

# CETEJ35 - Java Web - JAVA\_XXX (2024\_01)

[Meus cursos](#) / [CETEJ35 - Web \(2024\\_01\)](#) / [Semana 02: 09/09 a 15/09](#) / [Página Dinâmica](#)

## Página Dinâmica

✓ **Feito:** Ver

✓ **Feito:** Gastar pelo menos 20 minutos na atividade

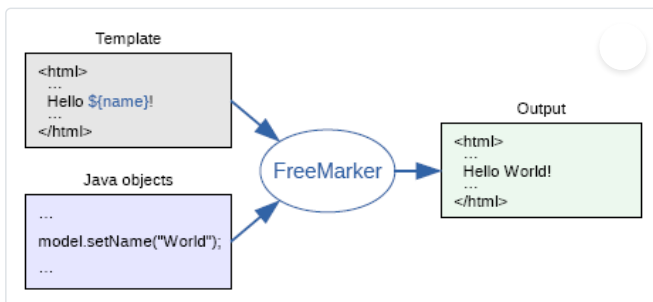
✓ **Feito:** Passar pela atividade até o fim

**Fecha:** segunda-feira, 2 dez. 2024, 00:00

Nesta aula, nós transformamos a página estática em uma página dinâmica. Isso é necessário porque queremos que a tabela de cidades seja atualizada à medida que novas cidades são inseridas. Para isso, vamos precisar de mais uma tecnologia - o Freemarker. Em seguida, mudamos a página existente para uma nova pasta. Assim, o Spring Boot reconhece a página como uma página dinâmica. Também alteramos a extensão da página. O próximo passo é colocar o código dinâmico na página, usando a sintaxe do Freemarker. Também fazemos os ajustes necessários para implementar o padrão MVC no projeto.

## TORNANDO UMA PÁGINA DINÂMICA COM FREEMARKER

O Spring Boot, por meio do [Spring MVC](#), fornece suporte para criação de páginas dinâmicas. Isso é feito com o auxílio de uma [template engine](#). A *template engine* substitui os valores fixos por valores variáveis na página Web durante o processamento da requisição (Figura abaixo). O Spring Boot coopera com esse processo fornecendo o que a *template engine* precisa para fazer seu trabalho.



O Spring Boot se integra muito bem com várias *template engines*. Nesse curso, nós vamos usar o [Freemarker](#). O Freemarker é um projeto estável da Apache Software Foundation. Ele possui as características necessárias para criar uma página dinâmica que atende uma gama de projetos.

<#FREEMARKER>
Home
Manual
Java API
Contribute
Report a Bug
Download

# What is Apache FreeMarker™?

WHAT IS APACHE FREEMARKER™?

DOWNLOAD / MAVEN

DOCUMENTATION

- Manual
- Java API
- Manual Chinese translation

TOOLING

- Editor and IDE plugins
- Online template tester
- File-to-file transformers

COMMUNITY

- Report bugs
- Report security vulnerability
- Ask help on Stack Overflow
- Source code (Git)
- Get news on Twitter
- Discuss on mailing lists
- Who uses FreeMarker?
- Contributors wanted!
- Committer how-to
- Project history

ASF

## What is Apache FreeMarker™?

Apache FreeMarker™ is a *template engine*: a Java library to generate text output (HTML web pages, e-mails, configuration files, source code, etc.) based on templates and changing data. Templates are written in the FreeMarker Template Language (FTL), which is a simple, specialized language (not a full-blown programming language like PHP). Usually, a general-purpose programming language (like Java) is used to prepare the data (issue database queries, do business calculations). Then, Apache FreeMarker displays that prepared data using templates. In the template you are focusing on how to present the data, and outside the template you are focusing on what data to present.

This approach is often referred to as the MVC (Model View Controller) pattern, and is particularly popular for dynamic web pages. It helps in separating web page designers (HTML authors) from developers (Java programmers usually). Designers won't face complicated logic in templates, and can change the appearance of a page without programmers having to change or recompile code.

While FreeMarker was originally created for generating HTML pages in MVC web application frameworks, it isn't bound to servlets or HTML or anything web-related. It's used in non-web application environments as well.

See the [Manual](#) for more details...

A primeira coisa para adicionar o Freemarker ao nosso projeto é adicionar a dependência do Freemarker no `pom.xml` do nosso projeto. As linhas 30 a 34 na Figura a seguir mostram a nova dependência inserida no `pom.xml`.

