Fazendo uma tela de cadastro e login com spring e java swing

Como relembrar é viver, vamos acessar o site do spring initializr

https://start.spring.io/

1. Em Project deixe a opção de Maven Project marcada;
2. A linguagem utilizada será java mesmo;
3. versão do Spring Boot será 2.1.8;
4. Em ***project metadata***, apague o ‘example’ e dê o nome que desejar, eu usarei: tutorial;
5. Em ***artifact*** apague o demo e dê o nome que desejar, eu usarei: spring;
6. O resto deixe apenas como está;
7. Chegou a hora de escolhermos nossas dependências, as que utilizaresmos serão as seguintes:
   7.1. Spring Data JPA;
   7.2. Spring Data JDBC;
   7.3. MySQL Driver;
   7.4. Spring Boot DevTools;
8. Com nossas dependências escolhidas, clique em ***generate the project***;
9. Descompacte o arquivo na área de trabalho;
10. Abra o NetBeans e utilize o atalho CTRL + SHIFT + O;

11. Selecione o arquivo que você acabou de descompactar, e o NetBeans irá abrir o projeto que você gerou;
12. Nas pastas do projeto acesse este caminho ***Other sources - src/main/resources - <default package> - application.properties***
13. Seu arquivo está vazio, então digite os seguintes comandos:

```
 spring.jpa.hibernate.ddl-auto=update

spring.datasource.url=jdbc:mysql://localhost/tutorialLoginCad?useUnicode=true&useJDBCCompliantTimezoneShift=true&useLegacyDatetimeCode=false&serverTimezone=UTC

spring.datasource.username=

spring.datasource.password=

spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQL5Dialect

spring.main.allow-bean-definition-overriding=true

spring.jmx.enabled= false
```

14. Todos estes códigos já foram explicado naquele primeiro documento, com excessão dos dois últimos, mas estes explico mais para frente, que ficará mais fácil para vocês entenderem;

> *Obs: Note que o nome da database é: **tutorialLoginCad**, então vocês devem criar uma com este mesmo nome no MySQL WorkBench de vocês;*
> *Notem tambem que tanto o username e password estão vazios, isto porque vocês devem acessar com o usuário e senha de vocês.*

Terminado isto, as propriedades do nosso projeto estão finalizadas, então iremos para o próximo passo.

Criar uma classe Java que irá servir de tabela para o nosso BD. Nela teremos apenas 4 campos, são eles:

1. ID;
2. Nome de usuário;
3. Login;
4. Senha.

Para fazermos isso, você deve seguir este caminho do seu projeto: ***Source Packages - com.<nome atribuido por você>***. Neste caminho já existe uma classe java, mas não mexeremos nela por enquanto. Então

clique com o botão direito na **com...**, selecione New/Folder e dê o nome desta pasta de tabelas.
Será gerado um arquivo com o nome com.<algumacoisa>.tabelas.

1. Neste arquivo, clique com o botão direito e selecione New/Java Class;
2. Dê o nome desta classe de tbUsuario;
3. Acima da classe tbUsuario, coloque estes comandos:

```
@Table(name = "TBUSUARIO")
@SequenceGenerator(name = "idUsuario", sequenceName = "aut
oUser", allocationSize = 1)
@Entity
```

> *Relembrando... @Table não é obrigatório, mas usaremos para definir o nome da tabela com tbusuario. @SequenceGenerator a regra de auto_increment para aquela tabela. Onde o campo **name** seria o nome da **sequência** reconhecida pelo java, **sequenceName** seria o nome da sequência reconhecida pelo banco de dados e, por fim, **allocationSize** seria de quanto em quanto será o auto_increment, no caso de 1 em 1. @Entity é o mais importante e obrigatório quando criamos uma classe que será uma tabela no banco de dados, ela define que aquela classe será uma tabela no banco.*

Após isso, dentro da nossa classe, temos que criar os campos que serão as colunas da nossa tabela:

```
@Id
@GeneratedValue(generator = "idUsuario", strategy = Genera
tionType.SEQUENCE)
private Long idUser;

@Column(name = "nmUser")
private String nmUser;

@Column(name = "loginUser", length = 15)
private String login;

@Column(name = "senhaUser", length = 12)
private String senha;
```

> *Relembrando mais uma vez...*
> *Definimos qual campo será nossa chave primária com o @Id e o @GeneratedValue atribuimos a regra de auto_increment que criamos acima, na classe. Quando falamos de campos de chave primária, nosso atributo sempre será do tipo Long.*
> *o @Column definimos uma coluna nova para a tabela e o name (não é obrigatório) é como aquela coluna irá aparecer na tabela. Todos nossos atributos, neste caso, serão do tipo String.*
> *Uma coisa nova que apareceu foi esse **lenght**, que como o próprio nome já diz, é o tamanho, no caso o tamanho da coluna, ou seja, quantos caracteres ela aceita.*

Com isso, ainda dentro da nossa classe Java, utilize Alt + Insert e

seleciona **Getter and Setter...** / Flag todos os atributos irão aparecer, ou seja, todos os que acabamos de criar.

