

Aula 1**CONCEITO DE FRAMEWORK SPRING**

Nesta aula você aprenderá sobre o comportamento, a estrutura e a usabilidade dos frameworks existentes no mercado e conhecerá os principais frameworks do mercado, podendo, assim, desenvolver sua aplicação com o melhor case dessa área.

49 minutos

INTRODUÇÃO

Nos dias atuais, é imprescindível a utilização de ferramentas que favorecem a programação de sistemas cada vez mais complexos. Para facilitar a compreensão e o reuso da codificação realizada por diversos programadores, surgiram estruturas, bibliotecas e ferramentas que estão sendo introduzidas para simplificar o ciclo de desenvolvimento de software.

Com o emprego de frameworks, pode-se criar aplicações complexas com alguns poucos comandos, por isso aprender a utilizá-los é muito importante para desenvolver códigos limpos e consistentes.

Nesta aula você aprenderá sobre o comportamento, a estrutura e a usabilidade dos frameworks existentes no mercado e conhecerá os principais frameworks do mercado, podendo, assim, desenvolver sua aplicação com o melhor case dessa área.

INTRODUÇÃO, PROPÓSITO E CONCEITO**O que é um framework?**

Um framework pode ser considerado uma estrutura ou plataforma para o desenvolvimento de aplicativos ou, ainda, um conjunto de códigos pré-escritos e usados por programadores para desenvolver programas em plataformas específicas. Essas estruturas podem incluir classes e funções predefinidas com o objetivo de que o programador não precise criar o mesmo código cada vez que desenvolve um novo aplicativo (CHRISTENSSON, 2013).

Um framework pode assumir uma forma de plataforma concreta ou conceitual, em que o código comum, com funcionalidade genérica, pode ser seletivamente especializado ou substituído pelos desenvolvedores. As estruturas assumem a forma de bibliotecas, onde uma interface de programa de aplicativo (API) bem definida é reutilizável em qualquer lugar dentro do software em desenvolvimento.

Por que utilizar framework?

O desenvolvimento de software é algo complexo e que deve requerer uma grande quantidade de tarefas, dentre elas a codificação, o design e o teste. Para a parte de códigos, os programadores devem ter cuidado com a sintaxe, as declarações, instruções, exceções e diversos outros parâmetros. Os frameworks para esses softwares facilitam tanto o desenvolvimento que lhes é permitido assumir o controle desse processo, ou seja, grande parte dele a partir de uma única plataforma.

Vamos a algumas vantagens de se usar frameworks:

- Processo de depuração fácil.
- Eficiência de código aprimorada.
- Fácil reutilização de código.
- Desenvolvimento acelerado.
- Segurança melhorada.

Diferença entre frameworks e bibliotecas

A principal diferença entre uma biblioteca e um framework é a "inversão de controle". Ao chamar um método de biblioteca, assume-se que você está no controle. Mas, com um framework, o controle é invertido: ele é quem chama você.

Em um framework, todo o fluxo de controle já está lá e há muitos pontos brancos predefinidos que devem ser preenchidos com código. Um framework normalmente é mais complexo e pode conter várias bibliotecas; ele define um esqueleto, que deverá ser preenchido pelo aplicativo, o qual define seus próprios recursos para isso.

Já uma biblioteca executa operações específicas e bem definidas, uma vez que é apenas uma coleção de definições de classe. O motivo é simplesmente a reutilização do código, ou seja, obtém o código que já foi escrito por outros desenvolvedores.

VIDEOAULA: INTRODUÇÃO, PROPÓSITO E CONCEITO

O vídeo apresentará a linguagem de programação Java juntamente com alguns ambientes para programação da linguagem como o NetBeans e o IntelliJ. Posteriormente, serão apresentados os sites para download de ambas as ferramentas e uma breve introdução ao que se refere o Spring utilizando o próprio site do software para exemplificação.

Videoaula: Introdução, propósito e conceito

Para visualizar o objeto, acesse seu material digital.

FRAMEWORKS E LINGUAGEM DE COMPUTAÇÃO

Linguagens de programação versus frameworks

Antigamente, a programação de computador envolvia o uso da linguagem certa. Entre sistemas como C, Lisp e Pascal, os programadores podiam escolher sua especialidade e formato. No entanto, as diferenças entre as linguagens de programação foram amplamente corrigidas com o aumento do poder de computação, que permite que os sistemas entendam e se movam facilmente entre todas as linguagens. Hoje, o foco está nos frameworks, que tendem a ser mais modernos e com visão de futuro e que podem superar muitas das práticas desatualizadas das linguagens de programação.

Frameworks são conjuntos coesos de código de biblioteca que simplificam a programação em qualquer linguagem (sintaxe e gramática reais para escrever um código). Esses frameworks vêm com uma série de vantagens. Embora as linguagens de programação nunca sejam completamente obsoletas, um número crescente de programadores opta por trabalhar com frameworks e vê-los como a opção mais moderna e de ponta por uma série de razões. A mudança em direção ao framework faz parte da transformação da Tecnologia da Informação (TI), que certamente ganhará força nos próximos anos.

Graças aos sistemas automatizados, ter um conhecimento profundo de várias linguagens de programação simplesmente não é tão importante quanto antes. Os erros podem ser corrigidos automaticamente através dos vários programas que estão constantemente procurando por erros de codificação. Assim, em vez de gastar tempo examinando os detalhes minuciosos de um código, os frameworks permitem que os programadores pensem no cenário maior. Com uma melhor compreensão do que os sistemas e APIs são capazes e com sistemas automatizados para cuidar dos detalhes mais tediosos, os programadores podem se esforçar mais para transformar seus programas em algo maior com mais recursos e potencial de alto nível.

Uma das partes mais importantes do uso de uma linguagem de programação é compreender os algoritmos e certificar-se de que o código se encaixa neles. No entanto, os algoritmos podem ser limitados pela linguagem, porque são realmente definidos pelos frameworks. Alterar e estabelecer algoritmos como parte do framework é muito mais seguro e eficaz do que tentar modificá-los como parte da linguagem.

Frameworks para desenvolvimento web

Frameworks para web são muito úteis e, de muitas maneiras, ajudam os desenvolvedores a construir bons aplicativos, fornecendo diferentes funções e recursos. A seguir estão os aspectos que mostram como eles são benéficos:

- Economia de tempo.
- Aplicativo organizado.
- Flexibilidade.

- Possibilidade de personalização.

A seguir destacamos alguns frameworks para utilização em desenvolvimento web:

Ruby on Rails

Ruby on Rails é um framework para web perfeito para desenvolver um aplicativo de alta velocidade. Descoberto por David Heinemeier Hansson, os aplicativos Ruby on Rails são geralmente dez vezes mais rápidos. É um dos melhores frameworks de back-end, pois comprehende tudo o que é necessário para formar um aplicativo orientado a banco de dados. Os sites que incorporaram Ruby on Rails são: GitHub, Airbnb, GroupOn, Shopify, Hulu, entre outros.

Django

Django é um dos frameworks de desenvolvimento web back-end confiáveis que auxiliam no desenvolvimento de um aplicativo web robusto. É perfeito para competir com prazos rápidos e com os requisitos desafiadores de desenvolvedores veteranos. Os aplicativos Django são extremamente escaláveis, rápidos, versáteis e seguros. As empresas que usam Django são: Pinterest, Disqus, YouTube, Spotify, Instagram e outros gigantes populares.

ASP.NET

Um framework popular de desenvolvimento para web que é imensamente útil para construir aplicativos dinâmicos para PC e dispositivos móveis é o ASP.NET. A Microsoft o inventou para permitir que os programadores criem sites, aplicativos e outros serviços. ASP.NET Core é uma nova versão do ASP.NET e é conhecido por sua velocidade, produtividade e potência; é leve e de alto desempenho. Algumas empresas famosas que o usam são TacoBell, GettyImages, StackOverflow, entre outras.

VIDEOAULA: FRAMEWORKS E LINGUAGEM DE COMPUTAÇÃO

Deverão ser mostrados os sites que utilizam o framework Spring e deverá ser reforçada a importância da utilização dos frameworks para o desenvolvimento de aplicações para web bem como outros frameworks que fazem parte do mundo Java, como o Framework Hibernate e suas funcionalidades.

Vídeoaula: Frameworks e linguagem de computação

Para visualizar o objeto, acesse seu material digital.

FRAMEWORKS PARA DESENVOLVIMENTO WEB EM JAVA

Java é uma linguagem robusta e, quando combinada com frameworks, pode oferecer as melhores soluções para qualquer domínio, seja e-commerce, seja banco, computação em nuvem, finanças, big data, mercado de ações, TI e muito mais.

Os frameworks fornecem uma estrutura para seus aplicativos. Por exemplo, se tivermos uma estrutura adequada para testes, poderemos automatizar muitos processos e obteremos resultados precisos e consistentes. Da mesma forma, se houver estruturas para ORM, aplicativos da web, registro, gerenciamento de dados etc., isso tornará a vida do desenvolvedor simples e o ajudará a se concentrar mais na lógica de negócios do que em partes comuns de código usadas em um domínio ou aplicativo. Destacam-se alguns a seguir:

Spring

Com seu conceito de injeção de dependência e recursos de programação orientados a aspectos, o Spring conquistou o mundo do desenvolvimento. É um framework de código aberto usado para aplicativos corporativos. Com ele, os desenvolvedores podem criar módulos fracamente acoplados, nos quais as dependências são manipuladas pela estrutura em vez de depender das bibliotecas no código.

O framework Spring é amplo e cobre muitos recursos, incluindo segurança e configuração, que são fáceis de aprender. Além disso, é um framework muito popular na web; pode-se encontrar muitas documentações sobre ele e uma comunidade ativa.

Alguns conceitos iniciais devem ser considerados:

- Injeção de dependência (DI) – Inversão de controle (IoC) – Nesse princípio, em vez de o aplicativo assumir o controle do fluxo sequencialmente, ele dá o controle a um controlador externo que conduz o fluxo. Os controladores externos são os eventos. Quando algum evento acontece, o fluxo do aplicativo continua, o que dá flexibilidade ao aplicativo. No Spring, o IoC é feito por DI, que é de três tipos: injeção de setter, injeção de método e injeção de construtor.
- Beans e contexto do Spring – No Spring, os objetos são chamados de *beans* e existe uma *BeanFactory* que gerencia e configura esses *beans*. Pode-se pensar na *BeanFactory* como um contêiner que instancia, configura e gerencia os *beans*. A maioria dos aplicativos usa xml (beans.xml) para a configuração. O *ApplicationContext*, que é um superconjunto de *BeanFactory* é usado para aplicativos mais complexos que precisam de propagação de eventos, mecanismos declarativos e integração com recursos orientados a aspectos do Spring.

Struts

Apache Struts é outra estrutura robusta de código aberto para aplicativos da web. Ele segue o modelo MVC (*Model-View-Controller*) e estende a API JSP. Em uma abordagem servlet-JSP tradicional, se um usuário enviar, digamos, um formulário com seus detalhes, a informação vai para um servlet para processamento ou o controle vai para o próximo JSP (*Java Server Pages* – onde você pode escrever código Java em um HTML). Isso se torna confuso para aplicativos complexos, pois a camada de “Visualização” ou de apresentação, idealmente, não deve ter lógica de negócios.

O Struts separa a visualização, o controlador e o modelo (dados) e fornece a ligação entre cada um por meio de um arquivo de configuração struts-config.xml.

O controlador é um *ActionServlet* por meio do qual você pode escrever modelos para a visualização e no qual os dados do usuário são mantidos usando *ActionForm JavaBean*. O objeto *Action* é responsável por encaminhar o fluxo da aplicação. A visualização é mantida por um rico conjunto de bibliotecas de *tags*.

Struts são fáceis de configurar e fornecem muito mais flexibilidade e extensibilidade em relação à abordagem MVC tradicional usando *servlets* e JSP apenas. Pode ser um bom ponto de partida para carreira como desenvolvedor web.

Hibernate

Embora o Hibernate não seja um framework *full-stack*, ele mudou completamente a maneira como olhamos o banco de dados. Para implementação do Java Persistence API (JPA), o Hibernate serve como um banco de dados *Object-Relational-Mapping* (ORM) para aplicativos Java. Assim como o SQL, as consultas no Hibernate são chamadas de HQL (*Hibernate Query Language*). O Hibernate mapeia diretamente as classes Java para as tabelas de banco de dados correspondentes e vice-versa. O arquivo principal no Hibernate é o hibernate.cfg.xml, que contém informações sobre o mapeamento de classes Java com a configuração do banco de dados.

O Hibernate resolve os principais problemas com JDBC, que não suporta relacionamento em nível de objeto e, se você decidir migrar para um banco de dados diferente, as consultas mais antigas podem não funcionar – o que significa muitas mudanças, isto é, tempo e dinheiro.

O Hibernate fornece uma camada de abstração para que o código seja fricamente acoplado ao banco de dados. Ações como estabelecer uma conexão com o banco de dados e executar operações CRUD (*Create, Read, Update e Delete*) são feitas pelo Hibernate – então os desenvolvedores não precisam implementar isso, tornando o código independente do banco de dados usado.

