

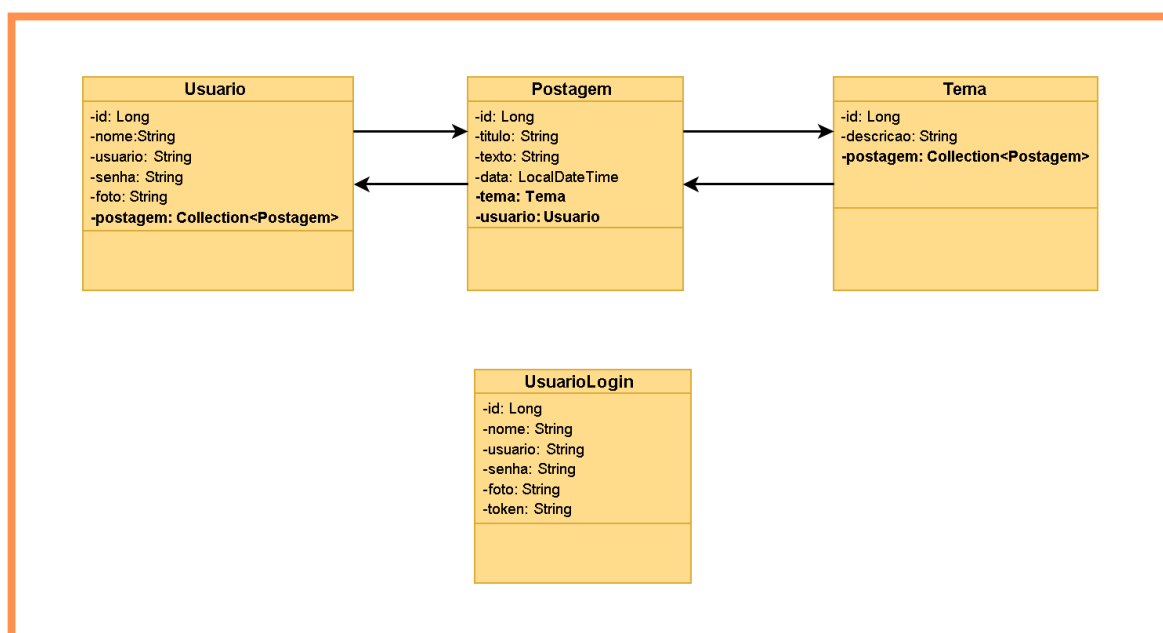
Projeto 02 - Blog Pessoal - CRUD 01

O que veremos por aqui:

1. Apresentação do Projeto
2. Criar o Projeto no Spring Initializr
3. Conhecer o arquivo pom.xml
4. Configurar o Banco de dados da aplicação

1. O Projeto Blog Pessoal

O Projeto Blog Pessoal será o nosso Projeto Guia no aprendizado do Framework Spring e suas principais funcionalidades. Todo o código que implementarmos no projeto Blog Pessoal servirá de base para a construção do Projeto Integrador, que sempre receberá novas funcionalidades depois que você adquirir os conhecimentos necessários através do Blog Pessoal. Veja o Diagrama de Classes do Projeto Blog Pessoal na figura abaixo:



O Projeto será composto por 3 Recursos (Conjunto de Classes e Interfaces responsáveis por mapear um tipo de Objeto e persistir no Banco de dados Relacional) e uma Classe auxiliar:

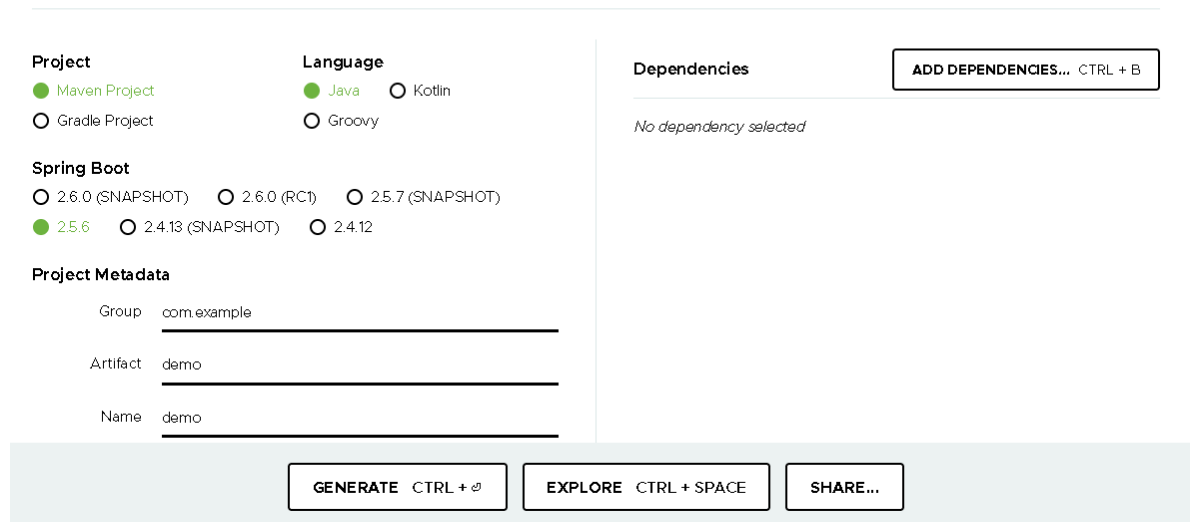
Classe	Descrição
Postagem	Recurso responsável por definir Objeto Postagem (posts) do Blog Pessoal
Tema	Recurso responsável por classificar as postagens através do Objeto Tema
Usuario	Recurso responsável por definir o Objeto Usuário, que poderá acessar e criar postagens no Blog Pessoal
UsuarioLogin	Classe auxiliar, que será utilizada para efetuar login no Blog Pessoal

Os Recursos irão gerar tabelas no Banco de dados da aplicação. A Classe auxiliar não irá gerar uma tabela no Banco de dados da aplicação, ela servirá de Classe auxiliar na implementação da Segurança da aplicação. Os recursos serão implementados na mesma sequência da tabela acima.

Antes de começar a criar as nossas Classes, vamos criar o nosso Projeto Spring Boot no gerador de templates Spring Initializr e configurar o Banco de dados da aplicação.