Feito isso, aperte o playzinho verde do NetBeans e seleciona a classe principal, assim que o programa terminar de compilar, vá até seu MySQL e verifique no seu BD se a tabela que acabamos de criar está lá.

O certo era que a tabela estivesse sido criada apenas com estes passos, mas caso tenha dado algum erro, me contate.

Com nossa tabela criada, entraremos em um conceito novo, que se chama repositório ou DAO

O que seria este Repositório ou DAO? Ele vem do JPA e é basicamente uma extensão das nossas classes de tabela, ou seja, toda classe que é uma tabela, ganhará uma extensão DAO. Com ele poderemos utilizar algumas funções padrões de BD ou criar nossas próprias.
Certo, e como faremos isso?

1. Ainda na pasta Tabelas que você criou, clique com o botão direito, vá em New -> Java Interface.
2. Dê o nome como TbUsuarioDAO;
3. O arquivo inicialmente estará assim:

```
public interface tbUsuarioDAO{
```

```
}
```

4. Você deve deixá-lo assim:

```
@Repository
public interface tbUsuarioDAO extends JpaRepository<tbUsua
rio, Long>{
}
```

5. Pronto, nosso repositório está criado, por enquanto não vamos alterar em nada aqui, mais para frente vocês poderão entender melhor porque deste repositório.

Chegou a hora de criarmos nossas telas swing, então...

1. Clique com o botão direito em Source Package, New -> Folder...
2. Dê o nome da pasta de telas.
3. Com este arquivo criado, clique com o botão direito no mesmo e selecione New -> JFrame Form;
4. Dê o nome do arquivo de Tela Cadastro;
5. Com a tela gerada, personalize-a da sua maneira, tendo obrigatoriamente 3 Text Field. Para os campos de nome de usuário, login e senha. Além de um botão, para confirmar.

> *Lembre-se de renomear suas TextFields e seu botão.*

6. Ao terminar o design da sua tela, vá até a classe principal do seu projeto, que fica em **com.<nome do proejto>** e abra a .Java lá;
7. Sua classe está assim:

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

Essa unica linha de comando, é o que inicia todo projeto em Spring, mas não inicia nossa tela Swing, então precisamos adionar um comando. Um que diz ao Spring para adionar um novo componente que ele não conhece (nossa tela) ao contexto dele, e depois pedimos para ele abir esta tela. O código deve ficar assim:

```
public static void main(String[] args) {
        // Estamos instânciando um ApplicationContext, par
a fazer com que o Spring comece a 'monitorar' nossa tela
        ApplicationContext context = new SpringApplication
Builder
        (Application.class).headless(false).run(args);

            //Instanciamos nossa tela, pegamos a variavel
que criamos acima e colocamos nossa tela dentro do Context
 do spring. Em seguida, pedimos para mostrar a tela.
            TelaCadastro telaInicial = context.getBean(Tel
aCadastro.class);
            telaInicial.setVisible(true);
```

```
    }
```

Voltando para quando estavamos dentro do Application.properties. Tinhamos dois comandos novos:

1. `spring.main.allow-bean-definition-overriding=true`
2. `spring.jmx.enabled= false`
   Os dois irão servir para limpar a context toda vez que instanciarmos, utilizamos isso porque nosso projeto irá ter mais de uma tela, então toda vez que iremos pedir para abrir para o Spring abrir uma, será adionado um Context, que pode ser repetido, e isso fará com que dê erro, por já existir aquele Context. Esses comandos limpam o context e depois adicionam de novo.

Antes de rodarmos nosso projeto, vá até sua tela de Swing, clique em Source, suba todo código e em cime de `public class <nomeclasse> extends java.swing.JFrame`, adicione um @Component, essa annotation dirá ao Spring que nosso JFrame é um componente que ele deve começar a 'monitorar'.

Feito isso, você pode rodar seu projeto clicando no playerzinho verde do NetBeans e sua tela deve aparecer.

Iniciando o Cadastro

Dentro da classe da nossa tela, você deve adicionar o seguinte código:

```
@Autowired
private TbUsuarioDAO tbUsuarioDAO;
```

Com este comando estamos referenciando o repósitorio que criamos antes, para que não precisamos ficar instanciando-o várias vezes. Agora, podemos começar com os comandos para fazer a inserção no BD.