VIDEOAULA: FRAMEWORKS PARA DESENVOLVIMENTO WEB EM JAVA

Neste bloco será feita uma introdução ao conceito de CRUD (*Create, Read, Update e Delete*), que são as quatro operações básicas para o desenvolvimento de aplicações persistentes. Dentro desse contexto, será mostrado o framework Hibernate, que é uma ferramenta eficaz para montagem desse cenário em Java.

Para visualizar o objeto, acesse seu material digital.

ESTUDO DE CASO

Para contextualizar sua aprendizagem, imagine que você trabalha para uma importante empresa de software na qual é líder do time de desenvolvimento.

A companhia montará um novo sistema web para um cliente externo e você precisa decidir as ferramentas e tecnologias que serão utilizadas para o desenvolvimento. O cliente, porém, exige o uso de tecnologias de ponta, pois há a necessidade de que esse sistema seja robusto e não ocasione um elevado processamento nos servidores.

Além disso, outra exigência da contratante é que o sistema seja feito na linguagem Java, pois existe um desenvolvedor na empresa que acompanhará o processo e que tem afinidade com essa linguagem especificamente.

Nesse sentido, você precisa verificar quais ferramentas e ambientes poderá utilizar, repassando uma lista com as tecnologias das quais disporá.

RESOLUÇÃO DO ESTUDO DE CASO

Em um cenário com tecnologias em constante mudança, as empresas precisam manter o controle sobre as últimas tendências. Além disso, com os aplicativos da web permeando todos os setores, empresas experientes têm procurado maneiras de garantir sua fatia do bolo. E, quando se trata de construir um aplicativo da web – não importa quanto complexo ou grande ele seja – escolher uma plataforma de desenvolvimento é uma decisão com resultados extremos, pois ela constitui o *backbone*, o eixo que mantém seu aplicativo unido.

Pode ser muito desafiador escolher a plataforma de desenvolvimento web certa para seu próximo projeto de desenvolvimento web, já que uma vasta gama de estruturas ou linguagens de programação são apresentadas ao mundo com uma arquitetura de estrutura variada.

Frameworks e linguagens de programação que variam de código aberto a tecnologias pagas são opções muito poderosas e, às vezes, um pouco confusas também.

Por esse motivo, é necessário pensar em aplicativos rápidos e escaláveis para utilização. Nesse caso, seria interessante utilizar a linguagem Java para a construção do aplicativo, o framework Hibernate para a criação das conexões com o banco, o Spring como framework para tratar das páginas web e das regras de negócios, e o Struts ou o JSF para *front-end*.

Resolução do Estudo de Caso

Para visualizar o objeto, acesse seu material digital.

6º Saiba mais

Para conhecer melhor os conceitos que envolvem frameworks, vale a pena conferir o vídeo a seguir:

https://www.youtube.com/watch?v=2zqzzTnfa0E&ab_channel=C%C3%B3digofonteTV

E para conhecer mais sobre framework Spring e sua estrutura, acesse:

<https://spring.io/>

Aula 2

FRAMEWORK SPRING: FRONT-END

Se você acabou de começar a aprender desenvolvimento web, provavelmente já ouviu falar muito sobre programação de front-end e back-end. Mas o que exatamente queremos dizer com isso?

48 minutos

INTRODUÇÃO

Se você acabou de começar a aprender desenvolvimento web, provavelmente já ouviu falar muito sobre programação de *front-end* e *back-end*. Mas o que exatamente queremos dizer com isso? Se você for um iniciante no campo, pode ser difícil saber não apenas qual é qual, mas também o que é abordado por um ou outro. Essencialmente, a diferença entre o desenvolvimento web de *front-end* e o de *back-end* é que o primeiro atende o lado do cliente (o que vemos na frente, isto é, uma tela) e o último dá suporte ao lado do servidor (o que está sob o “capô” de um site ou sistema).

Embora esses dois tipos de programação sejam certamente distintos um do outro, eles também são como os dois lados da mesma moeda. A funcionalidade de um site depende de cada lado se comunicar e operar efetivamente com o outro como uma única unidade. E um é mais importante do que o outro? Não, pois ambos desempenham papéis muito importantes no desenvolvimento web. Então, por onde devemos começar? Vamos, então, obter respostas para suas perguntas.

FRONT-END E BACK-END

A diferença entre desenvolvimento de *front-end* e *back-end*

O *front-end* de um site é o que você vê e com o qual interage no seu navegador. Também conhecido como “lado do cliente”, inclui tudo o que o usuário experimenta diretamente: de texto e cores a botões, imagens e menus de navegação. Digamos que você decidiu abrir um negócio, uma padaria gourmet para cães, por exemplo, e precisa de um site profissional para apresentar sua empresa aos clientes e dizer onde você está. Nele talvez você inclua algumas fotos e algumas informações sobre seus produtos e, para isso, tudo de que você precisa são tecnologias *front-end* para construir seu site.

Responsável por armazenar e organizar dados e garantir que tudo no lado do cliente realmente funcione, o *back-end* se comunica com o *front-end* enviando e recebendo informações para serem exibidas como uma página da web. Sempre que você preenche um formulário de contato, digita um endereço da web ou faz uma compra (qualquer interação do usuário com o lado do cliente), seu navegador envia uma solicitação para o lado do servidor, que retorna informações na forma de código de *front-end*, que o navegador pode interpretar e exibir.

Se você está interessado em aprender desenvolvimento web, mas não tem certeza se deve seguir a rota do *front-end* ou do *back-end*, é importante considerar as tarefas do dia a dia de cada um. Se você gosta da ideia de trabalhar com designs visuais e trazê-los à vida, criando uma experiência de usuário de primeira classe, provavelmente gostará de trabalhar no *front-end*. Agora, se você gosta de trabalhar com dados, descobrir algoritmos e maneiras de otimizar sistemas complexos, talvez prefira trabalhar como desenvolvedor de *back-end*. No entanto, saiba que a distinção entre eles nem sempre é tão clara. Alguns desenvolvedores são proficientes tanto em *front-end* quanto em *back-end* e são conhecidos como desenvolvedores *full-stack*.

Quais são as principais linguagens de desenvolvimento de *front-end*?

Estas três linguagens resolverão o problema:

- HTML: é a linguagem de codificação fundamental que cria e organiza o conteúdo da web para que ele possa ser exibido por um navegador.
- CSS: é uma linguagem que acompanha o HTML e define o estilo do conteúdo de um site, como layout, cores, fontes, etc.

- JavaScript: é uma linguagem de programação usada para elementos mais interativos, como menus suspensos, janelas modais e formulários de contato.

Juntos, esses fundamentos criam tudo o que é apresentado visualmente quando você visita uma página da web – seja para fazer compras on-line, seja para ler notícias, verificar e-mail ou fazer uma pesquisa no Google.

Quais são as principais linguagens de desenvolvimento de *back-end*?

Enquanto o desenvolvimento *front-end* trata de fazer sites e aplicativos da web renderizarem no lado do cliente, o de *back-end* trata de fazê-los renderizarem no lado do servidor. Mas ainda assim é um pouco mais do que isso. Embora a afirmação anterior seja verdadeira, os desenvolvedores de *back-end* também criam serviços que processam a lógica de negócios e acessam outros recursos, como bancos de dados, servidores de arquivos, serviços em nuvem e muito mais. Esses serviços são a espinha dorsal de qualquer aplicativo e podem ser acessados e usados não apenas por aplicativos de renderização do lado do servidor, mas também por aplicativos de renderização do lado do cliente.

Vejamos algumas linguagens para desenvolvimento *back-end*:

- PHP: é uma linguagem de script do lado do servidor projetada especificamente para desenvolvimento web, por isso é chamada de linguagem de script do lado do servidor.
- C++: é uma linguagem de programação de propósito geral e amplamente usada atualmente para programação competitiva. Ele também é usada como uma linguagem de *back-end*.
- Java: é uma das plataformas e linguagens de programação mais populares e amplamente utilizadas. É altamente escalonável e seus componentes estão facilmente disponíveis.
- Python: é uma linguagem de programação que permite trabalhar rapidamente e integrar sistemas com mais eficiência.
- JavaScript: pode ser usado como linguagem de programação (*front-end* e *back-end*).

VIDEOAULA: FRONT-END E BACK-END

O vídeo será feito para mostrar trechos de códigos de *front-end*, como HTML, CSS e JavaScript, e para ensinar como baixá-los e utilizá-los. Ainda, serão mostradas as linguagens de programação de *back-end* e quais programas e ferramentas podem ser utilizados com elas.

Videoaula: *Front-end e back-end*

Para visualizar o objeto, acesse seu material digital.

ARQUITETURA MVC

A arquitetura MVC

A estrutura *Model-View-Controller* (MVC) é um padrão arquitetônico que separa um aplicativo em três componentes lógicos principais: *Model*, *View* e *Controller*, daí a abreviatura MVC. Cada componente de arquitetura é construído para lidar com aspectos específicos de desenvolvimento de um aplicativo. Nesse sentido, o MVC separa a lógica de negócios e a camada de apresentação uma da outra.

Características do MVC

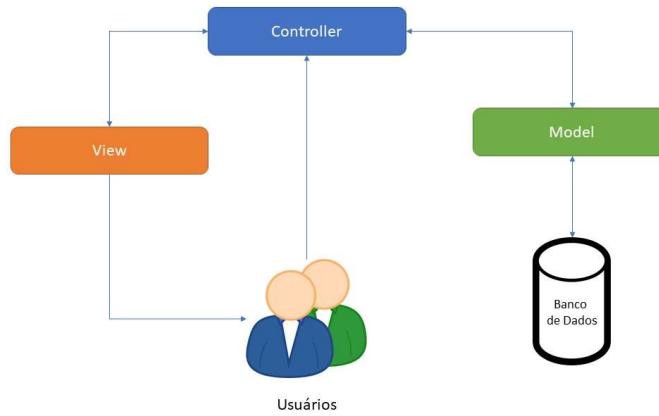
Hoje em dia, a arquitetura MVC em tecnologia da web se tornou popular para projetar tanto aplicativos da web quanto aplicativos móveis. Existem algumas características importantes a serem destacadas sobre essa arquitetura:

- Testabilidade fácil e sem atrito, isto é, estrutura altamente testável.
- Oferecimento de controle total sobre seu HTML bem como sobre seus URLs para projetar uma arquitetura de aplicativo da web usando o padrão MVC.
- Possibilidade de aproveitar os recursos existentes fornecidos pelo ASP.NET, JSP, Django, etc.
- Separação clara de lógica: modelo, visão, controlador.
- Separação de tarefas de aplicativo entre as visualizações de lógica de negócios, lógica UI e lógica de entrada.

- Roteamento de URL para URLs amigáveis para SEO. Mapeamento de URL poderoso para URLs comprehensíveis e pesquisáveis.
 - Suporte para *Test Driven Development* (TDD).
- MVC *Pattern* significa *Model-View-Controller Pattern*, padrão usado para separar os interesses do aplicativo.
- **Modelo:** representa um objeto ou JAVA POJO que carrega dados. Ele também pode ter lógica para atualizar o controlador se seus dados mudarem.
 - **View:** representa a visualização dos dados que o modelo contém.
 - **Controlador:** o controlador atua no modelo e na visualização. Ele controla o fluxo de dados no objeto do modelo e atualiza a visualização sempre que os dados são alterados. Ele mantém a visualização e o modelo separados.

Veja o exemplo a seguir:

Figura 1 | Modelo MVC



Fonte: elaborada pelo autor.

No contexto da programação Java, o model são as classes Java simples, a view exibe os dados e o controller consiste em *servlets*. Essa separação resulta em solicitações do usuário que são processadas da seguinte maneira:

1. O navegador no cliente envia uma solicitação de página ao controlador presente no servidor.
2. O controlador executa a ação de invocar o modelo, recuperando assim os dados de que precisa em resposta à solicitação.
3. O controlador então fornece os dados recuperados para a visualização.
4. A visualização é renderizada e enviada de volta ao cliente para o navegador exibir.

Implementação de MVC usando Java

Para implementar uma aplicação web baseada no padrão de design MVC, criaremos:

- *Classe CursoModel*, que atua como a camada do modelo.
- *Classe CursoView*, que define a camada de apresentação (camada de visualização).
- *Classe CursoController*, que atua como um controlador.

