<LAB365>

SPRING INICIAL

<LAB365> SENAI

AGENDA

- Configuração de ambiente
- O protocolo HTTP
- O começo do Java na web
- Como os servlets funcionam?
- O Apache Tomcat
- Fluxo básico de uma requisição
- Aplicações tradicionais *versus* API's
- JSON (Javascript Object Notation)
- Criando um novo projeto

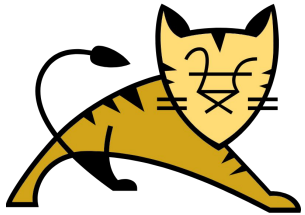
CONFIGURAÇÃO DE AMBIENTE

Nota: Para esses primeiros exemplos com Servlets, estarei usando o **Eclipse**, uma outra IDE com suporte gratuito para aplicações Java EE.

- Usaremos o **Maven** para gerenciamento de dependências e automação da construção de executáveis da aplicação, o processo de ***build***, como é chamado.
- Usaremos o **Apache Tomcat** como o servidor de nossa aplicação, para que possamos acessar as funcionalidades por meio de chamadas HTTP.
- Usaremos o **Postman**, para executar chamadas HTTP para a API, conseguindo assim testar de forma organizada: https://www.postman.com/downloads/

CONFIGURAÇÃO DE AMBIENTE: TOMCAT

- O download do Tomcat pode ser feito em: https://tomcat.apache.org/download-90.cgi
- Após o download, deve ser feita a extração no diretório desejado.
- O Tomcat não precisa de configuração extra, podemos simplesmente chamá-lo dentro da IDE, quando necessário.

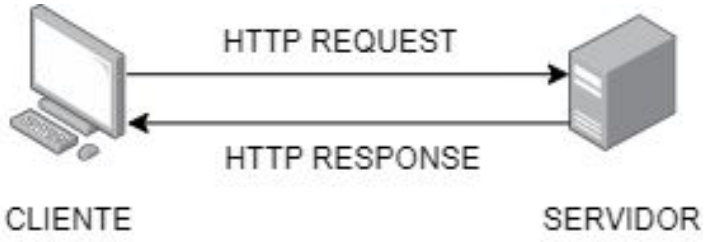


O PROTOCOLO HTTP

O protocolo HTTP (*Hypertext Transfer Protocol*) se responsabiliza pela transferência de dados entre redes no contexto da web. Ele atua sob uma arquitetura **cliente/servidor**, onde uma requisição é realizada pelo **cliente**, que aguarda uma resposta do **servidor**.

Para que a intenção da comunicação seja entendida pelo servidor, o protocolo utiliza alguns métodos (também chamado **verbos**), sendo os mais comuns:

- GET
- POST
- PUT
- DELETE



Para saber mais: Protocolo HTTP

O COMEÇO DO JAVA NA WEB

- O Java começou na web com os **applets**, pequenos programas escritos em Java que eram incorporados em páginas web e executados no navegador do usuário. No entanto, por problemas de desempenho e segurança, não houve uma boa adoção da tecnologia.
- Surgem então os **Servlets**, que são componentes Java executados do lado do servidor para gerar dinamicamente conteúdo web.
    - Criação da especificação para padronizar o desenvolvimento web com Java.
    - Introdução das **Java Server Pages (JSP)** para deixar as páginas mais dinâmicas.
    - **Não** é um framework!!

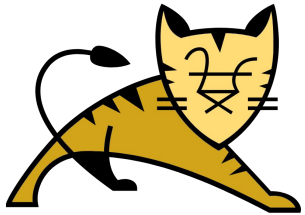
Para saber mais: Servlets

COMO OS SERVLETS FUNCIONAM?