Passo 01 - Abrindo o Spring Initializr

1. Abra o Navegador de sua preferência e acesse o **Spring Initializr**, através do endereço: <https://start.spring.io>

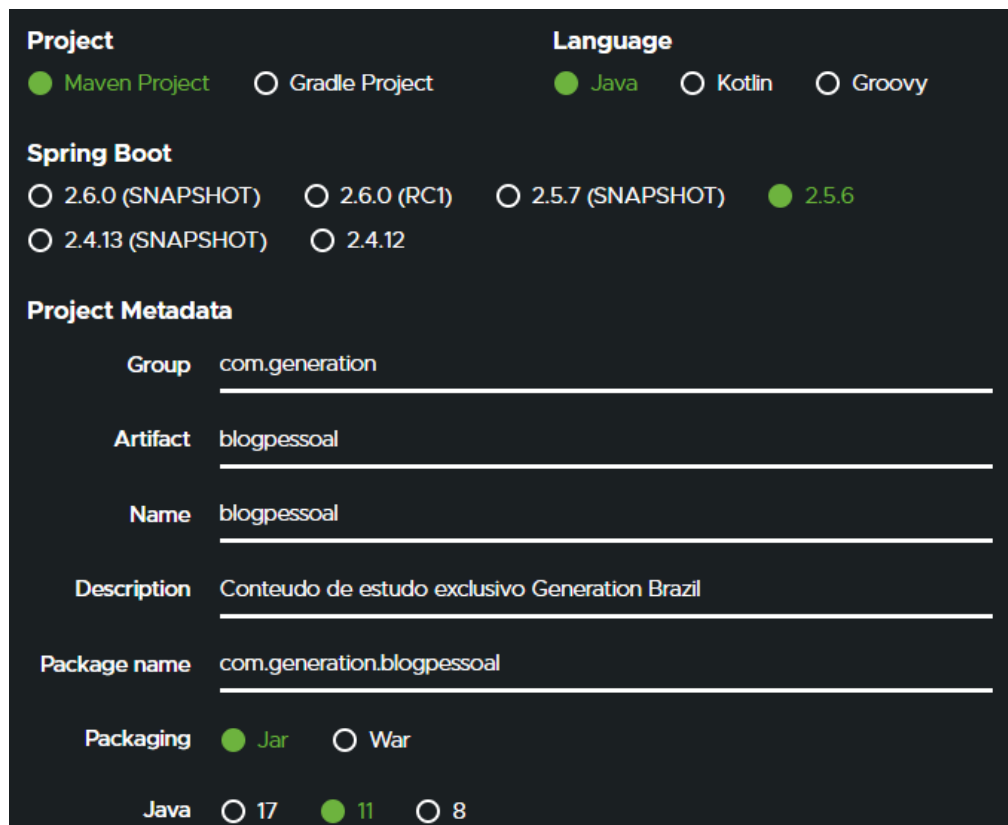


The screenshot shows the Spring Initializr web form. It has three main sections: Project, Language, and Spring Boot. The Project section has radio buttons for Maven Project (selected), Gradle Project, and Groovy. The Language section has radio buttons for Java (selected), Kotlin, and Groovy. The Spring Boot section has radio buttons for 2.6.0 (SNAPSHOT), 2.6.0 (RC1), 2.5.7 (SNAPSHOT), 2.5.6 (selected), 2.4.13 (SNAPSHOT), and 2.4.12. Below these is the Project Metadata section with input fields for Group (com.example), Artifact (demo), and Name (demo). To the right is a Dependencies section with a button 'ADD DEPENDENCIES... CTRL + B' and the text 'No dependency selected'. At the bottom are three buttons: 'GENERATE CTRL + G', 'EXPLORE CTRL + SPACE', and 'SHARE...'.

Passo 02 - Setup do Projeto

2.1 Configurações iniciais

Vamos configurar o template inicial do nosso projeto conforme a figura abaixo:



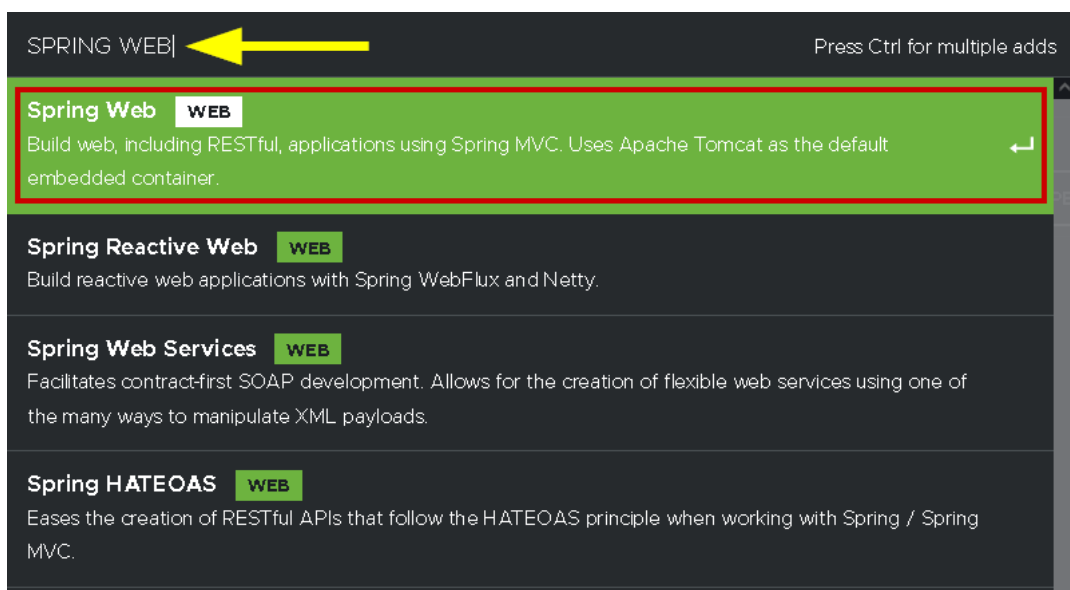
The screenshot shows the Spring Initializr web form with specific configurations. The Project section has radio buttons for Maven Project (selected), Gradle Project, and Groovy. The Language section has radio buttons for Java (selected), Kotlin, and Groovy. The Spring Boot section has radio buttons for 2.6.0 (SNAPSHOT), 2.6.0 (RC1), 2.5.7 (SNAPSHOT), 2.5.6 (selected), 2.4.13 (SNAPSHOT), and 2.4.12. Below these is the Project Metadata section with input fields for Group (com.generation), Artifact (blogpessoal), Name (blogpessoal), Description (Conteudo de estudo exclusivo Generation Brazil), and Package name (com.generation.blogpessoal). At the bottom are two buttons: 'Jar' (selected) and 'War'. Below these are two buttons: 'Java' (selected) and '17' (selected), and a button '8'.