A camada modelo

No padrão de design MVC, o modelo é a camada de dados que define a lógica de negócios do sistema e também representa o estado do aplicativo. O modelo são simplesmente os dados para nosso aplicativo. Os dados são “modelados” de uma forma que seja fácil de armazenar, recuperar e editar. O modelo é como aplicamos regras aos nossos dados, que eventualmente representam os conceitos gerenciados por nosso aplicativo. Vejamos a construção da classe *CursoModel*:

Figura 2 | Classe CursoModel

```

1 package projectMVC;
2
3 public class CursoModel {
4     private String CourseName;
5     private String CourseId;
6     private String CourseCategory;
7
8     public String getId() {
9         return CourseId;
10    }
11
12    public void setId(String id) {
13        this.CourseId = id;
14    }
15
16    public String getName() {
17        return CourseName;
18    }
19
20    public void setName(String name) {
21        this.CourseName = name;
22    }
23
24    public String getCategory() {
25        return CourseCategory;
26    }
27
28    public void setCategory(String category) {
29        this.CourseCategory = category;
30    }
31
32 }
33

```

Fonte: elaborada pelo autor.

A camada de visualização

Essa camada do padrão de design MVC representa a saída do aplicativo ou da interface do usuário. Ele exibe os dados buscados da camada do modelo pelo controlador e apresenta os dados ao usuário sempre que solicitado. Como o nome sugere, a visualização é responsável por renderizar os dados recebidos do modelo. Pode haver modelos predefinidos, nos quais você pode ajustar os dados, e pode até haver várias visualizações diferentes por modelo dependendo dos requisitos. Vamos criar uma visualização usando a classe *CursoView*.

Figura 3 | Classe *CursoView*

```

1 package projectMVC;
2
3 public class CursoView {
4     public void printCourseDetails(String CourseName, String CourseId, String CourseCategory) {
5         System.out.println("Course Details: ");
6         System.out.println("Name: " + CourseName);
7         System.out.println("Course ID: " + CourseId);
8         System.out.println("Course Category: " + CourseCategory);
9     }
10 }

```

Fonte: elaborada pelo autor.

Esse código serve simplesmente para imprimir os valores no console. Em seguida, temos o controlador do aplicativo da web.

A camada do controlador

O *Controller* é como uma interface entre *Model* e *View* que recebe as solicitações do usuário da camada de visualização e as processa, incluindo as validações necessárias. As solicitações são então enviadas ao modelo para processamento de dados, os quais, depois de processados, são enviados novamente para o controlador e, em seguida, exibidos na visualização.

A função principal de um controlador é chamar o modelo e coordenar com ele a busca de todos os recursos necessários para agir. Normalmente, ao receber uma solicitação do usuário, o controlador chama o modelo apropriado para a tarefa em questão.

Vamos, agora, criar a classe *CursoController*, que atua como um controlador.

Figura 4 | Classe *CursoController*

```

1 package projectMVC;
2
3 public class CursoController {
4     private CursoModel model;
5     private CursoView view;
6
7     public CursoController(CursoModel model, CursoView view) {
8         this.model = model;
9         this.view = view;
10    }
11
12    public void setCourseName(String name) {
13        model.setName(name);
14    }
15
16    public String getCourseName() {
17        return model.getName();
18    }
19
20    public void setCourseId(String id) {
21        model.setId(id);
22    }
23
24    public String getCourseId() {
25        return model.getId();
26    }
27
28    public void setCourseCategory(String category) {
29        model.setCategory(category);
30    }
31
32    public String getCourseCategory() {
33        return model.getCategory();
34    }
35
36    public void updateView() {
37        view.printCourseDetails(model.getName(), model.getId(), model.getCategory());
38    }
39 }

```

Fonte: elaborada pelo autor.

Uma rápida olhada no código nos dirá que essa classe de controlador é apenas responsável por chamar o modelo para obter/definir os dados e atualizar a visualização com base nisso. Agora, vamos dar uma olhada em como tudo isso está interligado.

Classe principal

Chamaremos essa classe de *MVCPatternDemo.java*. Confira o código adiante.

Figura 5 | Classe principal

```

1 package projectMVC;
2
3 public class MVCTest {
4     public static void main(String[] args) {
5
6         // fetch student record based on his roll no from the database
7         CursoModel model = retrieveCourseFromDatabase();
8
9         // Create a view : to write course details on console
10        CursoView view = new CursoView();
11
12        CursoController controller = new CursoController(model, view);
13
14        controller.updateView();
15
16        // update model data
17        controller.setCourseName("Python");
18        System.out.println("\nAfter updating, Course Details are as follows");
19
20        controller.updateView();
21    }
22
23    private static CursoModel retrieveCourseFromDatabase() {
24        CursoModel course = new CursoModel();
25        course.setName("Java");
26        course.setId("01");
27        course.setCategory("Programming");
28
29    }
30 }

```

Fonte: elaborada pelo autor.

A classe mostrada anteriormente busca os dados do curso da função usando o conjunto de valores inserido pelo usuário. Em seguida, ele envia esses valores para o modelo do curso e depois inicializa a visualização que criamos anteriormente. Além disso, ele também invoca a classe *CursoController* e a vincula às classes *Course* e *CursoView*. O método *updateView()*, que faz parte do controlador, atualiza os detalhes do curso no console. Confira a saída a seguir.

Figura 6 | Saída da classe principal

```
Course Details:  
Name: Java  
Course ID: 01  
Course Category: Programming  
  
After updating, Course Details are as follows  
Course Details:  
Name: Python  
Course ID: 01  
Course Category: Programming
```

Fonte: elaborada pelo autor.

A arquitetura MVC fornece um nível totalmente novo de modularidade ao seu código, o que o torna muito mais legível e sustentável.

VIDEOAULA: ARQUITETURA MVC

O vídeo a ser gravado deverá demonstrar, na prática, a implementação dos exemplos de códigos criados durante a explanação do bloco, o que consistirá em um exemplo prático da criação de um pequeno projeto em código Java com algumas classes, mostrando também importância do padrão MVC e a maneira de aplicá-lo a projetos que utilizam a linguagem de programação Java.

Videoaula: Arquitetura MVC

Para visualizar o objeto, acesse seu material digital.

CONFIGURANDO AMBIENTE DE DESENVOLVIMENTO PARA SPRING

Configurando ambiente de desenvolvimento para Spring

Como algumas ferramentas são essenciais para se trabalhar com Java e com o framework Spring, abordaremos algumas delas a seguir com o propósito de deixar o ambiente pronto para o trabalho.

Para isso devemos, inicialmente, configurar algumas ferramentas básicas: devemos baixar e configurar a IDE Eclipse, disponível no link: <https://www.eclipse.org/downloads/>.

Junto com a instalação da IDE Eclipse vem o kit de instalação do Java, o qual instala e configura as variáveis de ambientes necessárias para o correto funcionamento das ferramentas.

Em um segundo momento, devemos instalar e configurar uma ferramenta de gerenciamento de dependências; para esse caso de estudo, utilizaremos a ferramenta Maven.

O que é o Maven?

Apache Maven é uma ferramenta de gerenciamento e compreensão de projetos de software, que, com base no conceito de modelo de objeto de projeto (POM), pode gerenciar a construção, o relatório e a documentação de um projeto a partir de uma informação central.

O Maven tem como objetivo permitir que um desenvolvedor estude e comprehenda o estado completo do esforço de desenvolvimento no menor período de tempo. Para isso, o Maven tem alguns claros objetivos, dentre eles:

- Facilitar o processo de construção de uma aplicação.
- Efetuar a construção de um sistema uniforme com documentação.
- Fornecer informações do projeto e utilizar as melhores práticas de desenvolvimento.

Download do Apache Maven

O download do Apache Maven pode ser feito no [site oficial](#), no qual, após acessado, é possível baixar o arquivo zipado, conforme mostra a imagem:

Figura 7 | Arquivo Maven para download

Ligação	Checksums	Assinatura
Arquivo binário tar.gz	apache-maven-3.8.2-bin.tar.gz	apache-maven-3.8.2-bin.tar.gz.asc
Arquivo zip binário	apache-maven-3.8.2-bin.zip	apache-maven-3.8.2-bin.zip.asc
Arquivo fonte tar.gz	apache-maven-3.8.2-src.tar.gz	apache-maven-3.8.2-src.tar.gz.asc
Arquivo zip de origem	apache-maven-3.8.2-src.zip	apache-maven-3.8.2-src.zip.asc

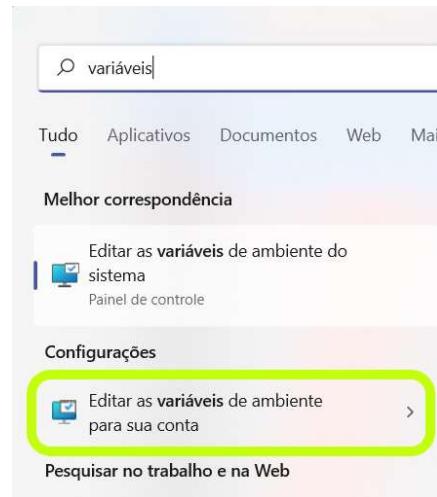
Fonte: captura de tela adaptada do site oficial do Apache Maven.

Após o download do arquivo, deve-se descompactá-lo na raiz do sistema, não sendo necessária a instalação de nenhum componente, apenas o desempacotamento do arquivo, por exemplo: "C:\Apache Maven".

Configurando Maven nas variáveis de ambiente do Windows

Para configurar o Maven nas variáveis de ambiente do Windows, é necessário abrir a janela de configurações, como mostrado na figura:

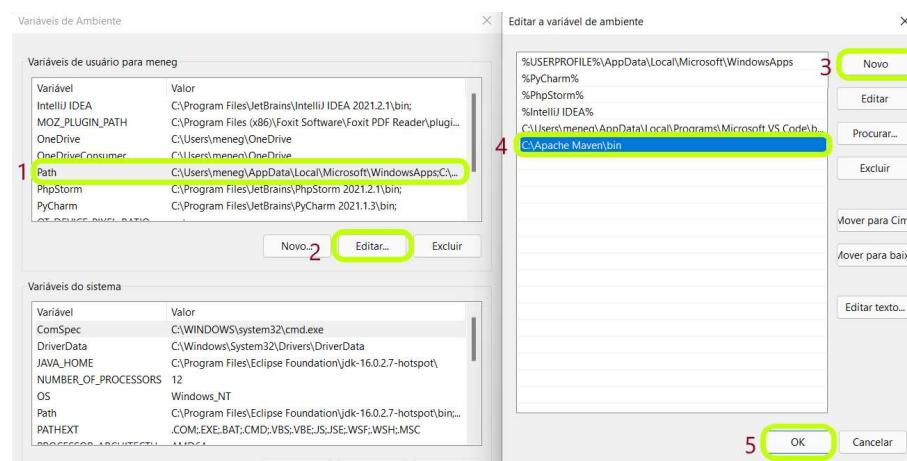
Figura 8 | Busca Windows para variáveis de ambiente



Fonte: captura de tela adaptada do Windows.

Depois de abrir a tela de configuração das variáveis do usuário, deve-se encontrar a variável "Path", clicar em "Editar" e adicionar o caminho da pasta "bin" do local onde foi descompactado, conforme exemplo a seguir:

Figura 9 | Passo a passo para configuração das variáveis de ambiente



Fonte: captura de tela adaptada do Windows.

Concluído o passo a passo, devemos verificar se o Maven foi instalado, abrindo o prompt de comando e digitando o seguinte comando "mvn -v". Com isso, deve aparecer a seguinte mensagem:

Figura 10 | Comando de verificação de versão do Maven

```
c:\>mvn -v
Apache Maven 3.8.2 (ea98e05a04480131370aa0c110b8c54cf726c06f)
Maven home: C:\Apache Maven
Java version: 16.0.2, vendor: Eclipse Foundation, runtime: C:\Program Files\Eclipse Foundation\jdk-16.0.2.7-hotspot
Default locale: pt_BR, platform encoding: Cp1252
OS name: "windows 10", version: "10.0", arch: "amd64", family: "windows"
c:\>
```

Fonte: captura de tela do prompt de comando.