1. Vá até a tela Swing e clique duas vezes no botão de confirmar.
2. No ActionPerformed insira este código: `TbUsuario tbUsuario = new TbUsuario();`
3. Basicamente, instânciamos nossa classe da tabela de usuário para começarmos a utilizar seus métodos.
4. Teremos que pegar os textos digitados pelo usuário e armazenar em cada atributo da tbUsuario. Fazendo assim:

```
TbUsuario tbUsuario = new TbUsuario();

tbUsuario.setNmUser(txtNome.getText());
tbUsuario.setLogin(txtLogin.getText());
tbUsuario.setSenha(txtSenha.getText());
```

> *Com a classe que acabamos de instancias, estamos inserindo dados em seus atributos, no caso, dados que o usuário digitou;*
> *Obs: Lembre-se que o nome das suas TextFields não são os mesmos da minha.*

5. Ao fazer isso, precisamos fazer o insert no banco de dados, e

para isso é bem simles, basta adicionar este comando depois do setSenha: `tbUsuarioDAO.save(tbUsuario)`;

> *Com este comando estamos dizendo ao nosso repositório que ele deve pegar nossa classe instânciada e seus atributos, que foram modificados acima, e depois salvar, ou seja, realizar um insert dentro do Banco de dados.*
> *Depois de feito isto, rode seu programa, insira alguma coisa nos textFields e depois pressione o botão, logo em seguida, entre no seu MySQL Workbench e veja na sua tabela se o que você digitou foi inserido.*

Com isso, a parte de cadastro está pronto, mas você ainda pode personalizar ela com muito, fazendo condições por exemplo, para caso os campos estejam vazios. Use a criatividade.

Iniciando o Login

1. Para iniciarmos o login, primeiro você deve fazer outra tela em swing chamada TelaLogin, dentro da sua pasta Telas. Neste tela devem conter os campos login e senha, mais o botão de entrar;
2. Ao criar sua tela, repita o mesmo passo da ultima, e crie o campo @Autowired e o @Component;
3. Agora, entre na sua classe TbUsuarioDAO;
4. Como foi dito anteriormente, os repositórios do JPA tem suas funções próprias, que realizam alguns comandos do banco de dados, mas nele também temos a opção de criar nossos próprios comandos e armazená-los em uma função.

5. Veja como:

```
    @Query("Select u from TbUsuario u where u.login = ?1 an
d u.senha = ?2")
    TbUsuario findByLogin(String login, String senha);
```

6. Utilizamos a annotation @Query para criarmos consultadas personalizadas dentro do repositório, note que os os nomes não estão completamente iguais ao que estão nas tabelas. Quando criamos nossas consultas no repositório, utilizamos o nome da classe que recebrá a consulta e o nome definidos nos seus atributos;
7. Logo abaixo definimos o tipo de retorno de dados, no caso irá retornar em atributos da Classe tbUsuario, que é a nossa clase. A função também recebe parâmetros do tipo String, que serão o login e senha digitados pelo usuário.
8. No nosso select, onde tem a atributição dos valores, existe ?1 e ?2. Isso quer dizer que essa consulta irá receber algum dado, seja do usuário ou do próprio sistema. E deve ser feito em ordem númerica, ou seja, se fossem 3 campos para realizar a consulta, seriam ?1, ?2 e ?3.
9. Voltando a nossa tela swing, clique duas vezes no botão Entrar/Confirmar, para acessar sua função/metédo.
10. Adicione estes comandos dentro da função:

```
TbUsuario resultado = tbUsuarioDAO.findByLogin(txtUsuario.
getText(), txtSenha.getText());
```

```
        if(resultado == null){
            lblRetorno.setText("Login ou senha incorretos")
;
        } else{
            lblRetorno.setText("Acesso permitido");
        }
```

11. Explicando o código... Nós criamos uma váriavel do tipo da classe de usuário, que recebe a função que acabamos de criar. repare que essa função recebe dois parâmetros, que seriam os campos que o usuário digitou.
12. Logo após isso, existe uma condição verificando se a variavel que acabamos de criar está nula, se estiver, quer dizer que não foram encontrados dados, o que significa, que não existe registro com aquele login e senha. Caso contrário, caso não esteja nula, isso significa que existe um campo na tabela com aquele usuário e senha. Mostramos o resultado dentro de uma Label.

Antes de iniciar sua aplicação, volte para a tela de cadastro e clique duas vezes no botão de Cadastrar, e no final da função, adicione este código:

```
        ApplicationContext context = new SpringApplication
Builder(Application.class).headless(false).run();

        TelaLogin telaInicial = context.getBean(TelaLogin.
class);
```

```
        telaInicial.setVisible(true);
```

É o mesmo código utilizado na Main do nosso projeto, com a diferença que irá abrir a tela de Login assim que o usuário se cadastrar. Então, agora rode o código e teste se deu tudo certo.

Bom, este é o básico para fazermos um cadastro e login. Qualquer dúvida, por favor perguntem, espero que tenham entendido bem.

Estarei a disposição.