Item	Descrição
Project	Define o Gerenciador de Dependências do Projeto (Maven Project).
Language	Define a Linguagem (Java).
Spring Boot	Define a versão do Spring Boot, que será utilizada. Mantenha a versão indicada pelo Spring Initializr.
Group	O Endereço reverso do Domínio da sua Organização. Exemplo: <i>generation.com</i> → <i>com.generation</i>
Artifact	O artefato a ser gerado, ou seja, o nome da aplicação que será criada (Mesmo nome do projeto).
Name	Nome do Projeto (escrito em letras minúsculas, sem acentos ou espaços).
Description	Breve descrição do projeto.
Package Name	Estrutura do pacote inicial da aplicação (Group + Artifact). Exemplo: <u><i>com.generation.blogpessoal</i></u>
Packaging	Define como a aplicação será empacotada (JAR).
Java Version	Versão do Java (a versão da imagem pode ser diferente da sua tela).

2.2 Dependências

No projeto Blog Pessoal vamos inserir 5 dependências: **Spring Web, Spring Boot Dev Tools, Spring Data JPA, MySQL Driver e Validation.**

1. Na página inicial do **Spring Initializr**, clique no botão **ADD DEPENDENCIES... CTRL + B**
2. Na caixa de pesquisa (indicada na figura abaixo com uma seta amarela), digite o nome da dependência que você deseja adicionar e clique sobre o nome da Dependência para adicionar (indicada na figura com um retângulo vermelho)