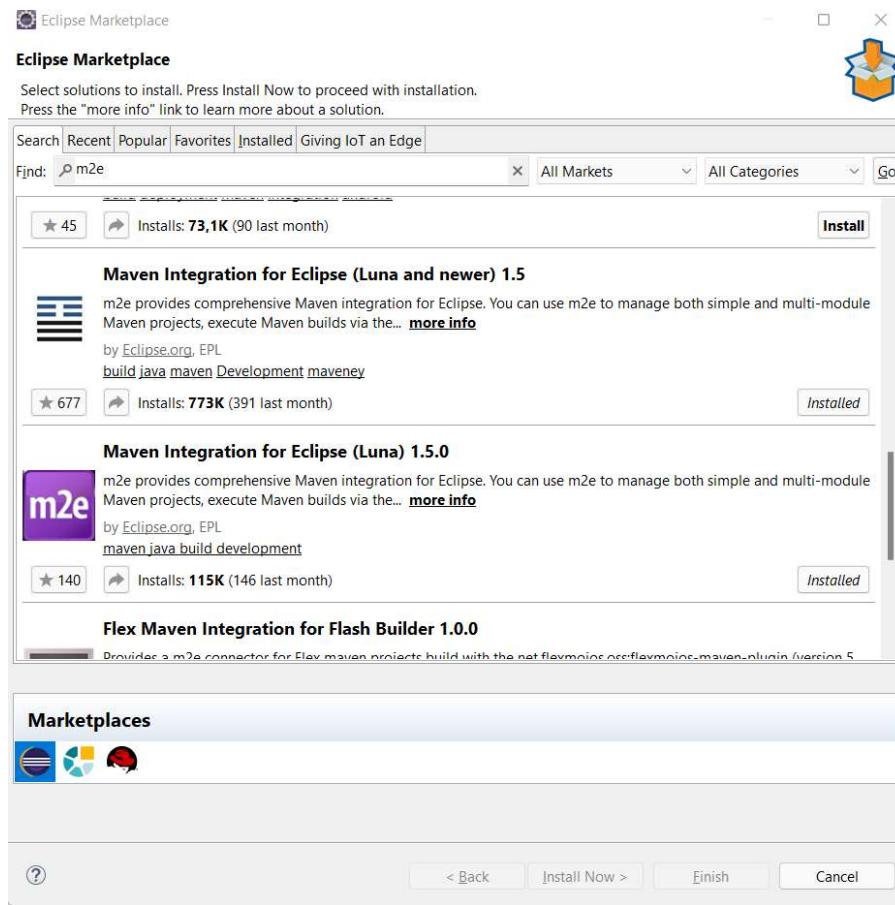
A mensagem mostra que a versão do Maven foi concluída com êxito; com isso, o Java e o Eclipse estão prontos para serem utilizados.

Repositório Maven e configuração do plugin M2E

O repositório Maven é um diretório no qual estão os arquivos a serem baixados e indexados no projeto, basta adicionar as dependências no arquivo POM, que é um acrônimo para *Project Object Model*. Nele ficam as informações do projeto bem como as configurações que serão utilizadas. O Maven lê o arquivo pom.xml e executa as meta *tags* baixando os arquivos necessários para o projeto.

Para usar o Maven junto com o Eclipse, é necessária a utilização de um plugin que pode ser baixado no *marketplace* da IDE. Para isso, siga o caminho "help >> Eclipse Marketplace" e uma nova janela abrirá; após isso basta pesquisar por "M2E" e instalá-lo, conforme mostrado a seguir:

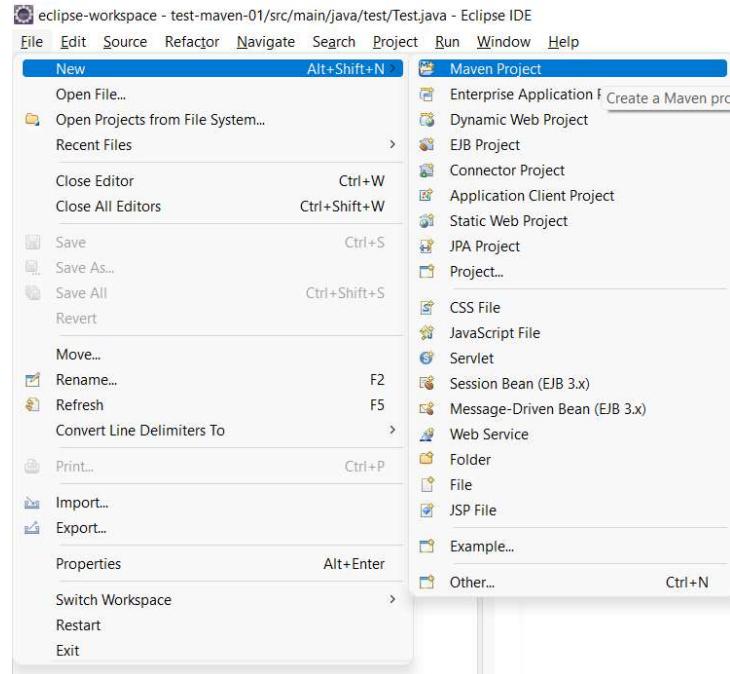
Figura 11 | Instalando Plugin M2E



Fonte: captura de tela adaptada do Apache Maven.

Depois de instalado o plugin, vamos criar um projeto teste do Maven seguindo os passos adiante:

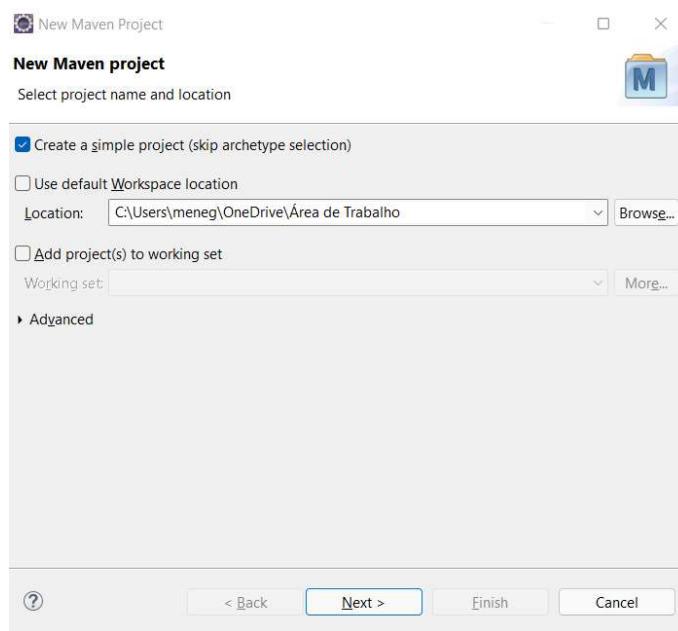
Figura 12 | Criando projeto Maven



Fonte: captura de tela do Apache Maven.

Na próxima janela, devemos marcar a opção “create a simple project” e desmarcar a opção “Use default Workspace” e clicar no botão “Next”.

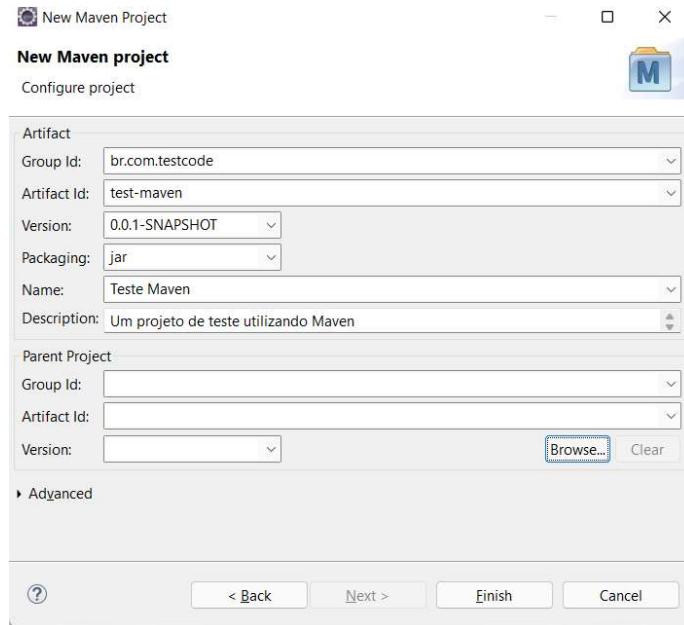
Figura 13 | Marcar opção de projeto simples



Fonte: captura de tela adaptada do Apache Maven.

Na próxima tela, devemos preencher os campos de acordo com o projeto que estamos criando, como é feito na imagem a seguir:

Figura 14 | Preencher campos



Fonte: captura de tela do Apache Maven.

Logo, após finalizar as configurações, será criada a estrutura pela ferramenta, a qual ficará com o seguinte formato:

Figura 15 | Estrutura do projeto



Fonte: captura de tela do Apache Maven.

Dentro do arquivo pom.xml, colocaremos todas as dependências que o projeto exige. Assim, temos uma ferramenta importante implantada em nossa estrutura e que facilitará a busca por frameworks e outros componentes.

VIDEOAULA: CONFIGURANDO AMBIENTE DE DESENVOLVIMENTO PARA SPRING

No vídeo em questão, será mostrado como fazer o download da IDE Eclipse, do Java e da ferramenta de automação Apache Maven. Será também demonstrado o passo a passo de cada instalação e a configuração do ambiente, bem como um exemplo da utilização do Maven.

Videoaula: Configurando ambiente de desenvolvimento para Spring

Para visualizar o objeto, acesse seu material digital.

ESTUDO DE CASO

Imagine que a empresa em que você trabalha está tendo muitos problemas no setor de desenvolvimento devido a alguns programadores novos não conseguirem achar as bibliotecas existentes no sistema (o qual está passando por uma manutenção), pois os programadores mais antigos já não trabalham mais ali.

Os desenvolvedores estão levando horas para procurar, na internet e nos repositórios, algumas bibliotecas que já foram substituídas ou cujos métodos foram depreciados e substituídos por novos, levando em consideração suas novas versões.

Ainda, um outro problema os assola: a quantidade de espaço em disco que se encontra em utilização está sobrecarregando o projeto, fazendo com que muitas bibliotecas que não são utilizadas fiquem perdidas no projeto.

Dante desse cenário, você foi escalado para encontrar uma forma de melhorar esse processo, trazendo algumas soluções para organizar todo esse cenário caótico entre os desenvolvedores.

RESOLUÇÃO DO ESTUDO DE CASO

Com o intuito de resolver tal problema, é necessário pensar sobre as ferramentas disponíveis no mercado e quais delas realmente podem fazer a diferença na hora de organizar um projeto que talvez necessite ser revisado constantemente. Por isso, é importante analisar o cenário como um todo e pensar em ferramentas que garantam um gerenciamento de projeto que seja ágil e que supra as necessidades dos programadores.

Sem uma ferramenta de construção, gerenciar e construir aplicativos Java seria um processo muito doloroso, longo e frequentemente repetitivo. Com o Maven, é fácil manter as bibliotecas do projeto usando o sistema de dependências e construir o projeto usando frameworks e bibliotecas.

Maven é, sem dúvida, uma das ferramentas de construção mais populares em Java. Junto com o Gradle – uma ferramenta de construção diferente, mas com funcionalidade semelhante –, eles são usados em mais de 90% dos projetos Java. O restante dos projetos está usando Ant ou outras ferramentas de construção proprietárias.

A solução ideal nesse cenário seria a utilização de ferramentas como Ant e Maven, que são gerenciadores de dependências que supririam a necessidade dos programadores mais novos. Nesse sentido, bastaria a utilização de um desses frameworks.

Resolução do Estudo de Caso

Para visualizar o objeto, acesse seu material digital.

6º Saiba mais

Para desenvolver melhor os conceitos e objetivos das ferramentas trabalhadas, visite os sites delas, pois neles também é possível fazer o download para utilização:

<https://www.apache.org/>

<https://maven.apache.org/>

<https://www.eclipse.org/>

Aula 3

FRAMEWORK SPRING: MID-TIER

Frequentemente, você descobrirá que combinar alguma programação declarativa e imperativa, que é suportada por algumas linguagens, é a solução ideal. Nesse sentido, o mais importante é encontrar um programador habilidoso e experiente, que saiba quando usar os diferentes paradigmas para maximizar os benefícios de cada um.

44 minutos

INTRODUÇÃO

Praticamente todo o mundo digital, que é tão central em nossas vidas hoje, é construído em programas de computador. Essencialmente, é assim que instruímos nossos dispositivos a executar certas funções e a resolver problemas. Enquanto as máquinas trabalham com código binário – apenas 0s e 1s – nós, humanos, desenvolvemos centenas de linguagens de programação que nos permitem comunicar-nos com nossos computadores.

Não está claro exatamente quantas linguagens de programação existem hoje, mas é possível afirmar que mais estão sendo criadas o tempo todo. Portanto, é útil organizar essas linguagens em categorias que nos ajudem a entender as semelhanças e diferenças entre elas. Embora não haja uma maneira definitiva de classificar ou agrupar tipos de linguagens de programação, uma abordagem comum é dividir as em paradigmas centrais, os quais determinam como seu código é estruturado e organizado.

É bastante comum, no desenvolvimento de software, não existir um paradigma de programação objetivamente "certo" ou "melhor", pois tudo depende do que você precisa fazer, dos recursos disponíveis e de quem mais está trabalhando no projeto. Frequentemente, você descobrirá que combinar alguma programação declarativa e imperativa, que é suportada por algumas linguagens, é a solução ideal. Nesse sentido, o mais importante é encontrar um programador habilidoso e experiente, que saiba quando usar os diferentes paradigmas para maximizar os benefícios de cada um. Por este motivo, o estudo desses paradigmas se faz tão necessário.

CONCEITOS DE AOP: FUNDAMENTOS DE ASPECT ORIENTED PROGRAMMING

Conceitos de AOP: fundamentos de *Aspect Oriented Programming*

A Programação Orientada a Aspectos (AOP) complementa a Programação Orientada a Objetos (POO), fornecendo outra maneira de pensar sobre a estrutura do programa. A unidade chave de modularidade em POO é a classe, enquanto em AOP é o aspecto. Aspectos permitem a modularização de questões, como gerenciamento de transações que abrangem vários tipos e objetos.