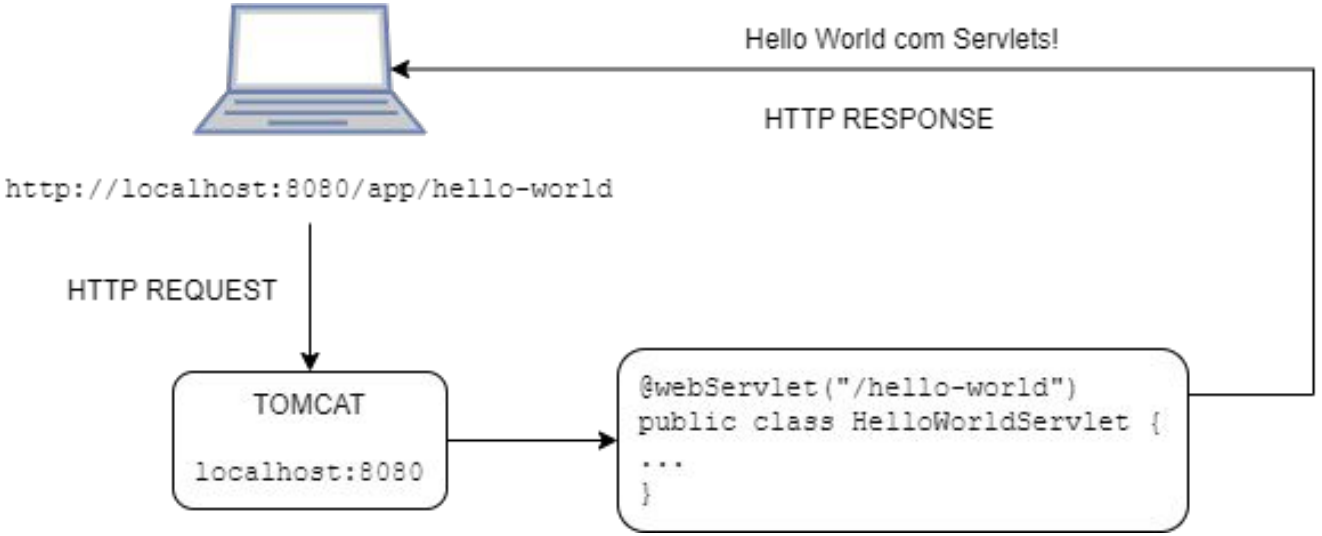
- No geral, os Servlets são classes Java programadas para processar requisições HTTP e gerar conteúdo dinâmico como resposta às requisições dos clientes.
- Em um servidor executando aplicações com Servlets, as requisições são interceptadas e encaminhadas ao **contêiner de servlets**, que serve como um guia para que o servidor/programa identifique qual a classe responsável por lidar com a requisição.
- A classe responsável (Servlet) então processa a requisição e produz uma resposta, que é retornada ao usuário (cliente).

O APACHE TOMCAT

- O **Apache Tomcat** é um servidor de aplicações web que funciona em conjunto com os Servlets, lidando com a interceptação das requisições feitas, e identificando qual destes Servlets dentro da aplicação deve lidar com o processamento de determinada requisição.
- Ele também é responsável pelo gerenciamento do ciclo de vida dos Servlets, criando a instância da classe quando requisitado, e excluindo-a quando em desuso.



FLUXO BÁSICO DE UMA REQUISIÇÃO



APLICAÇÕES TRADICIONAIS

Aplicações “tradicionais” costumam ser acopladas, o que significa que o back-end (Java/Servlets) não são acessados diretamente e demandam uma interface de usuário (HTML/JSP) para permitirem a interação de usuários.

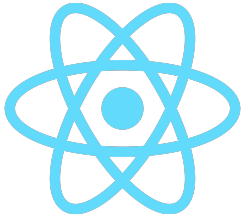
Isso faz com que as aplicações não possam ser flexíveis em sua comunicação com uma variedade de sistemas externos.

- Integração com complexidade desnecessária.
- Maior custo de desenvolvimento e manutenção.
- Escalabilidade limitada.

A SOLUÇÃO: API's

As **API's (Application Programming Interface)** são mecanismos para que dois componentes de software se comuniquem usando um conjunto de definições e protocolos.

Em nosso cenário, uma API (back-end) forneceria meios (endpoints) para que usuários de uma aplicação front-end escrita em **React** ou **Angular**, ou mesmo através de uma requisição feita no **Postman** pudessem realizar operações.



API’s REST

Uma das arquiteturas mais populares de API’s é a **REST (Representational State Transfer)**, que define um conjunto de funções como GET, POST, PUT e DELETE, que os clientes podem usar para acessar os dados do servidor.