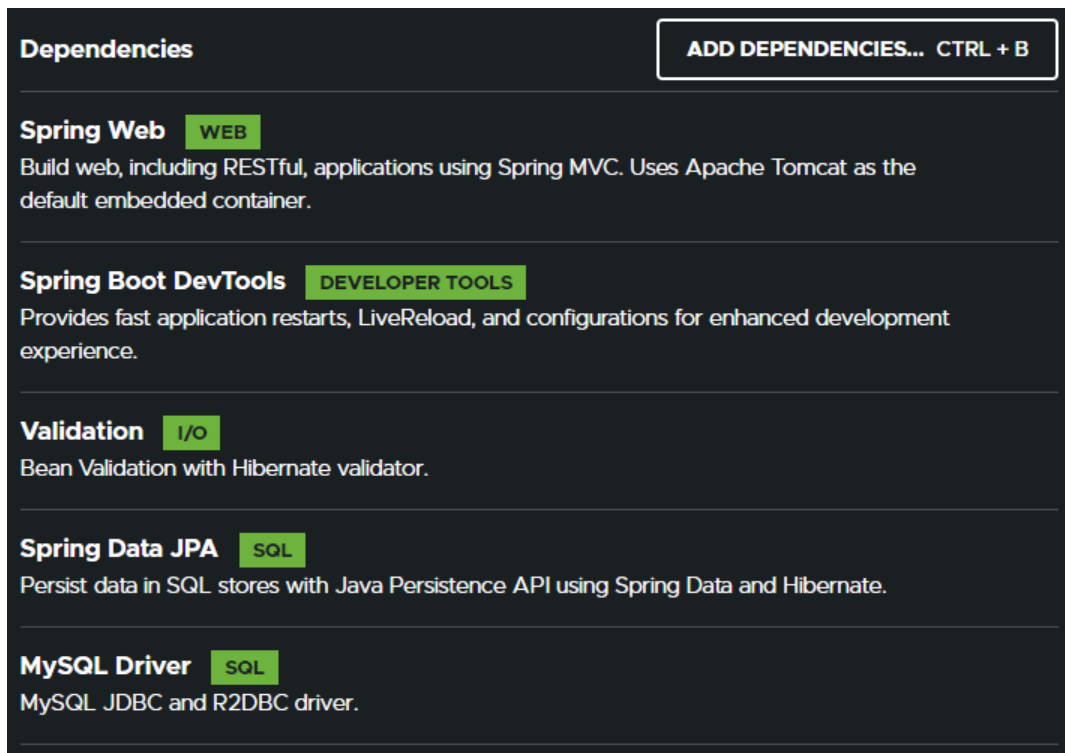


3. Repita os itens 1 e 2 para adicionar as demais Dependências.



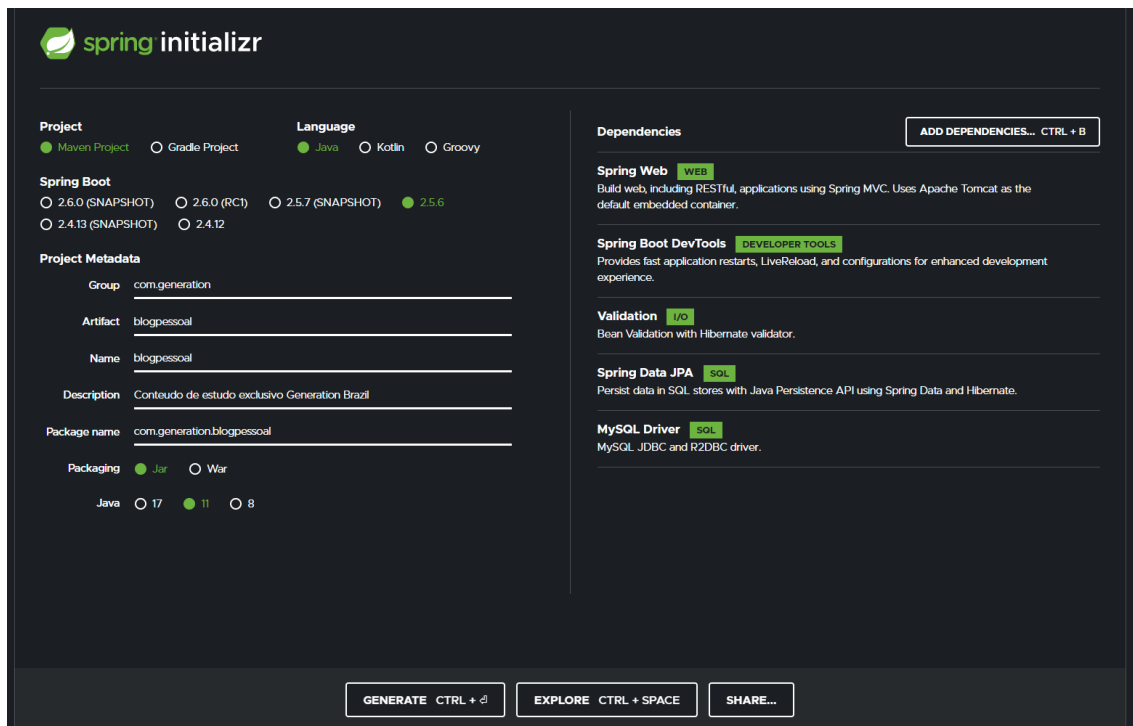
DICA: Mantenha a tecla CTRL pressionada ao clicar sobre a Dependência que será adicionada ao projeto. Desta forma, a janela se manterá aberta e você conseguirá selecionar todas Dependências do projeto de uma só vez.

4. Após adicionar as cinco Dependências, o item Dependencies do Spring Initializr estará igual a figura abaixo:



Dependência	Descrição
Spring Web	Fornecer todas as Bibliotecas necessárias para trabalhar com o protocolo HTTP.
Spring Boot Dev Tools	Permite a atualização do projeto em tempo real durante o processo de Desenvolvimento da aplicação.
Validation	Fornecer um conjunto de anotações que permite validar os atributos das Classes da Camada Model.
Spring Data JPA	Java Persistence API é uma biblioteca que armazena e recupera objetos que são armazenados em bancos de dados.
MySQL Driver	Responsável pela conexão entre nossa aplicação e o Banco de Dados MySQL. Se trocarmos de SGBD precisaremos trocar o Driver da nossa aplicação.

5. O seu projeto deverá estar semelhante a figura abaixo:



The image shows the Spring Initializr web interface. It has a dark theme. At the top left is the 'spring initializr' logo. The interface is divided into several sections: 'Project' with radio buttons for 'Maven Project' (selected) and 'Gradle Project'; 'Language' with radio buttons for 'Java' (selected), 'Kotlin', and 'Groovy'; 'Spring Boot' with radio buttons for versions 2.6.0 (Snapshot), 2.6.0 (RC1), 2.5.7 (Snapshot), 2.5.6 (selected), and 2.4.13 (Snapshot), 2.4.12; 'Project Metadata' with text input fields for 'Group' (com.generation), 'Artifact' (blogpessoal), 'Name' (blogpessoal), 'Description' (Conteudo de estudo exclusivo Generation Brazil), and 'Package name' (com.generation.blogpessoal); 'Packaging' with radio buttons for 'Jar' (selected) and 'War'; and 'Language' with radio buttons for 'Java' (selected), '17', '11' (selected), and '8'. On the right, there is a 'Dependencies' section with a button 'ADD DEPENDENCIES... CTRL + B'. It lists several dependencies: 'Spring Web' (WEB), 'Spring Boot DevTools' (DEVELOPER TOOLS), 'Validation' (I/O), 'Spring Data JPA' (SQL), and 'MySQL Driver' (SQL). At the bottom, there are three buttons: 'GENERATE CTRL + ⌘', 'EXPLORE CTRL + SPACE', and 'SHARE...'.

