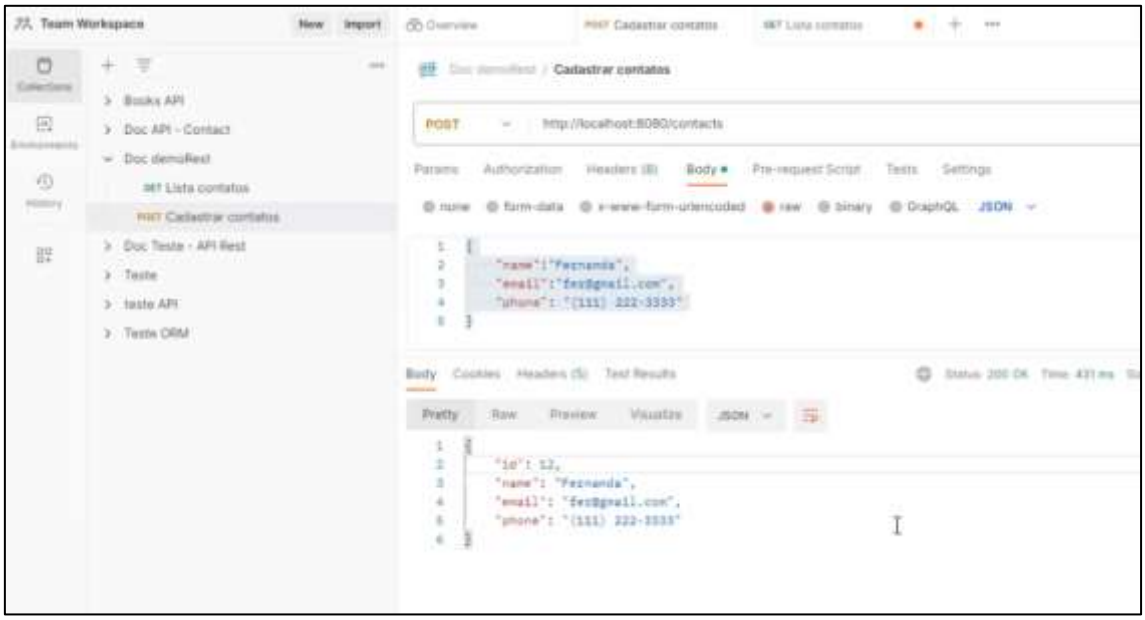
Os testes unitários e de integração são realizados para garantir que o código funcione conforme o esperado. O Spring Boot fornece suporte para testes com JUnit. O JUnit é um framework de teste para a linguagem de programação Java. Ele fornece um ambiente e um conjunto de anotações que facilitam a criação e execução de testes automatizados para código Java. O JUnit é amplamente utilizado na prática de Desenvolvimento Orientado a Testes (TDD) e é uma ferramenta muito utilizada para garantir a qualidade do software.

5.1.9 Ferramentas Complementares: Postman

Para testar os endpoints da aplicação, são utilizadas ferramentas de teste de API como Postman ou Insomnia . Veremos um pouco mais sobre o Postman a seguir.

Fonte: Postman, S.d.

O Postman é uma plataforma API usada por desenvolvedores para testar APIs (Interfaces de Programação de Aplicações). Ele fornece uma interface gráfica amigável para enviar solicitações HTTP e visualizar as respostas, facilitando o teste e a depuração de APIs. Muitos desenvolvedores utilizam o Postman também para fazer a documentação da API. Abaixo vemos uma tela do postman para uma requisição POST.



FINALIZANDO

Encerramos este conteúdo abrangendo aspectos essenciais do ecossistema Spring, destacando a sua importância no desenvolvimento de aplicações Java. O Spring Framework, com sua arquitetura modular e flexível, proporciona uma base sólida para a construção de sistemas empresariais escaláveis. Ao introduzir o Spring Boot, percebemos como a criação de aplicativos Java torna-se ainda mais simplificada, com uma configuração opinativa e a capacidade de gerar aplicativos autocontidos. Aprofundamos nosso entendimento no modelo MVC do Spring destacando a clara separação de responsabilidades entre a camada de modelo, visualização e controle. Vimos as Anotações Spring, uma ferramenta poderosa para simplificar a configuração e melhorar a legibilidade do código.

Além disso, ao considerar o ambiente de desenvolvimento, exploramos a facilidade proporcionada por ferramentas como Eclipse, aliadas à praticidade do

Spring Initializr. Esses elementos, combinados, formam uma base sólida para a construção eficiente de aplicações Java robustas e modernas. Estamos prontos para explorar ainda mais as possibilidades deste ecossistema dinâmico e continuar a aprimorar nossas habilidades de desenvolvimento.

REFERÊNCIAS

APACHE. Apache Tomcat. Disponível em: <https://tomcat.apache.org/>. Acesso em: 3 abr. 2024.

DEITEL, P. J.; DEITEL, H. M. **Java**: como programar. 10. ed. São Paulo, SP: Pearson, 2017.

MAVEN. Apache Maven Project. Disponível em: <https://maven.apache.org/>. Acesso em: 3 abr. 2024.

MYSQL. Mysql workbench. Disponível em: <https://www.mysql.com/products/workbench/>. Acesso em: 3 abr. 2024.

POSTMAN. Disponível em: <https://www.postman.com/>. Acesso em: 3 abr. 2024.

SPRING. Spring Framework Documentation. Disponível em: <https://docs.spring.io/spring-framework/reference/>. Acesso em: 3 abr. 2024.