Sim, elas trabalham muito bem com HTTP, que é o meio pelo qual clientes e servidores trocam dados em aplicações com essa arquitetura.

- APIs privadas
- APIs públicas

VANTAGENS DO USO DE API's

- São mais simples de integrar e mais escaláveis, sendo menos passíveis de problemas e conflitos enquanto o sistema cresce.
- Fácil manutenção, onde cada sistema pode ser ajustado separadamente, sem interferências.
- Baixo acoplamento, permitindo que diferentes aplicações consumam dados dela, e sem a obrigação de se adaptar a um front-end.

JSON (JAVASCRIPT OBJECT NOTATION)

O **JSON (Javascript Object Notation)** é um formato de dados leve e legível por humanos, e é recomendado para uso por ser simples e ter suporte nativo em várias linguagens de programação, além de ser amplamente adotado, o que permite maior compatibilidade com sistemas.

```
{
   “registration”:”12345”,
   “name”:”student”,
   “email”:”student@example.com”
}
```

JSON (JAVASCRIPT OBJECT NOTATION)

Apesar da compatibilidade de aplicações modernas com o JSON, aplicações mais antigas, como as que usam Servlets, ainda não possuem um suporte nativo, demandando assim uma biblioteca externa.

Uma das mais conhecidas é o Gson, mantida pelo Google. Você pode fazer o download do Jar clicando nesse link, e basta adicioná-lo ao projeto.

CRIANDO UM NOVO PROJETO

No Eclipse IDE, siga os seguintes passos para criar um projeto:

- No menu superior: **File > New > Dynamic Web Project**
- Defina o nome para seu projeto: *crud-servlets-api*
- Após o passo anterior, clique em **next** para as próximas duas telas
- Você chegará às configurações de **web module**, onde deverá especificar o **context root** e o **content directory** da sua aplicação.

CRIANDO UM NOVO PROJETO

- O que é o **context root**?

É o caminho base para sua aplicação. Se você especifica o caminho como ***futurodev***, esse será o endereço base para sua aplicação, que ficará algo como: ***http://localhost:8080/futurodev***.

- O que é o **content directory**?

Esse é o diretório dentro de sua aplicação onde arquivos estáticos ficarão armazenados. Tais como: HTML, CSS, imagens e outros possíveis arquivos.

Nota: Em versões mais recentes do Eclipse, o diretório **webapp** pode estar oculto ao terminar de criar o projeto. Vamos ajustar isso!

CRIANDO UM CRUD API COM SERVLETS

- **@WebServlet** é a anotação utilizada para mapear uma classe como um Servlet, dessa forma permitindo que seja reconhecida pelo contêiner de servlets em tempo de execução. Essa anotação também informa a URL deste Servlet.

```
@WebServlet(“/hello-world”)
```

- **HttpServlet** é a classe de qual todo Servlet herda os métodos para processamento de requisições HTTP. Uma classe só é considerada um Servlet ao herdar dessa classe.

```
@WebServlet(“/hello-world”)
public class HelloWorldServlet extends HttpServlet {
    // do something...
}
```

CONSTRUÇÃO DE UMA API

MÃO NA MASSA!
Agora, iniciamos a construção de uma API para exemplificar o uso de Servlets.

1. **Listagem de estudantes.**
2. Cadastro de estudante.
3. Atualização do cadastro do estudante.
4. Deleção de um estudante.

CONSTRUÇÃO DE UMA API

MÃO NA MASSA!
Agora, iniciamos a construção de uma API para exemplificar o uso de Servlets.

1. Listagem de estudantes.
2. **Cadastro de estudante.**
3. Atualização do cadastro do estudante.
4. Deleção de um estudante.

CONSTRUÇÃO DE UMA API

MÃO NA MASSA!
Agora, iniciamos a construção de uma API para exemplificar o uso de Servlets.

1. Listagem de estudantes.
2. Cadastro de estudante.
3. **Atualização do cadastro do estudante.**
4. Deleção de um estudante.

CONSTRUÇÃO DE UMA API

MÃO NA MASSA!
Agora, iniciamos a construção de uma API para exemplificar o uso de Servlets.