Nesse cenário, a vantagem do Spring AOP, por exemplo, é que ele fornece serviços corporativos declarativos, especialmente como um substituto para os serviços declarativos EJB (Enterprise Java Beans), o qual possui a lógica que atua sobre os dados do negócio. O mais importante desses serviços é o gerenciamento de transações declarativas, uma vez que permite aos usuários implementarem aspectos personalizados, complementando o uso de POO com AOP.

Vejamos, a seguir, os conceitos e as terminologias do AOP:

- **Aspect:** modularização de uma preocupação que permeia várias classes. O gerenciamento de transações é um bom exemplo de preocupação transversal em aplicativos Java corporativos. No Spring AOP, os aspectos são implementados através de classes regulares (abordagem baseada em esquema) ou de classes regulares com a anotação @Aspect.
- **Join point:** um ponto durante a execução de um programa, como a execução de um método ou o tratamento de uma exceção. No Spring AOP, um ponto de junção sempre representa a execução de um método.
- **Advice:** ação realizada por um aspecto em um ponto de junção específico. Diferentes tipos de *advices* incluem anotações "ao redor", "antes" e "depois". Muitos frameworks AOP, incluindo Spring, modelam um conselho como um interceptor, mantendo uma cadeia de interceptores em torno do ponto de junção.
- **Pointcut:** um predicado que combina os pontos de junção. O conselho está associado a uma expressão de *pointcut* e é executado em qualquer ponto de junção correspondido por ele (por exemplo, a execução de um método com determinado nome). O conceito de pontos de junção, conforme correspondências com expressões de *pointcut*, é central para AOP, e Spring usa a linguagem de expressão de *pointcut Aspect* por padrão.
- **Introduction:** declara métodos ou campos adicionais em nome de um tipo. Spring AOP permite que você introduza novas interfaces (e uma implementação correspondente) para qualquer objeto recomendado. Por exemplo, você pode usar uma introdução para fazer um *bean* implementar uma interface *IsModified* a fim de simplificar o armazenamento em cache.
- **Target object:** objeto que é aconselhado por um ou mais aspectos (também conhecido como objeto aconselhado). Uma vez que Spring AOP é implementado com proxies em tempo de execução, esse objeto sempre será um objeto com proxy.
- **AOP proxy:** um objeto criado pelo framework AOP para implementar os contratos de aspecto (aconselhar execuções de métodos e assim por diante). No Spring Framework, um proxy AOP será um proxy dinâmico JDK ou um proxy CGLIB.
- **Weaving:** liga aspectos com outros tipos de aplicativos ou objetos para criar um objeto recomendado. Isso pode ser feito em tempo de compilação (usando o compilador AspectJ, por exemplo), em tempo de carregamento ou em tempo de execução. O Spring AOP, como outros frameworks Java AOP puros, realiza entrelaçamento no tempo de execução.

Os conceitos e as terminologias de AOP refletem diretamente na composição do código e na lógica que será aplicada nos sistemas. Com o conhecimento adquirido sobre os principais conceitos, você pode trabalhar tranquilamente com qualquer framework que tenha AOP na estrutura.

VIDEOAULA: CONCEITOS DE AOP: FUNDAMENTOS DE ASPECT ORIENTED PROGRAMMING

O vídeo deverá mostrar os componentes AOP presentes na linguagem de programação Java e o foco no paradigma de programação que AOP envolve, juntamente com comparações com o paradigma de Programação Orientada a Objetos.

Videoaula: Conceitos de AOP: fundamentos de Aspect Oriented Programming

Para visualizar o objeto, acesse seu material digital.

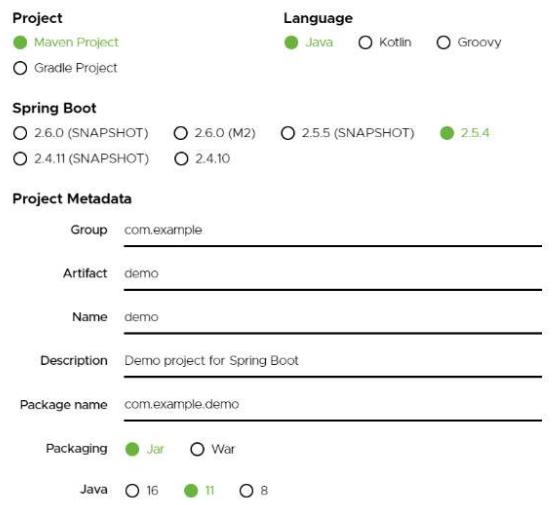
IMPLEMENTANDO SPRING

Implementando Spring

Para implementação do Spring, utilizaremos o padrão MVC e então construiremos o clássico “Hello World”, ponto de partida de qualquer projeto, com o intuito de conhecer a estrutura do framework. Para isso, utilizaremos a própria página do Spring: <https://start.spring.io/>, onde criaremos o projeto inicial e, a partir daí, importaremos em nosso IDE (Integrated Development Environment).

Ao entrarmos no site de ponto de partida do Spring, vemos algumas opções, as quais devemos alterar de acordo com nosso critério de utilização. Para este exemplo simples, apenas deixaremos as pré-configurações do modo como as encontramos: a fim de criarmos o projeto via Maven (gerenciador de dependências), marcaremos a opção Java como linguagem, a versão do Spring e por último inseriremos os dados no campo *Project Metadata*, nos campos de *Packaging* e Java, e selecionaremos a opção que está configurada em nossa máquina. O projeto ficará desta forma:

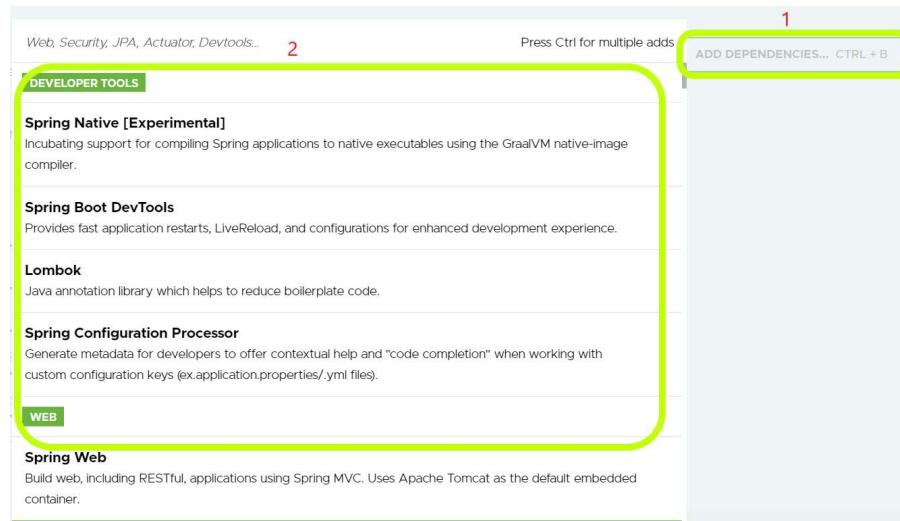
Figura 1 | Estrutura do projeto Spring



Fonte: adaptada de Spring (c2013-2021).

Neste mesmo contexto, podemos adicionar as dependências ao projeto que estamos criando, como dependências de ferramentas do Spring, web, templates, segurança, banco de dados e diversos outros

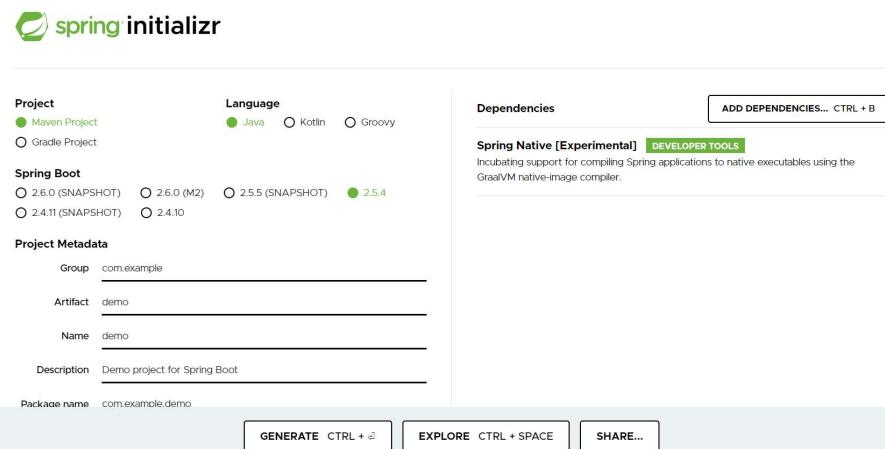
Figura 2 | Dependências Spring



Fonte: adaptada de Spring (c2013-2021).

No primeiro passo, deve-se clicar no botão “adicionar dependências” e depois podemos selecionar as ferramentas desejadas. Após concluída essa etapa, a tela deverá ser mostrada da seguinte forma:

Figura 3 | Organização de parâmetros do projeto

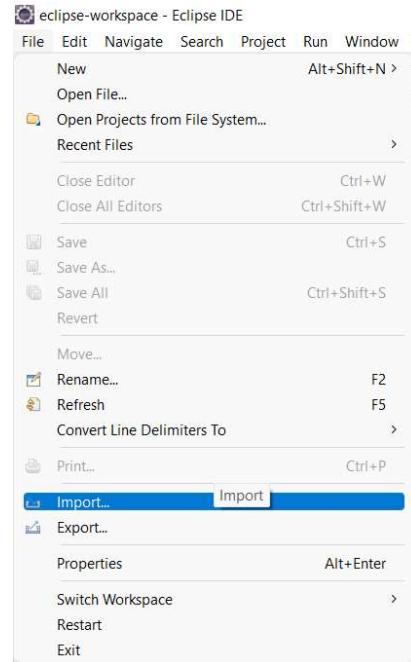


Fonte: adaptada de Spring (c2013-2021).

Nos botões “Generate”, “Explore” e “Share”, temos as opções de gerar o projeto para exportá-lo em nosso IDE ou, então, apenas explorar a estrutura do projeto, ação que permitirá copiar ou fazer alterações para utilizar o código em outras ferramentas; já a opção “Share” permite que compartilhemos o projeto externamente ao site.

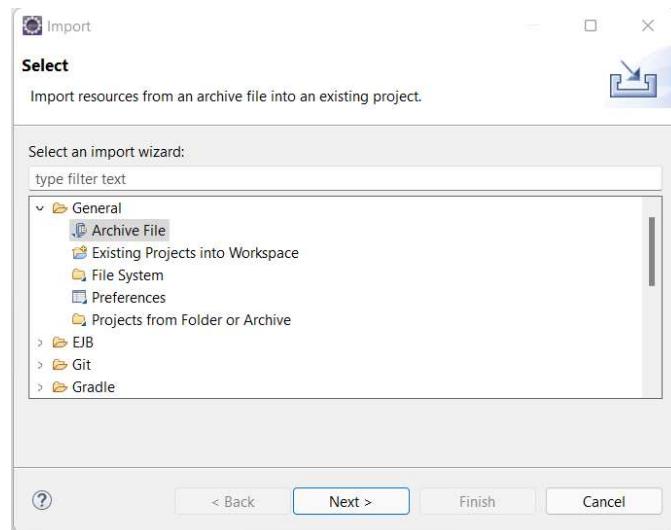
Quando fazemos o download, o arquivo é baixado em extensão “.zip” e deve ser importado na IDE desejada; para tanto, neste exemplo utilizaremos o Eclipse para importar o projeto.

Figura 4 | Importação do projeto no Eclipse



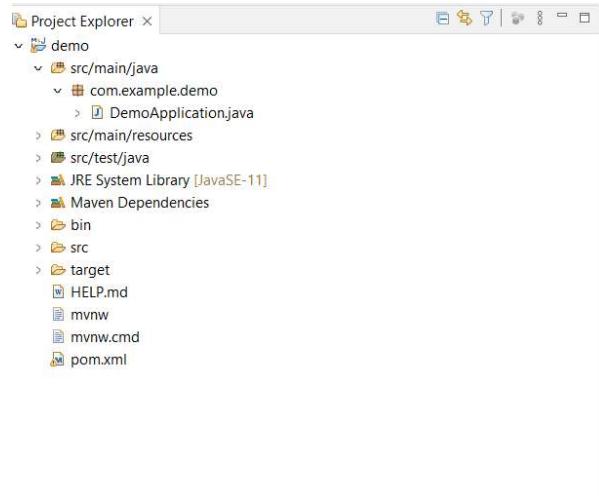
Nesse momento, é necessário selecionar o tipo do projeto em “General >> Archive File”, como mostrado na imagem:

Figura 5 | Selezionando o tipo do projeto



Além desse método, o projeto também pode ser descompactado e inserido no workspace do Eclipse, o que nos permite apenas abrir o projeto da fonte local sem ter que importá-lo. Após feito esse processo de importação, o projeto ganha forma e deve ficar com a seguinte estrutura:

Figura 6 | Estrutura do projeto importado



Fonte: captura de tela do Eclipse.