6. Clique em **Generate Ctrl + ↵** (CTRL + Enter)

7. Faça o Download do Projeto

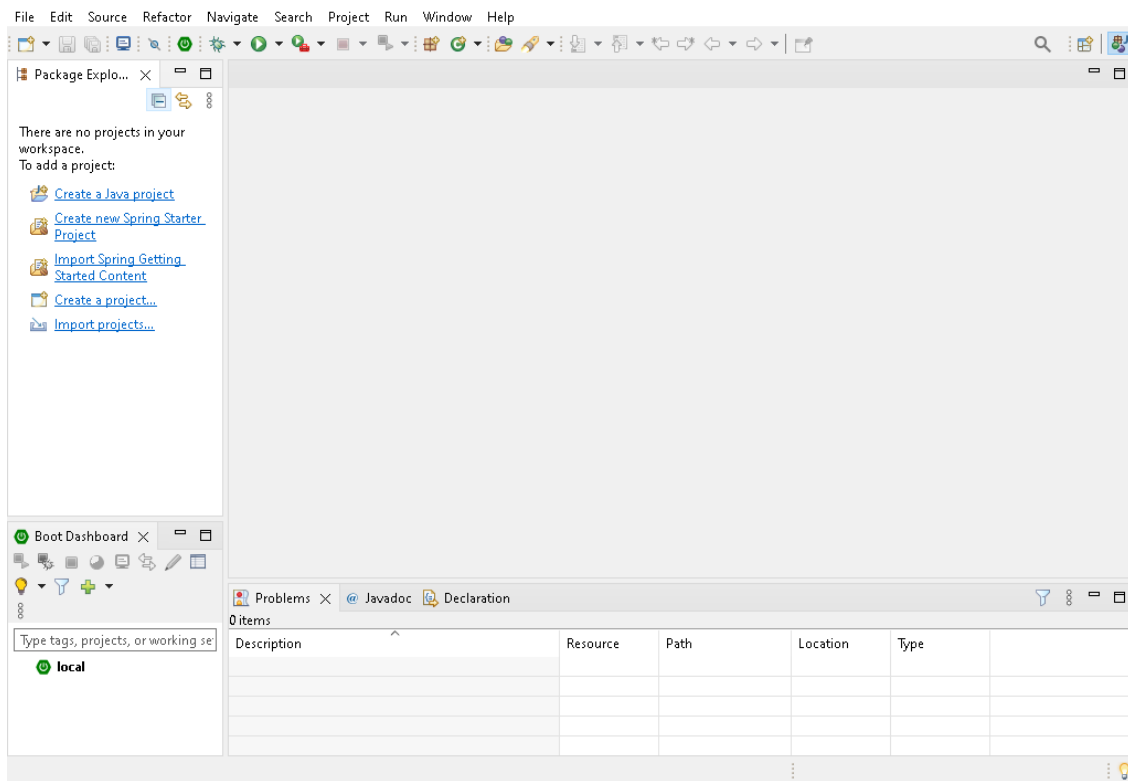
Passo 03 - Importando o Projeto no STS

1. Extraia o arquivo zip dentro da pasta **Workspace** do STS

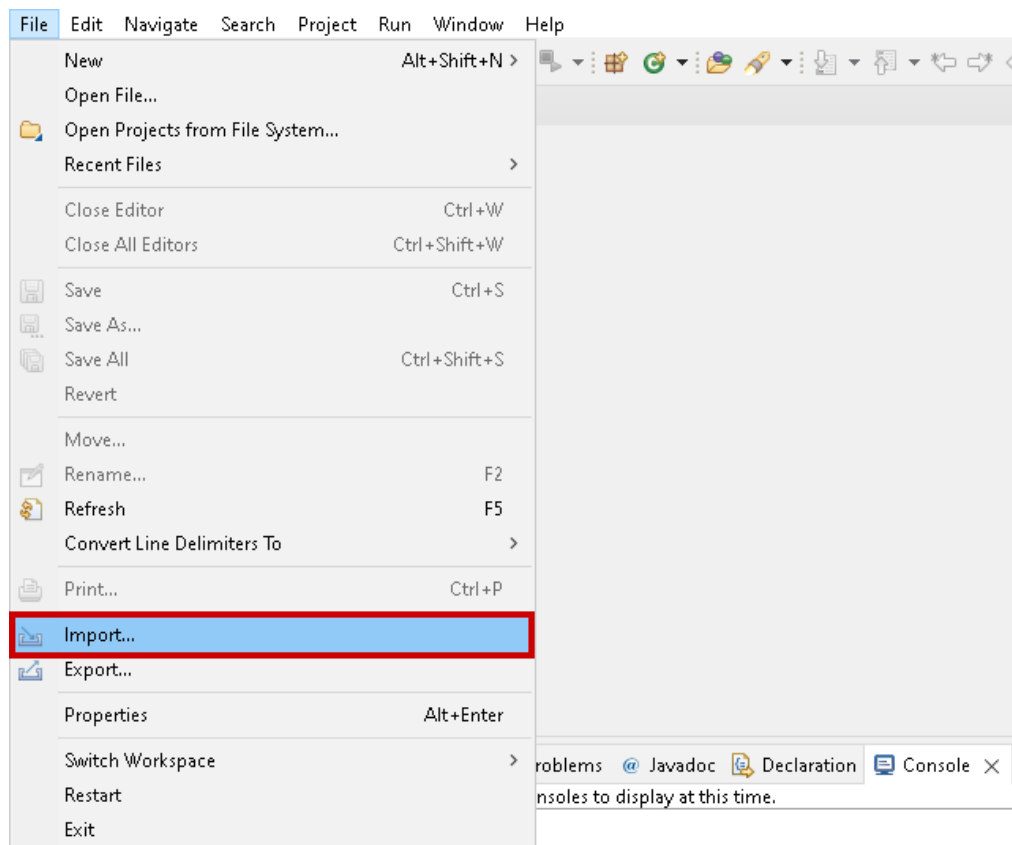


ATENÇÃO: A pasta *Workspace* do STS geralmente está localizada em:
C:\Users\seu usuario\Documents\workspace-spring-tool-suite-4-4.12.1.RELEASE,
onde **4-4.12.1.RELEASE** é a versão do STS