1. Listagem de estudantes.
2. Cadastro de estudante.
3. Atualização do cadastro do estudante.
4. **Deleção de um estudante.**

<LAB365>

SPRING INICIAL

<LAB365> SENAI

AGENDA

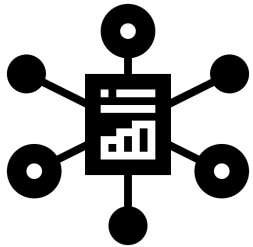
- Frameworks?
- Spring Framework
- Ecossistema Spring e seus módulos
- Spring Boot
- Maven e Gradle
- Spring Initializr
- Dependências do projeto
- Estrutura do projeto
- Entry point da aplicação
- Arquitetura Projeto Spring Boot MVC
- Projeto - Lista de tarefas
    - Propriedades de um controlador

FRAMEWORKS?

Um framework geralmente consiste em uma estrutura de software construída para ser reutilizável, fornecendo funcionalidades e abstrações genéricas para facilitar o desenvolvimento de aplicações.

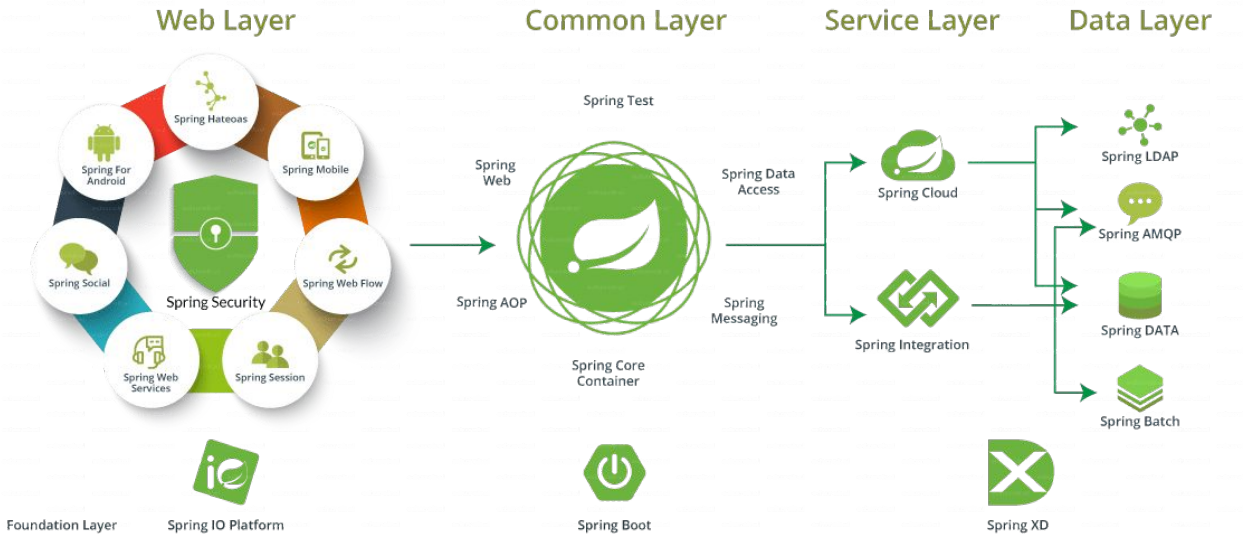
São constituídos de diretrizes, bibliotecas, padrões de projeto e componentes pré-construídos que podem ser usados como base para o desenvolvimento de novas aplicações.

Isso permite que desenvolvedores possam se concentrar na lógica de negócios exclusivas do projeto ao invés de funcionalidades repetitivas.



SPRING FRAMEWORK

O *Spring Framework* fornece uma gama de abordagens para o desenvolvimento de aplicações Java robustas e modulares. Ela promove diversos módulos e bibliotecas para resolver diversos desafios e tornar o desenvolvimento mais ágil.



ECOSSISTEMA SPRING E SEUS MÓDULOS

O ecossistema do Spring é bastante diversificado, abrangendo diversas tecnologias e componentes. Por ser um framework modular, o uso de seus componentes é determinado por quem está desenvolvendo, o que permite aplicações mais leves, contendo apenas o necessário.