Para rodar o projeto, basta abrir a classe *DemoApplication.java* e executá-lo. A seguinte saída é esperada:

Figura 7 | Saída da execução do projeto

```

2021-09-21 15:39:25,525 INFO 17540 --- [           main] com.example.demo.DemoApplication      : Starting DemoApplication using Java 17 on LAPTOP-381630F6 with PID 17540
2021-09-21 15:39:25,527 INFO 17540 --- [           main] com.example.demo.DemoApplication      : No active profile set, falling back to default profiles: default
2021-09-21 15:39:25,868 INFO 17540 --- [           main] com.example.demo.DemoApplication      : Started DemoApplication in 0.604 seconds (JVM running for 0.838)

```

Fonte: captura de tela do Eclipse.

Com isso, vemos que o projeto foi executado com sucesso. Agora, basta implementar os códigos para rodá-lo no *server*. Até o momento, as dependências dele foram incluídas e configuradas facilmente, sem muito esforço. O projeto, deste ponto em diante, está pronto para ser decodificado sem que o programador necessite perder horas para configurar as dependências e a estrutura do projeto em si. Nesse sentido, o framework Spring mostra sua utilidade e como são simples sua configuração e seu desenvolvimento.

VIDEOAULA: IMPLEMENTANDO SPRING

O vídeo discorrerá sobre a implementação do Spring, utilizando a linguagem de programação Java e trabalhando com o exemplo dado, alterando apenas alguns aspectos da programação. Ainda, serão mostradas as ferramentas necessárias para que o projeto seja executado com sucesso no IDE proposto, no caso, o Eclipse.

Videoaula: Implementando Spring

Para visualizar o objeto, acesse seu material digital.

EXECUTANDO SPRING BOOT COM TOMCAT

Executando Spring Boot com Tomcat

Projetos criados por start.spring.io contêm o Spring Boot, uma estrutura que deixa o Spring pronto para funcionar dentro do seu aplicativo, mas sem códigos ou configurações necessários. Spring Boot é a maneira mais rápida e popular de iniciar projetos Spring.

Para este exemplo, criaremos um projeto com Spring Boot a partir do download das dependências de servidor e de outras ferramentas, como mostrado adiante:

Figura 8 | Dependências do projeto

Dependencies

ADD DEPENDENCIES... CTRL + B

Spring Web WEB

Build web, including RESTful, applications using Spring MVC. Uses Apache Tomcat as the default embedded container.

Spring Boot DevTools DEVELOPER TOOLS

Provides fast application restarts, LiveReload, and configurations for enhanced development experience.

Spring Configuration Processor DEVELOPER TOOLS

Generate metadata for developers to offer contextual help and "code completion" when working with custom configuration keys (ex.application.properties/.yml files).

Fonte: Spring (c2013-2021, [s. p.]).

Assim, as seguintes dependências devem ser adicionadas: *Spring Web*, *Spring Boot DevTools* e *Spring Configuration Processor*. Elas contêm todas as ferramentas com que trabalharemos. Após esse procedimento, deve-se gerar o projeto e importar na ferramenta desejada. Na primeira execução, serão mostrados os serviços rodando com o servidor Tomcat, este, por sua vez, é um servidor web java desenvolvido pela fundação Apache. A saída da execução do Spring Boot com as dependências corretas deve aparecer como mostra a Figura 9:

Figura 9 | Saída da execução do projeto

```

2021-09-21 16:16:59.280 INFO 13492 ... [ restartedMain] com.example.demo.DemoApplication      : Starting DemoApplication using Java 17 on LAPTOP-381630F6 with PID 13492 (C:\Users
2021-09-21 16:16:59.282 INFO 13492 ... [ restartedMain] com.example.demo.DemoApplication      : No active profile set, falling back to default profiles: default
2021-09-21 16:16:59.315 INFO 13492 ... [ restartedMain] .DevToolsDefaultPropertiesProcessor : DevTools is currently inactive! Set 'spring.devtools.add-properties' to 'false' or 'skipPropertiesRegistration' to true to add standard logging consider setting the 'logging.level.web' proper
2021-09-21 16:16:59.345 INFO 13492 ... [ restartedMain] .DevToolsDefaultPropertiesProcessor : Additional add standard logging consider setting the 'logging.level.web' proper
2021-09-21 16:16:59.971 INFO 13492 ... [ restartedMain] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat initialized with port(s): 8080 (http)
2021-09-21 16:16:59.973 INFO 13492 ... [ restartedMain] o.apache.catalina.core.StandardService  : Starting service [Tomcat]
2021-09-21 16:16:59.976 INFO 13492 ... [ restartedMain] o.apache.catalina.core.StandardEngine : Starting Servlet engine: [Apache Tomcat/9.0.52]
2021-09-21 16:16:59.977 INFO 13492 ... [ restartedMain] o.apache.catalina.core.StandardContext : Root web application context: initialization completed in 712 ms
2021-09-21 16:17:00.027 INFO 13492 ... [ restartedMain] o.s.b.d.a.ServletWebServerApplicationContext : Root WebApplicationContext: initialization completed in 712 ms
2021-09-21 16:17:00.276 INFO 13492 ... [ restartedMain] o.s.b.d.a.OptionalLiveReloadServer   : LiveReload server is running on port 35729
2021-09-21 16:17:00.318 INFO 13492 ... [ restartedMain] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port(s): 8080 (http) with context path ''
2021-09-21 16:17:00.326 INFO 13492 ... [ restartedMain] com.example.demo.DemoApplication : Started DemoApplication in 1.298 seconds (JVM running for 1.611)

```

Fonte: elaborada pelo autor.

Adicionando código ao projeto

Abra o projeto em seu IDE e localize o arquivo *DemoApplication.java*, na pasta *src/main/java/com/example/demo*. Após aberto, será necessário adicionar métodos e anotações extras, como no exemplo:

Figura 10 | Adicionando método e anotação para teste

```

1 package com.example.demo;
2
3 import org.springframework.boot.SpringApplication;
4 import org.springframework.boot.autoconfigure.SpringBootApplication;
5 import org.springframework.web.bind.annotation.GetMapping;
6 import org.springframework.web.bind.annotation.RequestParam;
7 import org.springframework.web.bind.annotation.RestController;
8
9 @SpringBootApplication
10 @RestController
11 public class DemoApplication {
12
13     public static void main(String[] args) {
14         SpringApplication.run(DemoApplication.class, args);
15     }
16
17     @GetMapping("/hello")
18     public String hello(@RequestParam(value = "name", defaultValue = "World") String name) {
19         return String.format("Hello %s!", name);
20     }
21
22 }

```

Fonte: captura de tela de ... elaborada pelo autor.

O método *"hello()* foi projetado para receber um parâmetro Spring chamado *"name"*, que deve, em seguida, combinar esse parâmetro com a palavra "Hello". Após executar o projeto, obteremos uma saída dessa forma no console.

Figura 11 | Saída de resultado do projeto

```

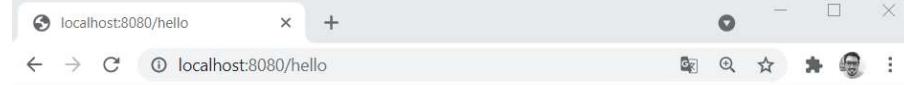
2021-09-21 16:40:51.235 INFO [ restartedMain] com.example.demo.DemoApplication : Starting DemoApplication using Java 17 on LAPTOP-201620F6 with PID : No active profile set, falling back to default profiles: default
2021-09-21 16:40:51.265 INFO [ restartedMain] com.example.demo.DemoApplication : DevTools property defaults active! Set 'spring.devtools.add-prope : For additional web related logging consider setting the 'logging
2021-09-21 16:40:51.265 INFO [e.DevToolsPropertyDefaultsPostProcessor] : e.DevToolsPropertyDefaultsPostProcessor : Tomcat initialized with port(s): 8080 (http)
2021-09-21 16:40:51.918 INFO [ restartedMain] com.example.demo.DemoApplication : Tomcat initialized with port(s): 8080 (http)
2021-09-21 16:40:51.923 INFO [ restartedMain] o.apache.catalina.core.StandardService : Starting service [Tomcat]
2021-09-21 16:40:51.923 INFO [ restartedMain] org.apache.catalina.core.StandardEngine : Starting Servlet Engine: [Apache Tomcat/9.0.52]
2021-09-21 16:40:51.960 INFO [ restartedMain] o.a.c.c.C.[Tomcat].[localhost].[/] : Root WebApplicationContext: initialization completed in 701 ms
2021-09-21 16:40:51.966 INFO [ restartedMain] o.a.c.c.C.[Tomcat].[localhost].[/] : LiveReload server is running on port 35729.
2021-09-21 16:40:52.205 INFO [ restartedMain] o.s.b.d.a.OptionalLiveReloadServer : Tomcat started on port(s): 8080 (http) with context path ''
2021-09-21 16:40:52.246 INFO [ restartedMain] com.example.demo.DemoApplication : Started DemoApplication in 1.281 seconds (JVM running for 1.595)

```

Fonte: captura de tela do Eclipse.

As últimas linhas mostradas na figura nos dizem que o Spring iniciou. O servidor Apache Tomcat, incorporado do Spring Boot, está agindo como um servidor da web e ouvindo solicitações na localhost, porta 8080. Tendo isso em vista, abra seu navegador e, na barra de endereço, no topo, digite <http://localhost:8080/hello>. Você deve obter uma resposta amigável como esta:

Figura 12 | Saída no navegador do projeto Spring



Hello World!

Fonte: captura de tela do navegador.

Teste de unidade (JUnit)

A injecção de dependência deve tornar seu código menos dependente do contêiner do que seria com o desenvolvimento Java EE tradicional. Os POJOs (Plain Old Java Objects) que compõem sua aplicação devem ser testáveis em testes JUnit ou TestNG, com objetos instanciados que usem o *new operador* sem Spring ou qualquer outro contêiner. Para isso, pode-se usar objetos fictícios (em conjunto com outras técnicas de teste valiosas) com o objetivo de testar seu código isoladamente. Se você seguir as recomendações de arquitetura para Spring, as camadas limpas resultantes e os componentes de sua base de código facilitarão o teste de unidade. Por exemplo, você pode testar objetos da camada de serviço fazendo *stub* ou simulando DAO (Organização Autônoma Descentralizada) ou interfaces de repositório, sem a necessidade de acessar dados persistentes durante a execução de testes de unidade.

Os verdadeiros testes de unidade normalmente são executados com extrema rapidez, pois não há infraestrutura de tempo de execução para configurar. Enfatizar verdadeiros testes de unidade como parte de sua metodologia de desenvolvimento pode aumentar sua produtividade.

Para utilizar os testes de unidade juntamente com o projeto, é possível baixar no link <https://junit.org/junit5/> ou então adicionar às dependências do Maven. Assim, o download dos arquivos de configuração do JUnit será feito automaticamente.

VIDEOAULA: EXECUTANDO SPRING BOOT COM TOMCAT

O vídeo para complemento ao que foi estudado no bloco 3 será para demonstrar a utilização do Spring com o servidor Java web Tomcat e um projeto experimental para implementar um pequeno trecho inicial de desenvolvimento com essas ferramentas.

Videoaula: Executando Spring Boot com Tomcat

Para visualizar o objeto, acesse seu material digital.

ESTUDO DE CASO

A empresa onde você trabalha resolveu migrar de um sistema antigo que existia na empresa e que utilizava a linguagem PHP e o servidor Apache. Para isso, solicitou a você que fizesse uma apresentação com demonstrações de possíveis ambientes e linguagens que pudessem substituir as atuais ferramentas. Além disso, a empresa pediu-lhe que pensasse em uma ferramenta e em uma linguagem mais robustas, as quais trouxessem mais segurança e que fossem mais fáceis de utilizar e configurar.

RESOLUÇÃO DO ESTUDO DE CASO

Para a resolução da situação-problema, você precisará de um ambiente de programação composto por uma linguagem robusta e por ferramentas confiáveis, como o Java e seus aplicativos.

Java é uma linguagem de programação e uma plataforma de computação lançada pela primeira vez pela Sun Microsystems em 1995. Existem muitos aplicativos e sites que não funcionarão a menos que você tenha o Java instalado, e mais são criados todos os dias. Java é rápido, seguro e confiável, por isso deve ser admitido como opção.

Para a configuração de máquina, será necessário ter uma configuração dos ambientes integrados de desenvolvimento (IDE) e de frameworks como Junit, Spring, Hibernate, Maven, entre outros. Ainda, pode-se dizer que poderá utilizar os conceitos de AOP para facilitar a programação.

Resolução do Estudo de Caso

Para visualizar o objeto, acesse seu material digital.

6º Saiba mais

Para iniciar com o framework Spring, clique em: <https://start.spring.io/>; para baixar e saber mais sobre os testes de unidade, acesse: <https://junit.org/junit5/>; e para as dependências de projeto, utilize Maven em: <https://maven.apache.org/>.

