

## Aula 1: Fundamentos de Arquitetura e a Plataforma Java EE

A primeira aula estabelece a base para o desenvolvimento de software, diferenciando conceitos essenciais e introduzindo a plataforma Java Enterprise Edition (Java EE).

### Arquitetura de Software e Padrões de Projeto

- **Arquitetura de Software:** Refere-se à estrutura fundamental e de alto nível de um sistema, compreendendo seus componentes, relacionamentos e princípios de design. Exemplos incluem arquitetura de três camadas e microserviços.
- **Padrões de Projeto:** São soluções reutilizáveis para problemas recorrentes e de baixo nível no design de software, focando na implementação de classes e objetos. Exemplos notáveis são Singleton, Observer e MVC (Model-View-Controller). A colaboração entre uma boa arquitetura e padrões de projeto resulta em sistemas robustos, flexíveis e de fácil manutenção.

### Arquitetura Multicamadas

- **Modelo Cliente-Servidor:** Uma arquitetura de duas camadas que se tornou popular nos anos 1990, onde o cliente lidava com a interface do usuário e o servidor, geralmente um banco de dados, armazenava os dados.
- **Modelo de 3 Camadas:** Surgiu com a necessidade de aplicações web mais complexas e divide o sistema em três camadas lógicas principais:
  - **Camada de Apresentação:** Responsável pela interação com o usuário, exibindo informações e traduzindo comandos do usuário.
  - **Camada de Domínio (ou Lógica de Negócios):** Contém o trabalho principal da aplicação, como cálculos e validações.
  - **Camada de Dados:** Comunica-se com outros sistemas, principalmente com o banco de dados para armazenamento persistente.
- **Diferença entre Camada Lógica (Layer) e Física (Tier):** É comum confundir os termos. **Layer** refere-se à divisão funcional e lógica do software, enquanto **Tier** implica uma separação física, onde cada camada é executada em uma máquina ou infraestrutura diferente.

### Plataforma Java EE e Jakarta EE

- **Arquitetura Java EE:** É uma plataforma para construir aplicações corporativas em Java, usando um modelo distribuído de múltiplas camadas. Embora possa ser descrita com quatro camadas lógicas (Cliente, Web, Negócios e Sistema de Informação Corporativa - EIS), é considerada uma aplicação de três camadas

(three-tier) por ser distribuída em três locais: máquinas clientes, servidor Java EE e o banco de dados.

- **Evolução para Jakarta EE:** A Oracle transferiu as tecnologias Java EE para a Eclipse Foundation, resultando no **Eclipse Jakarta EE**, que é a continuação do Java EE sob um novo nome e governança comunitária. Para os programadores, a transição é gradual, mas mudanças se tornaram mais visíveis a partir de 2022, com a alteração de pacotes de javax para jakarta.

### *Revisão de Java Básico*

A aula termina revisitando conceitos fundamentais de Java, como a classe Object e seus métodos toString(), equals() e hashCode(), além de outras classes essenciais como System, Scanner, String, Math, Classes Wrapper e o Collection Framework.

## **Aula 2: Ecossistema Spring e Ambiente de Desenvolvimento**

A segunda aula foca no ecossistema Spring, uma estrutura poderosa que simplifica o desenvolvimento de aplicações Java, e detalha as ferramentas necessárias para configurar um ambiente de desenvolvimento.

### *Spring Framework e Spring Boot*

- **Spring Framework:** É o projeto principal do ecossistema Spring, focado em velocidade, simplicidade e produtividade. Suas características principais incluem Inversão de Controle (IoC), Injeção de Dependências (DI), Programação Orientada a Aspectos (AOP), e módulos para acesso a dados, testes, integração e desenvolvimento web (Spring MVC).
- **Spring Boot:** É uma extensão do Spring Framework projetada para simplificar ainda mais o desenvolvimento. Ele introduz o conceito de **configuração automática**, eliminando a necessidade de configurações manuais extensas. Permite criar aplicações autônomas com servidores web embutidos (como Tomcat) e utiliza "starters" para gerenciar dependências de forma opinativa.