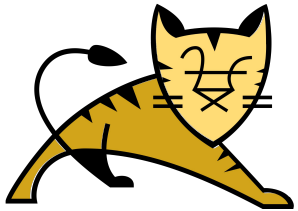
- Spring Web
- Spring Data
- Spring Security
- Spring Boot

Para ver mais sobre os projetos Spring: https://spring.io/projects

SPRING BOOT

O *Spring Boot* é um dos projetos que fazem parte do ecossistema Spring e tem como objetivo a rápida criação de aplicações, sem a preocupação com extensas configurações iniciais.

- Tomcat embutido, sem necessidade de configuração manual.
- Compatibilidade com Maven e Gradle para gerenciamento simplificado de dependências
- Sem necessidade de configurações em arquivos XML.



MAVEN E GRADLE

- Maven possui configurações em XML, enquanto Gradle usa um DSL (*Domain-Specific Language*) baseado em Groovy ou Kotlin.
- Maven é menos flexível, você precisa se adequar às suas convenções, enquanto Gradle é mais flexível.
- Maven pode ser mais lento que o Gradle em suas tarefas, especialmente em aplicações de grande porte.
- Gradle é mais performático.
- Maven possui mais suporte da comunidade e adoção, enquanto Gradle ainda está em período de crescimento.

SPRING INITIALIZR

O Spring Initializr fornece um formulário com várias configurações para a criação de um projeto com uma estrutura pré-definida, já com as dependências que você escolher.

Para criar seu projeto: https://start.spring.io/

SPRING INITIALIZR - CONVENÇÕES

Quando criamos o projeto, é importante seguir algumas *convenções*, que são práticas de desenvolvimento comuns, e que padronizam a estrutura, tornando-a mais legível.

- O *group* segue a nomenclatura da organização que desenvolveu o projeto. No caso do Senai, por exemplo, poderia ser: *senai.com.br*.
- Nomenclatura reversa (br.com.senai).
- Nome do pacote como nome do projeto.

O resultado final poderia ser: *br.com.senai.crudspring*

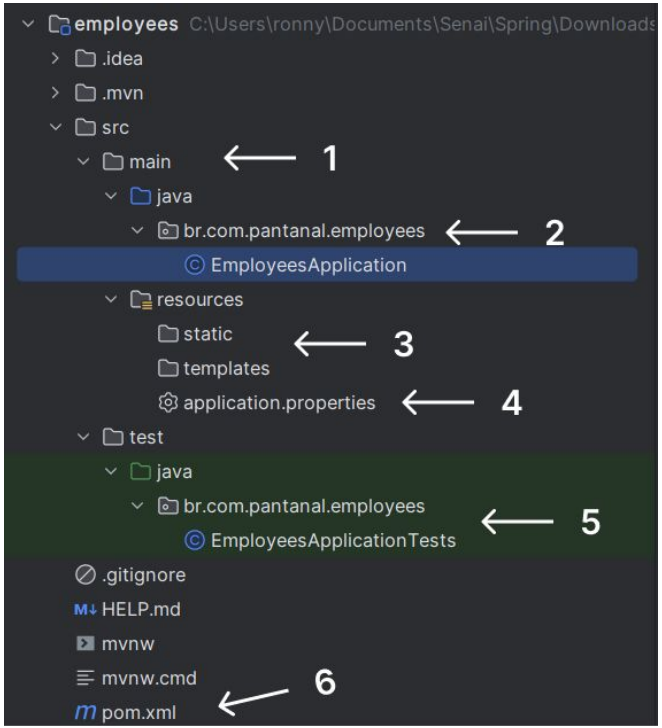
DEPENDÊNCIAS DO PROJETO

Para gerenciar as dependências do projeto, utilizaremos o Apache Maven, que consegue lidar mais facilmente com o gerenciamento, além de ter um amplo repositório de dependências que pode ser usado.

- Spring Boot DevTools
- Spring Web
- Validation (posteriormente)

ESTRUTURA DO PROJETO

1. Diretório onde ficam os arquivos Java. Estrutura *src/main/java*.
2. Pacote base do projeto *Group* + *name* e classe base da aplicação.
3. Diretório *static*, onde ficam arquivos CSS, imagens, dentre outros arquivos estáticos. E *templates*, onde ficariam arquivos HTML.
4. Arquivo de configurações adicionais da aplicação.
5. Classe de teste que já vem com o projeto base, para servir de exemplo.
6. Arquivo XML contendo informações do projeto e a declaração das dependências.