Aula 4

FRAMEWORK SPRING: BACK-END

O padrão DAO (Organização Autônoma Descentralizada) é um dos mais importantes e comumente usados em aplicativos J2EE, tanto é que a arquitetura de acesso a dados do Spring fornece suporte sofisticado diretamente para ele.

45 minutos

INTRODUÇÃO

O padrão DAO (Organização Autônoma Descentralizada) é um dos mais importantes e comumente usados em aplicativos J2EE, tanto é que a arquitetura de acesso a dados do Spring fornece suporte sofisticado diretamente para ele. A principal API (*Applications Protocol Interface*) de persistência para bancos de dados relacionais no mundo Java é certamente JPA (*Java Persistence API*), que tem seu próprio módulo Spring Data.

Nesse sentido, aprenderemos a construir aplicações com o Spring por meio de padrões de projetos, pois especificam relações genéricas entre os elementos abstratos do projeto, o que o torna bem mais produtivo.

Portanto, é necessário que você busque o aprendizado sobre os padrões para compreender melhor o mundo da programação orientada a objetos e também o dos frameworks com os quais estamos trabalhando. Alguns padrões básicos podem fornecer um entendimento base importante para implantação e gerenciamento de grandes projetos de software.

ACESSO A DADOS COM JPA

Acesso a dados com JPA

Como uma especificação, a *Java Persistence API* (JPA) preocupa-se com a persistência, o que significa vagamente qualquer mecanismo que faz os objetos Java sobreviverem ao processo de aplicativo que os criou. Nem todos os objetos Java precisam ser persistidos, mas a maioria dos aplicativos persiste nos principais objetos de negócios. A especificação JPA permite definir quais objetos devem ser persistidos e como eles devem ser em seus aplicativos Java (SPRING, c2021a).

O que será construído?

Construiremos um aplicativo que armazena Customer POJOs (*Plain Old Java Objects*) em um banco de dados baseado em memória.

Iniciando com Spring Initializr

Para inicializar o projeto:

1. Navegue até <https://start.spring.io>. Esse serviço extrai todas as dependências de que você precisa para um aplicativo e faz a maior parte da configuração para você.
2. Escolha *Gradle* ou *Maven* e o idioma que deseja usar.
3. Escolha a linguagem Java.
4. Clique em *Dependencies* e selecione Spring Data JPA, H2 Database e Spring Boot DevTools.
5. Clique em Gerar.
6. Baixe o arquivo ZIP resultante, que é um arquivo de aplicativo da web configurado com suas escolhas.
7. Se o seu IDE possui a integração Spring Initializr, você pode concluir esse processo a partir do seu IDE.

Definindo uma entidade simples

Nesse exemplo, vamos criar uma entidade chamada *Customer* dentro de um novo pacote, que vamos nomear de “*accessingdatajpa*”. A estrutura ficará desta forma:

(src/main/java/com/example/accessingdatajpa/Customer.java) e seguirá o padrão da Figura 1:

Figura 1 | Classe *Customer*

```

1 package com.example.accessingdatajpa;
2
3 import javax.persistence.Entity;
4 import javax.persistence.GeneratedValue;
5 import javax.persistence.GenerationType;
6 import javax.persistence.Id;
7
8 @Entity
9 public class Customer {
10
11@   @Id
12@   @GeneratedValue(strategy=GenerationType.AUTO)
13@   private Long id;
14@   private String firstName;
15@   private String lastName;
16
17@   protected Customer() {}
18
19@   public Customer(String firstName, String lastName) {
20     this.firstName = firstName;
21     this.lastName = lastName;
22   }
23
24@   @Override
25@   public String toString() {
26     return String.format(
27       "Customer[id=%d, firstName='%s', lastName='%s']",
28       id, firstName, lastName);
29   }
30
31@   public Long getId() {
32     return id;
33   }
34
35@   public String getFirstName() {
36     return firstName;
37   }
38
39@   public String getLastName() {
40     return lastName;
41   }

```

Fonte: elaborada pelo autor.

Aqui você tem uma classe *Customer* com três atributos, *id*, *firstName* e *lastName*, além de dois construtores. O construtor padrão existe apenas para fins de JPA. Você não o usa diretamente, por isso é designado como *protected*. O outro construtor é aquele que você usa para criar instâncias *Customer* que serão salvas no banco de dados.

A classe *Customer* é anotada com *@Entity*, indicando que é uma entidade JPA. Como não existe nenhuma anotação com *@Table*, presume-se que essa entidade seja mapeada para uma tabela chamada *Customer*.

A propriedade *Customer* do objeto *id* é anotada com *@Id* para que a JPA reconheça como o ID do objeto. A propriedade *id* também é anotada com *@GeneratedValue* para indicar que o ID deve ser gerado automaticamente.

As outras duas propriedades, *firstName* e *lastName*, não são anotadas. Presume-se, então, que elas sejam mapeadas para colunas que compartilham os mesmos nomes das próprias propriedades.

O *toString()* é o método conveniente que imprime as propriedades do cliente.

Nessa etapa inicial, construímos a primeira parte da implantação de dados via armazenamento de memória e, consequentemente, temos a primeira etapa da construção JPA. Vale lembrar que a camada ORM (*Object-Relational Mapping*) é adaptadora: ela adapta a linguagem dos gráficos de objetos à linguagem SQL e às tabelas relacionais. A camada ORM permite que desenvolvedores orientados a objetos criem um software que persiste nos dados sem nunca sair do paradigma orientado a objetos.

Ao usar o JPA, você cria um mapa do armazenamento de dados para os objetos do modelo de dados do seu aplicativo. Desse modo, em vez de definir como os objetos são salvos e recuperados, você determina o mapeamento entre os objetos e seu banco de dados e, em seguida, chama o JPA para persisti-los.

VIDEOAULA: ACESSO A DADOS COM JPA

O vídeo a ser criado deverá demonstrar as ferramentas utilizadas para elaborar a aplicação, dentre elas a IDE Eclipse, o framework Maven e o Spring inicializador, que dá acesso ao framework Spring Boot.

Videoaula: Acesso a dados com JPA

Para visualizar o objeto, acesse seu material digital.

CRIANDO CONSULTAS SIMPLES COM SPRING

Criando consultas simples com Spring

Para ver como isso funciona, crie uma interface de repositório que opere com entidade *Customer*, como em “src/main/java/com/example/accessingdatajpa/CustomerRepository.java”, e que mostre a seguinte listagem:

Figura 2 | Interface JPA

```
1 package com.example.accessingdatajpa;
2
3 import java.util.List;
4
5 import org.springframework.data.repository.CrudRepository;
6
7 public interface Accessingdatajpa extends CrudRepository<Customer, Long> {
8
9     List<Customer> findByLastName(String lastName);
10
11    Customer findById(long id);
12 }
```

Fonte: elaborada pelo autor.

CustomerRepository estende a *CrudRepository*. O tipo de entidade e de ID com que se trabalha, *Customer* e *Long*, é especificado nos parâmetros genéricos em *CrudRepository*. Ao estender *CrudRepository*, *CustomerRepository* herda vários métodos para trabalhar com a persistência da classe *Customer*, incluindo métodos para salvar, excluir e localizar entidades *Customer*.

O Spring Data JPA também permite definir outros métodos de consulta, declarando sua assinatura de método. Por exemplo, *CustomerRepository* inclui o método *findByLastName()*. Em um aplicativo Java típico, você pode esperar escrever uma classe que implemente *CustomerRepository*. No entanto, é isso que torna o Spring Data

JPA tão poderoso: você não precisa escrever uma implementação da interface do repositório. O Spring Data JPA cria uma implementação quando você executa o aplicativo. Agora você pode conectar esse exemplo e ver como ele aparece!

Criando uma classe de aplicação

O Spring Initializr cria uma classe simples para o aplicativo. A lista a seguir mostra a classe que Initializr criou para este exemplo em “src/main/java/com/example/accessingdatajpa/AccessingDataJpaApplication.java”:

Figura 3 | Classe de acesso

```

1 package com.example.accessingdatajpa;
2
3 import java.util.List;
4
5 public interface Accessingdatajpa extends CrudRepository<Customer, Long> {
6
7     List<Customer> findByLastName(String lastName);
8
9     Customer findById(long id);
10
11 }
12
13

```

Fonte: elaborada pelo autor.

`@SpringBootApplication` é uma anotação de conveniência que adiciona todas as seguintes configurações:

- `@Configuration`: marca a classe como uma fonte de definições de *bean* para o contexto do aplicativo.
- `@EnableAutoConfiguration`: diz ao Spring Boot para começar a adicionar *beans* com base nas configurações de *classpath*, de outros *beans* e de várias configurações de propriedade. Por exemplo, se *spring-webmvc* estiver no caminho de classe, essa anotação sinaliza o aplicativo como um aplicativo da web e ativa comportamentos-chave, como configurar um *DispatcherServlet*.
- `@ComponentScan`: diz ao Spring para procurar outros componentes, configurações e serviços no *com/example*, permitindo que ele encontre os controladores.

O método main() usa o método Spring Boot *SpringApplication.run()* para lançar um aplicativo. Você percebeu que não havia uma única linha do *web.xml*? E também que não havia nenhum arquivo *web.xml*? Esse aplicativo da web é 100% Java e, por isso, você não precisa configurar nenhuma ligação ou mexer na estrutura do programa.

Agora você precisa modificar a classe simples que o Initializr criou para você. Para obter a saída (para o console, neste exemplo), você precisa configurar um *logger*. Em seguida, você precisa configurar alguns dados e usá-los para gerar saída. Vamos alterar a classe *AccessingDataJpaApplication* adicionando o código a seguir:

Figura 4 | Adicionando anotação `@bean`

```

19 @Bean
20 public CommandLineRunner demo(CustomerRepository repository) {
21     return (args) -> {
22         // save a few customers
23         repository.save(new Customer("Jack", "Bauer"));
24         repository.save(new Customer("Chloe", "O'Brian"));
25         repository.save(new Customer("Kim", "Bauer"));
26         repository.save(new Customer("David", "Palmer"));
27         repository.save(new Customer("Michelle", "Dessler"));
28
29         // fetch all customers
30         Log.info("Customers found with findAll()");
31         Log.info("-----");
32         for (Customer customer : repository.findAll()) {
33             Log.info(customer.toString());
34         }
35         Log.info("");
36
37         // fetch an individual customer by ID
38         Customer customer = repository.findById(1L);
39         Log.info("Customer found with findById(1L)");
40         Log.info("-----");
41         Log.info(customer.toString());
42         Log.info("");
43
44         // fetch customers by last name
45         Log.info("Customer found with findByLastName('Bauer')");
46         Log.info("-----");
47         repository.findByLastName("Bauer").forEach(bauer -> {
48             Log.info(bauer.toString());
49         });
50         // for (Customer bauer : repository.findByLastName("Bauer")) {
51         //     log.info(bauer.toString());
52         // }
53         Log.info("");
54     };
55 }

```

Fonte: elaborada pelo autor.

A classe *AccessingDataJpaApplication* inclui um método *demo()* em que *CustomerRepository* passa para alguns testes. Primeiro, ele busca *CustomerRepository* a partir do contexto do aplicativo Spring. Em seguida, ele salva alguns objetos *Customer*, demonstrando o método *save()* e configurando alguns dados para trabalhar. Depois ele chama *findAll()* para buscar todos os objetos *Customer* do banco de dados e, após isso, chama *findById()* para buscar um único *Customer* por seu ID. Por fim, ele chama *findByLastName()* para localizar todos os clientes cujo sobrenome seja "Bauer". O método *demo()* retorna um *CommandLineRunnerbean* que executa automaticamente o código quando o aplicativo é iniciado.

VIDEOAULA: CRIANDO CONSULTAS SIMPLES COM SPRING

Serão explicados em vídeo a funcionalidade do *bears* no Java e o motivo de se utilizar as anotações dentro da linguagem. Na prática os logs mostram o funcionamento e a importância do retorno desse framework para o programa em questão.

Videoaula: Criando consultas simples com Spring

Para visualizar o objeto, acesse seu material digital.

CONSTRUINDO O JAR DA APLICAÇÃO

Construindo o JAR da Aplicação

Você pode executar o aplicativo a partir da linha de comando com *Gradle* ou *Maven*; também pode construir um único arquivo JAR executável, que contém todas as dependências, classes e recursos necessários, e executá-lo. A construção de um JAR executável facilita o envio, a versão e a implantação do serviço como um aplicativo em todo o ciclo de vida de desenvolvimento, em diferentes ambientes.