### *Spring MVC*

- **Padrão MVC no Spring:** O Spring MVC é um módulo do Spring Framework para criar aplicações web seguindo o padrão Model-View-Controller.
  - **Model:** Representa a lógica de negócios e os dados da aplicação.
  - **View:** Responsável por exibir os dados ao usuário, utilizando tecnologias como JSP ou Thymeleaf.

- **Controller:** Recebe as requisições do usuário, interage com o Model e seleciona a View apropriada para a resposta.
- **Funcionamento:** Um componente central chamado **DispatcherServlet (Front Controller)** recebe todas as requisições HTTP e as encaminha para os controladores adequados para processamento.

### ***Anotações Spring***

As anotações são metadados que simplificam a configuração e o desenvolvimento. As fontes detalham diversas categorias de anotações, como:

- **Configuração:** @Configuration, @Bean, @ComponentScan.
- **Gerenciamento de Componentes:** @Component, @Service, @Repository, @Controller.
- **Injeção de Dependência:** @Autowired, @Qualifier.
- **Anotações Web:** @RequestMapping, @GetMapping, @PostMapping.

### ***Ambiente de Desenvolvimento***

Para desenvolver com Spring Boot, são necessárias várias ferramentas:

- **JDK (Java Development Kit):** Ferramenta básica para programar em Java.
- **IDE (Ambiente de Desenvolvimento Integrado):** Como Eclipse ou IntelliJ IDEA.
- **Spring Initializr:** Ferramenta online para gerar a estrutura inicial de um projeto Spring Boot com as dependências necessárias.
- **Maven:** Ferramenta de automação e gerenciamento de projetos, que utiliza um arquivo POM.xml para gerenciar dependências.
- **Servidor Web:** Como o Apache Tomcat, que pode ser embutido em aplicações Spring Boot.
- **Banco de Dados e Ferramentas:** Como MySQL Workbench para gerenciamento visual e **Spring Data JPA** para simplificar o acesso a dados.

## **Aula 3: Desenvolvimento Web com Spring MVC**

A terceira aula aprofunda os conceitos do desenvolvimento web com Spring MVC, focando nas camadas de visão e controle, e no uso de tecnologias como HTML e Thymeleaf.

## Camada de Visão (View)

- **Responsabilidade:** É a camada responsável por apresentar os dados ao usuário. No Spring MVC, ela é flexível e pode usar tecnologias como JSP e Thymeleaf.
- **ViewResolver:** O Spring MVC utiliza um mecanismo chamado ViewResolver para mapear nomes de visualizações lógicas (definidas no controller) para os arquivos de template reais.
- **Objeto Model:** A camada de visão acessa os dados contidos no objeto Model, que é preenchido pelo controller.

## Thymeleaf e HTML

- **HTML:** É a linguagem de marcação usada para estruturar páginas web. As páginas HTML compõem a camada de visão no modelo MVC.
- **Thymeleaf:** É um *template engine* Java moderno que permite a criação de páginas HTML dinâmicas. Sua principal vantagem é a sintaxe natural, que se assemelha ao HTML puro, facilitando a colaboração entre desenvolvedores e designers. Ele é processado no lado do servidor, o que melhora o desempenho.
- **Integração:** Em uma página HTML, o Thymeleaf é integrado usando um namespace (`xmlns:th="http://www.thymeleaf.org"`) e atributos especiais como `th:object`, `th:action` e `th:field` para vincular dados do Model aos elementos do formulário.

## Front Controller e Camada de Controle

- **Front Controller:** É um padrão de design onde um componente centralizado, o **DispatcherServlet** no Spring MVC, recebe todas as requisições HTTP e as roteia para os controllers apropriados.
- **Camada de Controle (Controller):** É um componente crucial que recebe as requisições do DispatcherServlet, interage com a lógica de negócios (serviços e Model), prepara os dados para a View e seleciona qual View será renderizada para responder ao cliente.
- **Implementação:** Controladores são classes Java anotadas com `@Controller` e seus métodos, anotados com `@GetMapping` ou `@PostMapping`, mapeiam URLs específicas para a lógica de manipulação de requisições. A anotação `@Autowired` é usada para injetar dependências, como classes de serviço.