ENTRY POINT DA APLICAÇÃO

Toda aplicação Spring Boot possui uma classe criada por padrão que consiste no nome do projeto, com o sufixo *Application*.

Esta classe possui um método *main* que é o *entry point* da aplicação, ou seja, o ponto de entrada para iniciar tudo o que é necessário para seu projeto em tempo de execução.

- Configuração de banco de dados.
- Configurações de segurança.
- Inicialização dos *beans*.
- Injeção das dependências.

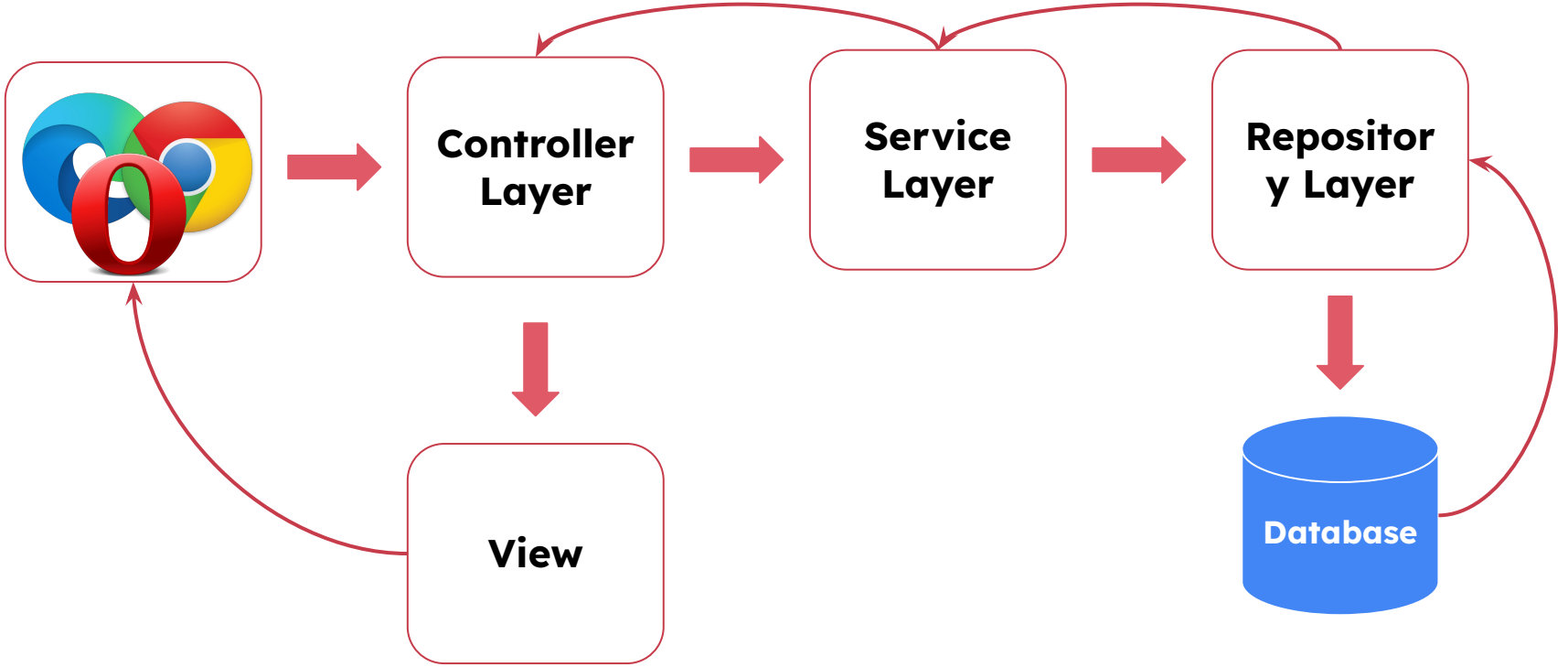
```
import ...

@SpringBootApplication
public class EmployeesApplication {

    public static void main(String[] args) {
        SpringApplication.run(EmployeesApplication.class, args);
    }

}
```

ARQUITETURA DO PROJETO SPRING BOOT MVC



MINI PROJETO - LISTA DE TAREFAS

MÃO NA MASSA!