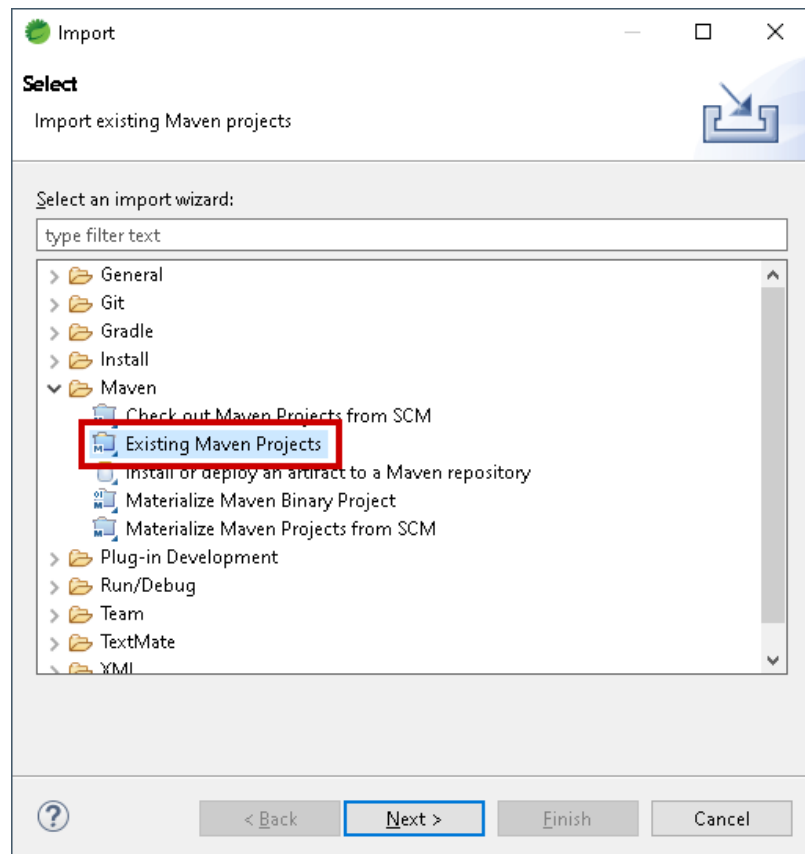
2. Abra o STS



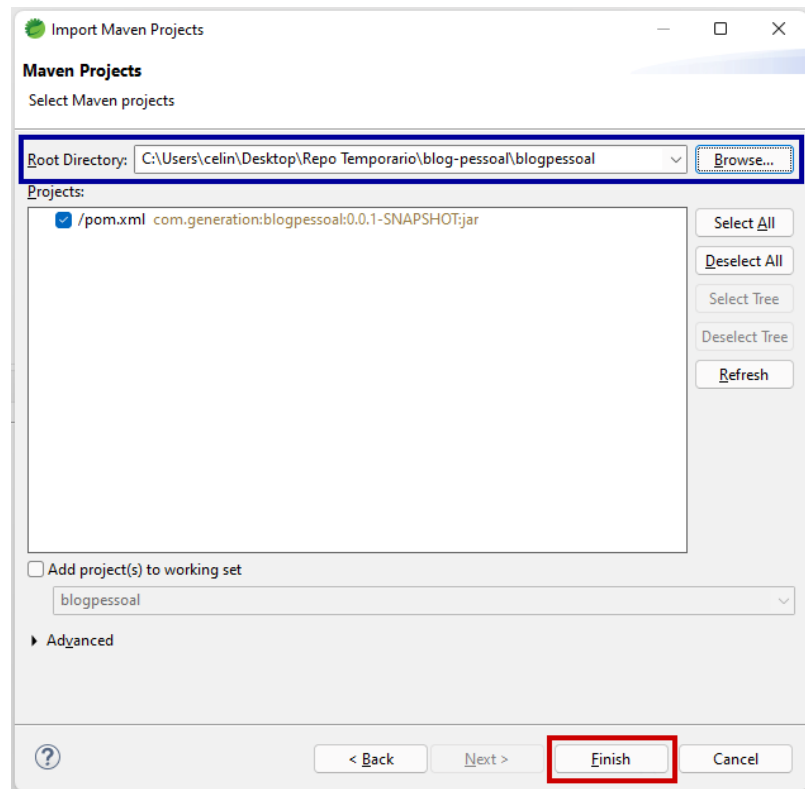
3. Importe o projeto no STS através da opção **File → Import**



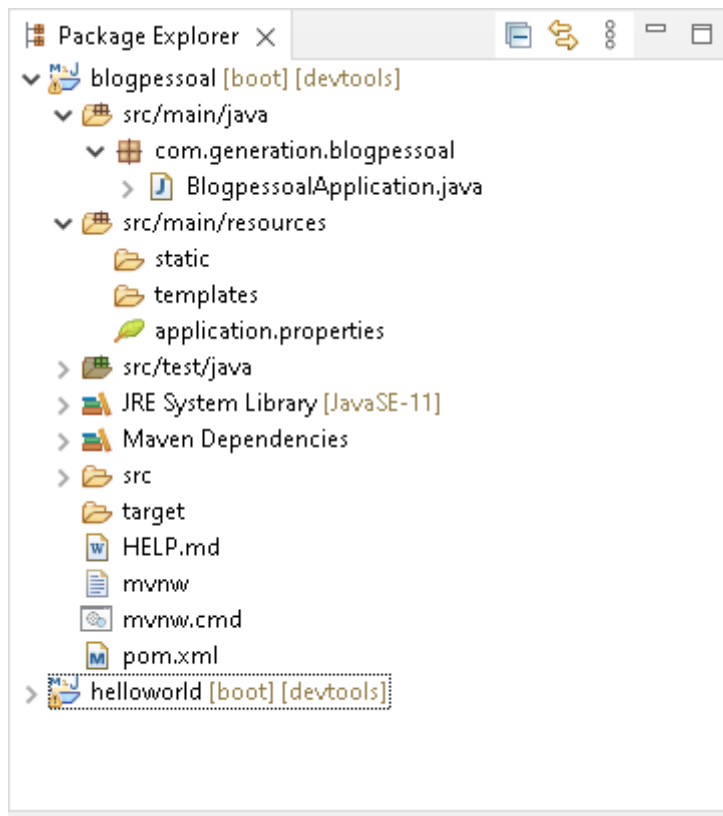
4. Na janela **Import**, clique na pasta **Maven**, na opção **Existing Maven Projects**



5. No item **Root Directory**, clique no botão **Browse** e selecione a pasta do projeto (indicado em azul na imagem abaixo).
6. No item **Projects**, selecione o arquivo **pom.xml** e clique no botão **Finish** para concluir a importação



7. A estrutura do nosso projeto importado para o STS ficará de acordo com a imagem abaixo:



Passo 04 - O arquivo pom.xml

O POM (**Project Object Model** - "projeto modelo do objeto"), é um arquivo **XML** (eXtensible Markup Language), que dentro do contexto de um projeto Maven, é o arquivo mais importante do projeto. O POM guarda todas as informações básicas de um projeto Maven, bem como as diretivas de como o artefato (resultado) final deste projeto deve ser construído.

Dentro do POM colocamos as informações das Bibliotecas (Dependências), que o nosso projeto necessita para funcionar e o Maven se encarrega de efetuar o download e inserir no **Build Path** (Caminho para as Bibliotecas utilizadas pelo compilador Java para construir o projeto final) e **Classpath** (Caminho para os pacotes utilizados no projeto durante o desenvolvimento das Classes, ou seja, as instruções Import).

O Maven irá buscar por essas Dependências em locais chamados **Repositórios**. Existem basicamente dois repositórios, o repositório local que está localizado na pasta **.m2/repository** e o repositório remoto **Maven Repository**.



ATENÇÃO: A pasta **.m2** geralmente está localizada no seu computador em: **C:\Users\seu usuario\.m2\repository**

Ao adicionar uma nova dependência no projeto, o Maven primeiro realiza a busca em nosso repositório local, caso não encontre irá buscar no repositório remoto e então fazer o download da biblioteca e disponibiliza-la no repositório local. Dessa forma, caso você necessite utilizar a mesma biblioteca em outro projeto não será necessário realizar o download novamente.