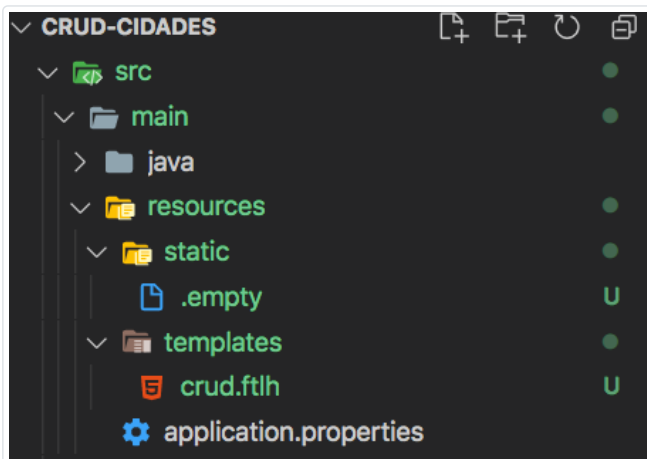
```

19      <dependencies>
20          <dependency>
21              <groupId>org.springframework.boot</groupId>
22              <artifactId>spring-boot-starter-web</artifactId>
23          </dependency>
24
25          <dependency>
26              <groupId>org.springframework.boot</groupId>
27              <artifactId>spring-boot-starter-test</artifactId>
28              <scope>test</scope>
29          </dependency>
30
31          <dependency>
32              <groupId>org.springframework.boot</groupId>
33              <artifactId>spring-boot-starter-freemarker</artifactId>
34          </dependency>
35      </dependencies>

```

O Spring Boot é um *framework opinionated*. Isso significa que ele usa configurações padrão. Como o Spring Boot tem suporte nativo ao Freemarker, ele já pré-configura tudo que é necessário para que o Freemarker funcione no projeto. Agora, só precisamos colocar as coisas no lugar certo e inserir o código.

Para que o Spring Boot reconheça nossa página dinâmica com o Freemarker, precisamos mover a página `crud.html` (atualmente estática) para a pasta `/resources/templates/`. Essa é a pasta padrão para páginas dinâmicas no Spring Boot. Outra alteração é a mudança da extensão da página. O Spring Boot reconhece por padrão páginas dinâmicas com o Freemarker desde que a página use a extensão `.ftlh`. A Figura a seguir mostra como ficou a estrutura de arquivos após essas modificações.



O próximo passo é alterar o código estático da página usando a sintaxe do Freemarker. Vamos alterar a tabela para que as linhas da tabela sejam criadas dinamicamente de acordo com uma lista de cidades. Observe a diretiva `list` do Freemarker nas linhas 41 e 52, na Figura abaixo.

```
32 <table class="table table-striped table-hover mt-5">
33   <thead class="thead-dark">
34     <tr>
35       <th>Nome</th>
36       <th>Estado</th>
37       <th>Ações</th>
38     </tr>
39   </thead>
40   <tbody>
41     <#list listaCidades as cidade >
42       <tr>
43         <td>Londrina</td>
44         <td>Paraná</td>
45         <td>
46           <div class="d-flex d-justify-content-center">
47             <a class="btn btn-warning mr-3">ALTERAR</a>
48             <a class="btn btn-danger">EXCLUIR</a>
49           </div>
50         </td>
51       </tr>
52     </#list>
53   </tbody>
54 </table>
```


Uma diretiva é uma instrução do Freemarker que faz algo no código. Nesse caso, a diretiva `list` é usada para iterar sobre uma lista, chamada `listaCidades`. Essa diretiva funciona de forma similar ao `forEach` no Java. Cada cidade na lista será mapeada para uma variável chamada `cidade`. Depois, vamos usar essa variável para extrair o nome e o estado da cidade, nas linhas 43 e 44. Mas, por enquanto, vamos manter o código como está.

Retroceder

Avançar

◀ Verificação de aprendizado - Gerenciamento de Cidades

Seguir para...

✉ Contate o suporte do site 

Você acessou como RAFAEL ROCHA DA SILVA PROENÇA (Sair)  
CETEJ35 - Web (2024\_01)

Tema

Adaptable

Boost

Clássico

Campus

Apucarana

Campo Mourão

Cornélio Procópio

Curitiba

Dois Vizinhos

Francisco Beltrão

Guarapuava

Londrina

Medianeira

Pato Branco

Ponta Grossa

Reitoria

Santa Helena

Toledo

UTFPR

Ajuda

Chat UTFPR

Calendário Acadêmico

Biblioteca

e-Mail

Nuvem (OwnCloud )

Produção Acadêmica

Secretaria Acadêmica

Sistemas Corporativos

Sistema Eletrônico de Informação - SEI

Suporte ao usuário

Criação de curso

Comunidade

Português - Brasil (pt\_br)



Deutsch (de)

English (en)

Português - Brasil (pt\_br)

Resumo de retenção de dados

Baixar o aplicativo móvel.

 Dê um feedback sobre este software 

Universidade Tecnológica Federal do Paraná - UTFPR  
Suporte ao usuário