- Construção de uma API REST que contenha as 4 operações básicas para o gerenciamento de tarefas simples.
- As tarefas contam com um identificador, descrição, data de entrega e data de início, responsável, status e prioridade da tarefa.
- Criar, listar, atualizar e deletar.

MINI PROJETO - LISTA DE TAREFAS

- Como deve ficar o objeto?

```
{
    “id”: 1,
    “description”:”Realizar atividade…”,
    “startDate”:”2023-10-25”,
    “endDate”:”2023-10-26”,
    “status”:”DOING”, // PENDING, DOING, COMPLETED
    “priority”:”LOW”, // LOW, MEDIUM, HIGH
    “assignee”: {
        “id”: 1,
        “name”:”Lucas”
    }
}
```

MINI PROJETO - LISTA DE TAREFAS

- **Criando um objeto para representar o banco de dados em memória.**
- Trabalhando com records.
- Criando um controlador.
- Criando um serviço.
- Listando as tarefas.
- Adicionando tarefas.
- Atualizando tarefas.
- Deletando tarefas.

MINI PROJETO - LISTA DE TAREFAS

- Criando um objeto para representar o banco de dados em memória.
- **Trabalhando com records.**
- Criando um controlador.
- Criando um serviço.
- Listando as tarefas.
- Adicionando tarefas.
- Atualizando tarefas.
- Deletando tarefas.

O QUE SÃO RECORDS?

- Classes imutáveis, ou seja, valores definidos sem possibilidade de alteração.
- Elimina a verbosidade de construtores, *getters*, *setters*, *toString*, *equals*, *hashcode*, etc.
- Fácil legibilidade, poucas linhas de código.
- https://docs.oracle.com/en/java/javase/16/language/records.html

```
public record Task(String description) {

}
```

MINI PROJETO - LISTA DE TAREFAS

- Criando um objeto para representar o banco de dados em memória.
- Trabalhando com records.
- **Criando um controlador.**
- Criando um serviço.
- Listando as tarefas.
- Adicionando tarefas.
- Atualizando tarefas.
- Deletando tarefas.

PROPRIEDADES DE UM CONTROLADOR

- Similar a estrutura dos Servlets que construímos para nossa API com Servlets.
- Configurado através de anotações.

- @RestController

Como o nome sugere, informa ao Spring que nossa classe age como um controlador REST que receberá requisições HTTP.

- @RequestMapping

Define qual o nome que identifica o controlador, para que o Spring saiba para onde redirecionar a requisição, similar ao *path* que usamos nos @WebServlets.

PROPRIEDADES DE UM CONTROLADOR

Aplicações Spring trabalham com requisições através de anotações (*annotations*)!

@PostMapping: Mapeia uma requisição em um controller para receber requisições POST.

@RequestBody: Sinaliza que dado objeto recebido como parâmetro é o corpo da requisição.

Como o objeto chega ao controlador quando realizamos uma requisição?

Como ele é transformado em um objeto Java?

PROPRIEDADES DE UM CONTROLADOR

- **Serialização**: Converter um objeto em um formato que possa ser transportado pela rede, como JSON, por exemplo.
- **Desserialização**: Converter um JSON, por exemplo, em um objeto.

O Spring já possui uma ferramenta de serialização/desserialização embutida, então não precisamos nos preocupar em importar bibliotecas externas manualmente e lidar com isto!

PROPRIEDADES DE UM CONTROLADOR

Quando executarmos uma operação, queremos retornar o status HTTP apropriado, além de retornar objetos caso necessário. Para isso, podemos usar o **ResponseEntity**.

- Representa toda a resposta HTTP, incluindo status, corpo e cabeçalhos.

MINI PROJETO - LISTA DE TAREFAS

- Criando um objeto para representar o banco de dados em memória.
- Trabalhando com records.
- Criando um controlador.
- **Criando um serviço.**
- Listando as tarefas.
- Adicionando tarefas.
- Atualizando tarefas.
- Deletando tarefas.
- Validando objetos.

