Um pouco sobre Spring

Mas antes, vamos relembrar...

Acessando este link:

https://start.spring.io/

Podemos criar nosso projeto e adicionarmos as dependências que quisermos, mas, quais dependências estamos utilizando?

1. Spring Boot DevTools
   Que nada mais é, que uma dependência que nos possibilita um melhor desenvolvimento, nos oferecendo um restart na aplicação mais rápido e em tempo real.

2. Spring Web
   Seria nosso facilitador para o desenvolvimento web, ele trabalha com RESTful(GET e POST), é nele que faremos nossas requisições do site.

3. Apache Freemarker
   Uma lib do Java que nos auxilia junto com o HTML, nele teremos

a possibilidade de utilizar algumas ferramentas que o próprio HTML não pode fazer.

4. Spring Data JPA e Spring Data JDBC
   O Spring Data JPA é um framework que irá facilitar na criação dos nossos repositórios para o Banco de Dados. Dentro dele temos algumas funções de BD e podemos criar também nossas próprias.
   Spring Data JDBC proporciona a conexão do nosso projeto com o Banco de Dados.

5. MS SQL Server Driver
   É a dependência que nos ajuda com o acesso ao SQL Sever e ao banco de dados do Azure.

6. Azure Storage
   A dependência que irá nos fornecer integração com o banco de dados dentro do Azure.

Bom, depois de criarmos o nosso projeto, o que faremos?

1. Abra o NetBeans;
2. Vá em File -> Open Project (CTRL + SHIFT + O);
3. Abra o projeto que foi gerado pelo Spring Initializr;

> Relembrando alguns conceitos...

- **Pom.xml:** Seria o nosso node.json do semestre passado, nele ficará todas as dependências do nosso projeto, podemos alterá-lo seja adicionando ou excluindo uma nova dependência. Mas sempre tendo certeza de que aquela dependência não está sendo utilizada em mais nenhum outro lugar.
- **SRC -> MAIN -> Resources**: Onde estará a nossa aplicação web e todas suas funcionalidades. Dentro da pasta ***Resources*** temos os seguintas arquivos: *static, templates e application.properties*.

1. **Static**: É a pasta que irá manter nossos arquivos para o site, exemplo: javascript ou css;
2. **Resources**: É a pasta que irá manter nossos arquivos html / páginas html;
3. **Application.properties**: O arquivo que terá as propriedades da aplicação. Por exemplo: *conexão como banco de dados e definição do tipo de arquivo HTML*.

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

### CONEXÃO COM MySQL WorkBench ###########
spring.datasource.url=jdbc:mysql://localhost/teste?useUnic
ode=true&useJDBCCompliantTimezoneShift=true&useLegacyDatet
imeCode=false&serverTimezone=UTC
spring.datasource.username=bandtec
spring.datasource.password=digitalschool
spring.jpa.properties.hibernate.dialect=org.hibernate.dial
ect.MySQL5Dialect
```

```
### CONEXÃO COM O AZURE ######
#spring.datasource.url=jdbc:sqlserver://startec-fire.datab
ase.windows.net:1433;database=BancoProjeto;user=Gf01191014
@startec-fire;password={your_password_here};encrypt=true;t
rustServerCertificate=false;hostNameInCertificate=*.databa
se.windows.net;loginTimeout=30;
#spring.datasource.username=Gf01191014
#spring.datasource.password=#Gf37762756850