4.1. Estrutura básica do arquivo pom.xml

Vamos analisar o arquivo pom.xml gerado em nosso Projeto Blog Pessoal:

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
3   xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">
4   <modelVersion>4.0.0</modelVersion>
```


A primeira parte informa a versão do XML e a versão do POM (4.0)

```
5    <parent>
6      <groupId>org.springframework.boot</groupId>
7      <artifactId>spring-boot-starter-parent</artifactId>
8      <version>2.6.2</version>
9      <relativePath /> <!-- lookup parent from repository -->
10   </parent>
```

A segunda parte informa que o Spring Boot será o Projeto Pai (parent) do Projeto Blog Pessoal (child). Esta relação é semelhante ao conceito de Herança da Programação Orientada à Objetos (POO), onde o projeto Spring Blog Pessoal herdará todas as características de um Projeto Spring Boot. A versão do Spring Boot poderá ser diferente.

```
11   <groupId>com.generation</groupId>
12   <artifactId>blogpessoal</artifactId>
13   <version>0.0.1-SNAPSHOT</version>
14   <name>blogpessoal</name>
15   <description>Projeto Blog Pessoal</description>
```

A terceira parte informa os dados que identificam o projeto (groupid, artifactId e version), que foram informados no Spring Initializr durante a construção do projeto.

```
16   <properties>
17     <java.version>11</java.version>
18   </properties>
```

A quarta parte informa a versão do Java. Neste item é possível configurar outras propriedades além da versão.

```
19   <dependencies>
20     <dependency>
21       <groupId>org.springframework.boot</groupId>
22       <artifactId>spring-boot-starter-data-jpa</artifactId>
23     </dependency>
24     <dependency>
25       <groupId>org.springframework.boot</groupId>
26       <artifactId>spring-boot-starter-validation</artifactId>
27     </dependency>
28     <dependency>
29       <groupId>org.springframework.boot</groupId>
30       <artifactId>spring-boot-starter-web</artifactId>
31     </dependency>
32     <dependency>
33       <groupId>org.springframework.boot</groupId>
34       <artifactId>spring-boot-devtools</artifactId>
35       <scope>runtime</scope>
36       <optional>true</optional>
37     </dependency>
38     <dependency>
39       <groupId>mysql</groupId>
40       <artifactId>mysql-connector-java</artifactId>
41       <scope>runtime</scope>
42     </dependency>
43     <dependency>
44       <groupId>org.springframework.boot</groupId>
45       <artifactId>spring-boot-starter-test</artifactId>
46       <scope>test</scope>
47     </dependency>
48   </dependencies>
```

A quinta parte informa lista todas as Dependências que o projeto. A lista de Dependências foi criada no Spring Initializr durante a construção do projeto. Durante a implementação do projeto iremos inserir outras Dependências nesta parte de código.

```
50     <build>
51         <plugins>
52             <plugin>
53                 <groupId>org.springframework.boot</groupId>
54                 <artifactId>spring-boot-maven-plugin</artifactId>
55             </plugin>
56         </plugins>
57     </build>
```

Na sexta e ultima parte temos as informações de build que dizem como o projeto deve ser compilado pelo Maven. Nessa parte também estão definidos quais plugins do Maven o nosso projeto necessita para ser compilado.



[Documentação: pom.xml](#)



Passo 05 - Configurar a Conexão com o Banco de dados

Diferente do Projeto Hello World, no Projeto Blog Pessoal vamos utilizar um Banco de dados para persistir os nossos Objetos, ou seja, gravar dados nas Tabelas.

Antes de iniciarmos o processo de Desenvolvimento do código das nossa Classes, precisamos configurar o o acesso ao nosso Banco de dados, caso contrário ao executar o projeto receberemos a mensagem de erro abaixo no Console, informando que o não foi configurado o acesso ao Banco de dados:

```
*****
APPLICATION FAILED TO START
*****

Description:

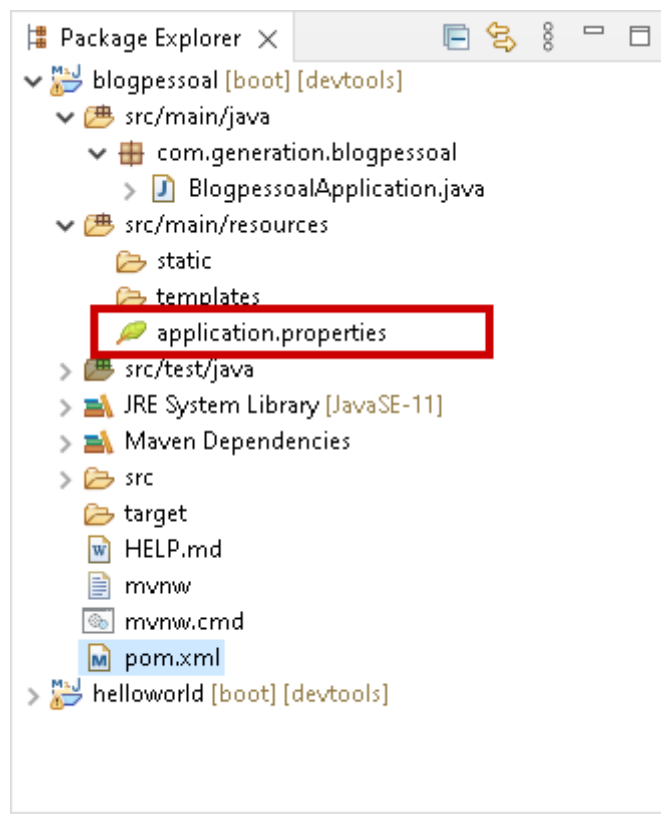
Failed to configure a DataSource: 'url' attribute is not specified and no embedded datasource could be configured.

Reason: Failed to determine a suitable driver class

Action:

Consider the following:
  If you want an embedded database (H2, HSQL or Derby), please put it on the classpath.
  If you have database settings to be loaded from a particular profile you may need to activate it (no profiles are currently active).
```

A configuração do Banco de dados será implementada no arquivo **application.properties**, localizado na Source Folder **src/main/resources**, como mostra a figura abaixo:



Os **Arquivos de Propriedades** (properties) são usados para manter diversas propriedades do projeto em um único arquivo para executar o aplicativo em um ambiente diferente. Neste arquivo você pode configurar além do Banco de dados o Spring Mail para enviar e-mails pela aplicação, Configurações do Servidor, entre outras.

 [Documentação: application.properties](#)



ALERTA DE BSM: Mantenha a Atenção aos Detalhes ao inserir as configurações do Banco de dados no arquivo *application.properties*, especialmente na linha 3, que possui muitas variações entre letras minúsculas e maiúsculas.

Para configurar o nosso Banco de dados MySQL, vamos Inserir as linhas abaixo no arquivo **application.properties**.