MINI PROJETO - LISTA DE TAREFAS

- Criando um objeto para representar o banco de dados em memória.
- Trabalhando com records.
- Criando um controlador.
- Criando um serviço.
- **Listando as tarefas.**
- Adicionando tarefas.
- Atualizando tarefas.
- Deletando tarefas.
- Validando objetos.

MINI PROJETO - LISTA DE TAREFAS

- Criando um objeto para representar o banco de dados em memória.
- Trabalhando com records.
- Criando um controlador.
- Criando um serviço.
- Listando as tarefas.
- **Adicionando tarefas.**
- Atualizando tarefas.
- Deletando tarefas.
- Validando objetos.

MINI PROJETO - LISTA DE TAREFAS

- Criando um objeto para representar o banco de dados em memória.
- Trabalhando com records.
- Criando um controlador.
- Criando um serviço.
- Listando as tarefas.
- Adicionando tarefas.
- **Atualizando tarefas.**
- Deletando tarefas.
- Validando objetos.

MINI PROJETO - LISTA DE TAREFAS

- Criando um objeto para representar o banco de dados em memória.
- Trabalhando com records.
- Criando um controlador.
- Criando um serviço.
- Listando as tarefas.
- Adicionando tarefas.
- Atualizando tarefas.
- **Deletando tarefas.**
- Validando objetos.

MINI PROJETO - LISTA DE TAREFAS

- Criando um objeto para representar o banco de dados em memória.
- Trabalhando com records.
- Criando um controlador.
- Criando um serviço.
- Listando as tarefas.
- Adicionando tarefas.
- Atualizando tarefas.
- Deletando tarefas.
- **Validando objetos.**

MINI PROJETO - LISTA DE TAREFAS

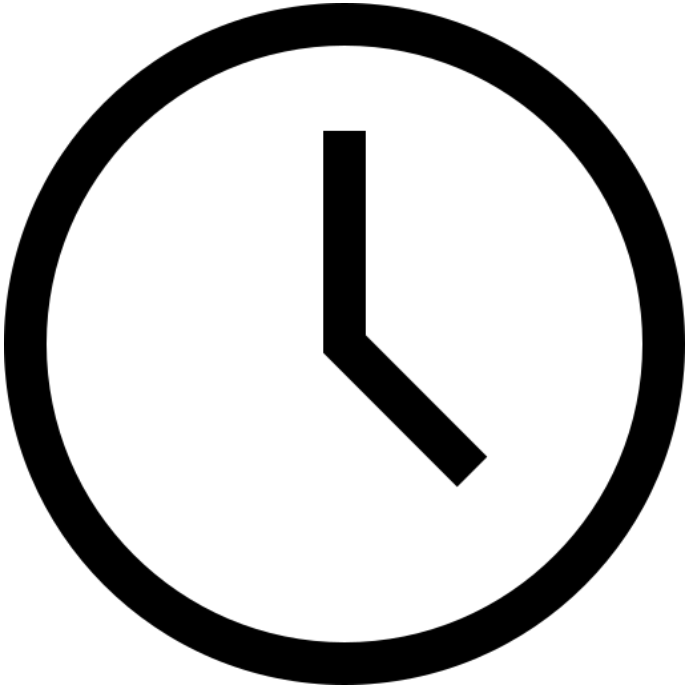
- ***Baeldung*** *(https://www.baeldung.com/rest-with-spring-series):* Possui diversos tutoriais com exemplos e explicações que ajudam no desenvolvimento.
- ***Spring*** *(https://spring.io/quickstart):* Site oficial do Spring, com vários guias de aprendizado.
- ***Livro: Vire o Jogo com Spring Framework****:* Aborda diversos conceitos, desde o básico até temas mais avançados envolvendo o Spring.

INTERVALO!

Finalizamos o nosso primeiro período de hoje. Que tal descansar um pouco?!

Nos vemos em 20 minutos.

Início: 20:20
Retorno: 20:40



AVALIAÇÃO DOCENTE

O que você está achando das minhas aulas neste conteúdo?

Clique aqui ou escaneie o QRCode ao lado para avaliar minha aula.

Sinta-se à vontade para fornecer uma avaliação sempre que achar necessário.

<LAB365>