Se você usa o *Gradle*, pode executar o aplicativo usando *./gradlew bootRun*. Como alternativa, pode construir o arquivo JAR usando *./gradlew build* e, em seguida, executar o arquivo JAR da seguinte maneira:

** java -jar build/libs/gs-accessing-data-jpa-0.1.0.jar*

Se você usa o *Maven*, pode executar o aplicativo usando *./mvnw spring-boot:run*. Como alternativa, pode ainda construir o arquivo JAR com *./mvnw clean package* e, em seguida, executar o arquivo JAR, da seguinte forma:

** java -jar target/gs-accessing-data-jpa-0.1.0.jar*

Ao executar seu aplicativo, você verá uma saída semelhante a esta:

Figura 5 | Saída da execução do aplicativo

```
== Customers found with findAll():
Customer[id=1, firstName='Jack', lastName='Bauer']
Customer[id=2, firstName='Chloe', lastName='O'Brian']
Customer[id=3, firstName='Kim', lastName='Bauer']
Customer[id=4, firstName='David', lastName='Palmer']
Customer[id=5, firstName='Michelle', lastName='Dessler']

== Customer found with findById(1L):
Customer[id=1, firstName='Jack', lastName='Bauer']

== Customer found with findByLastName('Bauer'):
Customer[id=1, firstName='Jack', lastName='Bauer']
Customer[id=3, firstName='Kim', lastName='Bauer']
```

Fonte: elaborada pelo autor.

Repositórios JPA

O *Java Persistence API* (JPA) é uma especificação do Java e é usado para persistir dados entre o objeto Java e o banco de dados relacional. O JPA atua como uma ponte entre os modelos de domínio orientado a objetos e os sistemas de banco de dados relacionais.

Como o JPA é apenas uma especificação, ele não executa nenhuma operação sozinho. Requer uma implementação. Portanto, ferramentas ORM como Hibernate, TopLink e iBatis implementam especificações JPA para persistência de dados.

A primeira versão do *Java Persistence API*, JPA 1.0, foi lançada em 2006 como parte da especificação EJB 3.0.

A seguir estão as outras versões de desenvolvimento lançadas sob a especificação JPA:

JPA 2.0 – Essa versão foi lançada no ano de 2009, e os recursos importantes dela estão listados a seguir:

- Suporta validação.
- Expande a funcionalidade do mapeamento relacional de objetos.
- Compartilha o objeto de suporte de cache.

JPA 2.1 – O JPA 2.1 foi lançado em 2013 com os seguintes recursos:

- Permite a busca de objetos.
- Fornece suporte para atualização/exclusão de critérios.
- Gera esquema.

JPA 2.2 – O JPA 2.2 foi lançado como um desenvolvimento de manutenção em 2017. Alguns de seus recursos importantes são:

- Suporta Java 8 *Date and Time*.
- Fornece a anotação @Repeatable, que pode ser usada quando queremos aplicar as mesmas anotações a uma declaração ou uso de tipo.
- Permite que a anotação JPA seja usada em meta-anotações.
- Fornece a capacidade de transmitir um resultado de consulta.

A configuração do Spring Data JPA pode ser realizada por meio de duas configurações possíveis, entre elas:

- *Spring Namespace* – configuração XML.
- Configuração baseada em anotação – configuração Java.

O módulo JPA do Spring Data contém um *namespace* customizado que permite definir *beans* de repositório. Ele também contém determinados recursos e atributos de elemento que são especiais para JPA. Geralmente, os repositórios JPA podem ser configurados usando o elemento *repositories* como mostrado a seguir:

Figura 6 | Exemplo de configuração JPA usando namespace

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:jpa="http://www.springframework.org/schema/data/jpa"
       xsi:schemaLocation="http://www.springframework.org/schema/beans
                           https://www.springframework.org/schema/beans/spring-beans.xsd
                           http://www.springframework.org/schema/data/jpa
                           https://www.springframework.org/schema/data/jpa/spring-jpa.xsd">

    <jpa:repositories base-package="com.acme.repositories" />

</beans>
```

Fonte: elaborada pelo autor.

O suporte a repositórios Spring Data JPA pode ser ativado não apenas por meio de um *namespace* XML, mas também usando uma anotação por meio de JavaConfig, conforme mostrado no exemplo a seguir:

Figura 7 | Exemplo de configuração JPA usando anotações

```

19@ 19@  @Bean
20@ 20@  public CommandLineRunner demo(CustomerRepository repository) {
21@ 21@  return (args) -> {
22@ 22@  // save a few customers
23@ 23@  repository.save(new Customer("Jack", "Bauer"));
24@ 24@  repository.save(new Customer("Chloe", "O'Brian"));
25@ 25@  repository.save(new Customer("Kim", "Bauer"));
26@ 26@  repository.save(new Customer("David", "Palmer"));
27@ 27@  repository.save(new Customer("Michelle", "Dessler"));
28@ 28@ 
29@ 29@  // fetch all customers
30@ 30@  Log.info("Customers found with findAll():");
31@ 31@  Log.info("-----");
32@ 32@  for (Customer customer : repository.findAll()) {
33@ 33@   Log.info(customer.toString());
34@ 34@ }
35@ 35@ Log.info("");
36@ 36@ 
37@ 37@  // fetch an individual customer by ID
38@ 38@  Customer customer = repository.findById(1L);
39@ 39@  Log.info("Customer found with findById(1L):");
40@ 40@  Log.info("-----");
41@ 41@  Log.info(customer.toString());
42@ 42@  Log.info("");
43@ 43@ 
44@ 44@  // fetch customers by last name
45@ 45@  Log.info("Customer found with findByLastName('Bauer')");
46@ 46@  Log.info("-----");
47@ 47@  repository.findByLastName("Bauer").forEach(bauer -> {
48@ 48@   Log.info(bauer.toString());
49@ 49@ });
50@ 50@  // for (Customer bauer : repository.findByLastName("Bauer")) {
51@ 51@  // log.info(bauer.toString());
52@ 52@  // }
53@ 53@  Log.info("");
54@ 54@ };
55@ 55@ }

```

Fonte: elaborada pelo autor.

VIDEOAULA: CONSTRUINDO O JAR DA APLICAÇÃO

Serão tratados os métodos de utilização do modo JPA seja por anotações utilizando o beans (recurso próprio do Java), seja utilizando namespaces para JPA, que é a configuração XML dentro da aplicação que está sendo criada, independentemente dos frameworks utilizados.

Videoaula: Construindo o JAR da Aplicação

Para visualizar o objeto, acesse seu material digital.

ESTUDO DE CASO

Para criar uma aplicação em uma corporação, pediram para você ficar com a parte *back-end* de um projeto a ser rodado em um servidor remoto. Para tanto, sugeriram que utilize a linguagem de programação Java.

Para construir essa aplicação, deixaram à sua escolha as ferramentas e os frameworks a serem utilizados. Nesse sentido, você precisa decidir quais serão utilizados. Sendo assim, imagine o cenário e tente pensar em uma solução formidável para o *case*, levando em consideração que será necessária a escolha também do banco de dados da aplicação.

RESOLUÇÃO DO ESTUDO DE CASO

Tendo em vista a criação de uma aplicação para empresas de qualquer porte, é necessário pensar em qual linguagem será utilizada primeiro. Neste caso, como já está dada a linguagem a ser utilizada, fica fácil pensar em ferramentas incríveis de utilização para o desenvolvimento do sistema.

Primeiramente, devemos pensar em qual banco de dados utilizaremos, qual tipo de persistência será mais segura e também em relação à complexidade utilizada para configuração. Ao longo da aula, vimos que a persistência em Java existe em dois modos JPA (XML e *annotations*), basta aqui você verificar qual seria a mais simples para você. As *annotations* são próprias do Java e têm suas vantagens, porém o XML é um arquivo fácil de configurar e permite alteração em tempo real. O banco de dados fará a diferença no sentido de que ele será remoto, então precisamente você terá de optar pelo Spring, um framework que apresenta uma utilização rápida e eficaz.

Resolução do Estudo de Caso

Para visualizar o objeto, acesse seu material digital.

6º Saiba mais

Para se aprofundar mais no assunto, entre nos projetos exemplos do Spring em:

<https://spring.io/projects>.

Para banco de dados em nuvem, utilize: <https://spring.io/projects/spring-cloud>.

São ótimas soluções que podem ser implantadas do zero e que nos ensinam a utilização básica das ferramentas.

REFERÊNCIAS

4 minutos

Aula 1

CHRISTENSSON, P. Framework. **TechTerms**, [S. I.], 7 mar. 2013. Disponível em:

<https://techterms.com/definition/framework>. Acesso em: 21 ago. 2021.

FRAMEWORKS // Dicionário do Programador. Rio de Janeiro: HostGator, [s. d.]. 1 vídeo (6 min.). Publicado pelo canal Código Fonte TV. Disponível em: https://www.youtube.com/watch?v=2zqzzTnfa0E&ab_channel=C%C3%B3digoFonteTV. Acesso em: 28 dez. 2021.

SPRING. **Spring**, [S. I.], c2021. Disponível em: <https://spring.io/>. Acesso em: 28 dez. 2021.

Aula 2

APACHE. **The Apache software foundation**, [S. I.], c2021. Disponível em: <https://www.apache.org/>. Acesso em: 29 dez. 2021.

DEVMEDIA. Front-end Web na prática. DevMedia, Rio de Janeiro, [s. d.]. Disponível em: <https://www.devmedia.com.br/guia/front-end-web-na-pratica/38151>. Acesso em: 13 set. 2021.

ECLIPSE. Download Eclipse Technology that is right for you. **Eclipse Foundation**, [S. I.], 2021a. Disponível em: <https://www.eclipse.org/downloads/>. Acesso em: 29 dez. 2021.

ECLIPSE. **Eclipse foundation**, [S. I.], 2021b. Disponível em: <https://www.eclipse.org/>. Acesso em: 29 dez. 2021.

LEITE, A. F. Frameworks e Padrões de Projeto. **DevMedia**, Rio de Janeiro, 2006. Disponível em: <https://www.devmedia.com.br/frameworks-e-padroes-de-projeto/1111>. Acesso em: 11 set. 2021.

MAVEN. Welcome to Apache Maven. **Apache Maven Project**, [S. I.], c2002-2021. Disponível em: <https://maven.apache.org/>. Acesso em: 1 out. 2021.

MUTIMUTEMA, T. 10 of the Most Popular Java Frameworks of 2020. **Stackify**, Huntington Beach, 24 fev. 2020. Disponível em: <https://stackify.com/10-of-the-most-popular-java-frameworks-of-2020/>. Acesso em: 1 set. 2021.

SPRING. **Spring**, [S. I.], c2021. Disponível em: <https://spring.io/>. Acesso em: 28 dez. 2021.

Aula 3

JOHNSON, R. et al. Introduction to Spring Framework. In: JOHNSON, R. et al. **Spring Framework Reference Documentation**. [S. I.]: Spring, c2004-2016. Disponível em: <https://docs.spring.io/spring-framework/docs/3.2.x/spring-framework-reference/html/overview.html>. Acesso em: 22 set. 2021.

JUNIT. The 5th major version of the programmer-friendly testing framework for Java and the JVM. **JUnit 5**, [S. I.], c2021. Disponível em: <https://junit.org/junit5/>. Acesso em: 30 dez. 2021.

LEITE, A. F. Frameworks e Padrões de Projeto. **DevMedia**, Rio de Janeiro, 2006. Disponível em: <https://www.devmedia.com.br/frameworks-e-padroes-de-projeto/1111>. Acesso em: 11 set. 2021.

MAVEN. Welcome to Apache Maven. **Apache Maven Project**, [S. /], c2002-2021. Disponível em: <https://maven.apache.org/>. Acesso em: 1 out. 2021.

SPRING. Accessing Data with JPA. **Spring**, [S. /], c2021a. Disponível em: <https://spring.io/guides/gs/accessing-data-jpa/>. Acesso em: 25 set. 2021.

SPRING. Spring Framework. **Spring**, [S. /], c2021b. Disponível em: <https://spring.io/projects/spring-framework>. Acesso em: 25 set. 2021.

SPRING. **Spring initializr**, [S. /], c2013-2021. Disponível em: <https://start.spring.io/>. Acesso em: 30 dez. 2021.

Aula 4

CADU. ORM: Object Relational Mapper. **DevMedia**, Rio de Janeiro, 2011. Disponível em: <https://www.devmedia.com.br/orm-object-relational-mapper/19056>. Acesso em: 27 set. 2021.

SPRING. Accessing Data with JPA. **Spring**, [S. /], c2021a. Disponível em: <https://spring.io/guides/gs/accessing-data-jpa/>. Acesso em: 25 set. 2021.

SPRING. Projects. **Spring**, [S. /], c2021b. Disponível em: <https://spring.io/projects>. Acesso em: 30 dez. 2021.

SPRING. Spring Cloud. **Spring**, [S. /], 2021a. Disponível em: <https://spring.io/projects/spring-cloud>. Acesso em: 30 dez. 2021.

SPRING. Spring Framework Overview. **Spring**, [S. /], 2021b. Disponível em: <https://docs.spring.io/spring-framework/docs/current/reference/html/overview.html#overview>. Acesso em: 27 set. 2021.

SPRING. **Spring initializr**, [S. /], c2013-2021. Disponível em: <https://start.spring.io/>. Acesso em: 30 dez. 2021.