```
1 spring.jpa.hibernate.ddl-auto=update
2 spring.jpa.database=mysql
3 spring.datasource.url=jdbc:mysql://localhost/db_blogpessoal?
  createDatabaseIfNotExist=true&serverTimezone=America/Sao_Paulo&useSSL=false
4 spring.datasource.username=root
5 spring.datasource.password=root
6
7 spring.jpa.show-sql=true
8
9 spring.jpa.properties.hibernate.dialect = org.hibernate.dialect.MySQL8Dialect
10
11 spring.jackson.date-format=yyyy-MM-dd HH:mm:ss
12 spring.jackson.time-zone=Brazil/East
```



[Código fonte do arquivo application.properties](#)

Item	Descrição
spring.jpa.hibernate.ddl-auto	<p>Define como o JPA irá inicializar o Banco de dados.</p> <p>update ⇒ O modelo de objeto criado com base nas Anotações na Classe Model é comparado com o esquema existente, e então o Hibernate atualiza o esquema de acordo com as diferenças. Ele nunca exclui as tabelas ou colunas, mesmo que não sejam mais utilizadas. Nesta opção os dados persistidos não são apagados.</p>
spring.jpa.database	Define o SGBD que será utilizado (MySQL)
spring.datasource.url	<p>Define os dados da conexão com o Banco de dados:</p> <p>jdbc:mysql://localhost/db_blogpessoal ⇒ endereço (jdbc:mysql://localhost/) + nome do Banco (db_blogpessoal)</p> <p>?createDatabaseIfNotExist=true ⇒ criar automaticamente o Banco de dados no MySQL caso ele não exista (true)</p> <p>&serverTimezone=America/Sao_Paulo ⇒ define o fuso horario do servidor MySQL (America/Sao_Paulo)</p> <p>&useSSL=false ⇒ desabilita a camada de segurança da conexão com o MySQL (SSL)</p>
spring.datasource.username	Define o usuário do MySQL (root)
spring.datasource.password	Define a senha do usuário do MySQL (root)
spring.jpa.show-sql	Exibe todas as Queries SQL no Console do STS durante a execução da aplicação
spring.jpa.properties.hibernate.dialect	Configura a versão do MySQL, em nosso projeto estamos utilizando a versão 8 (org.hibernate.dialect.MySQL8Dialect).
spring.jackson.date-format	Configura o formato da Data (yyyy-MM-dd) e da Hora (HH:mm:ss) da aplicação
spring.jackson.time-zone	Configura o fuso horario do servidor da aplicação (Brazil/East)

Observações importantes:

- O Nome do banco de dados deve seguir o padrão **db_nome-do-banco**. O prefixo **db** indica que se trata de um Database (Banco de dados). O nome do banco é recomendado que seja **o mesmo do projeto** (blogpessoal), em **letras minúsculas, sem espaços em branco ou caracteres especiais e acentos**. Para separar as palavras em um nome composto, utilize o **_** (underline). **Exemplo:** db_blog_pessoal.
- O endereço **localhost** é o endereço local, ou seja, o seu próprio computador. Quando a aplicação estiver na nuvem, o endereço do Banco de dados será alterado para um endereço remoto.
- Para fins de aprendizagem, estamos utilizando no SGBD MySQL o usuário **root**. Vale ressaltar que no mercado de trabalho, uma aplicação em produção, jamais utilizará o usuário root, por se tratar do usuário administrador do SGBD, que tem plenos poderes sobre o Servidor. Em geral, o DBA (Database Administrator), cria um usuário apenas com os direitos de acesso que a aplicação irá precisar.
- Durante o Desenvolvimento da aplicação, você pode manter a opção: **spring.jpa.show-sql** habilitada (true). Em produção ela deve ser desabilitada.
- A configuração do **fuso-horário** no **Servidor de Banco de dados** e no **Servidor da aplicação** são itens essenciais para evitar que data e hora incorretas sejam persistidas no Banco de dados, especialmente em atributos do tipo Timestamp, que obtém a data e a hora do Servidor.



ATENÇÃO: Caso a senha do seu MySQL não seja root, atualize a linha: `spring.datasource.password` inserindo a senha que você cadastrou no MySQL no momento da instalação.



DICA: Caso você tenha esquecido a senha do seu usuário root do MySQL, consulte o Guia de desinstalação do MySQL e siga as instruções.



[Código fonte do projeto](#)

Anexo I - Modos de inicialização do Banco de dados

Forma	Descrição
update	O modelo de objeto criado com base nos mapeamentos (Anotações na Classe Model), é comparado com o esquema existente, e então o Hibernate atualiza o esquema de acordo com as diferenças. Ele nunca exclui as tabelas ou colunas existentes, mesmo que não sejam mais exigidas pelo aplicativo. Nesta opção os dados persistidos não são apagados.
create	O Hibernate primeiro elimina todas as tabelas existentes no Banco de dados e então cria novas tabelas. Com esta opção todos os seus dados serão perdidos a cada inicialização do projeto.
create-drop	Semelhante ao create, com a adição de que o Hibernate irá descartar o banco de dados depois que todas as operações forem concluídas. Esta opção é normalmente utilizada para testes de unidade.
validate	Verifica apenas se a estrutura do Banco de dados corresponde às entidades definidas na Classe Model . Se o esquema não corresponder, a inicialização do aplicativo lançará um erro (Exception).
none	Desativa a inicialização do Banco de dados.

Hibernate

O **Hibernate** é um framework para o mapeamento objeto-relacional escrito na linguagem Java, cujo objetivo é diminuir a complexidade entre os programas Java, baseado no modelo orientado a objeto, que precisam trabalhar com um banco de dados do modelo relacional (presente na maioria dos SGBDs). Em especial, no desenvolvimento de consultas e atualizações dos dados.



[Site Oficial: Hibernate](https://hibernate.org/)