spring.freemarker.template-loader-path: classpath:/templat
es
spring.freemarker.suffix: .ftl
spring.freemarker.request-context-attribute: rc
```

Como vocês podem ver, este é o código que utilizaremos no nosso Application.properties. Então, vamos por partes.

1. A primeira linha de comando nada mais é do que ao iniciar a aplicação, ira executar um comando de update no banco de dados. Isto será utilizado para caso as tabelas já estejam criadas, e por algum motivo mexemos em alguma classe java, o que poderia alterar alguma tabela ou coluna do banco, então a aplicação atualizaria isto toda vez que subisse;
2. `spring.datasource.url=jdbc:mysql://localhost/teste?...` é o comando que irá acessar o link do banco de dados. Note que depois de `localhost/` existe a palavra `teste`, que nada mais é

do que o nome do banco de dados que eu criei na minha máquina. Fiquem atentos com este nome, porque a aplicação não irá subir se for definido o nome de uma database que não existe;

3. `spring.datasource.username=bandtec` é o comando que define o usuário que está se conectando ao banco de dados. Eu defini como **bandtec** para caso algum dia não estivermos com o computador pessoal em mãos, e tivermos que utilizar o da faculdade, não vamos precisar trocar o usuário e nem a senha no código só para se conectar. Então, o usuário e a senha sempre serão os mesmos. Após estes passos, explicarei como criar um novo usuário dentro do WorkBench;
4. `spring.datasource.password=digitalschool` é o comando que define a senha para poder acessar ao banco de dados. Também será uma senha padrão;
5. `spring.jpa.properties.hibernate.dialect=org.hibernate.di` é o comando que irá definir o tipo de linguagem utilizada por aquele banco, neste caso, está definido como ***MySql***.

Antes de irmos para os próximos passos, irei mostrar como criar um novo usuário dentro do seu WorkBench

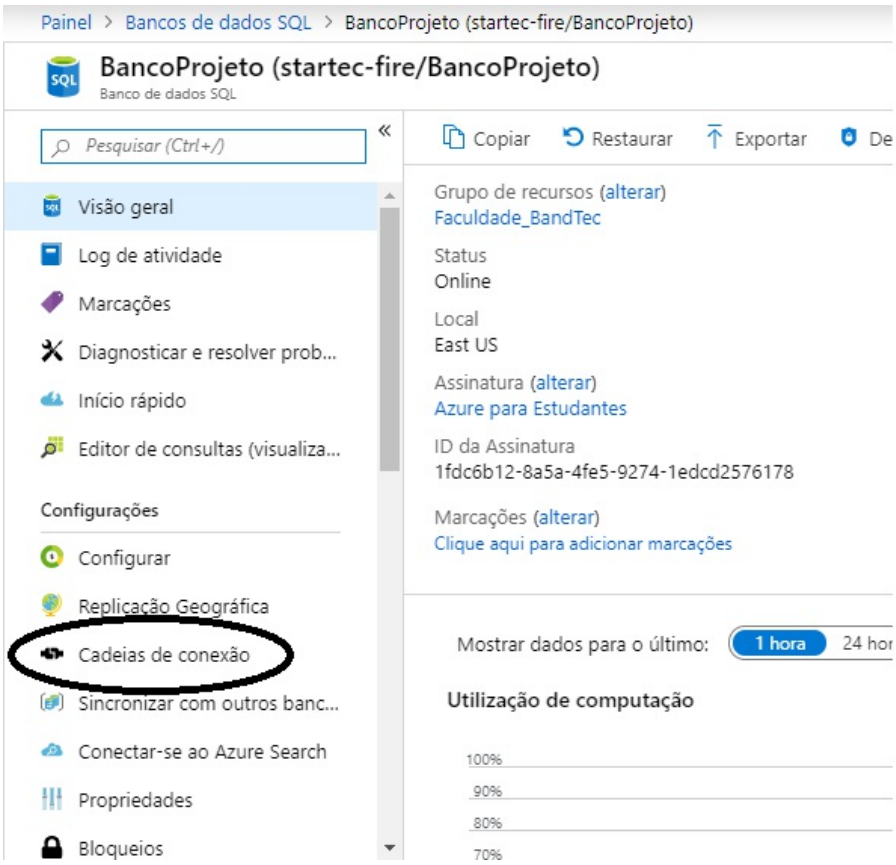
1. Abra o MySQL WorkBench;
2. Utilize estes comandos:
   `CREATE USER 'bandtec'@'localhost' IDENTIFIED BY 'digitalschool';`

`GRANT ALL PRIVILEGES ON * . * TO 'bandtec'@'localhost';`

3. O primeiro comando cria o usuário bandtec dentro do servidor localhost, e define sua senha como digitalschool;
4. O segundo comando dá todos acessos e permissões a este usuário criado.

Com isto mostrado, voltaremos a nossa programação normal

6. `spring.datasource.url=jdbc:sqlserver://start...` é o comando que acessa o banco de dados, o mesmo de cima, só que com o link de acesso ao azure. Segue abaixo como obter este link:
   1.Entre no portal Azure -> Vá até Banco de Dados SQL -> Cadeias de Conexão



2.Clique em JDBC e copie o primeiro link que irá aparecer

3.Pronto! Agora é só colocar este link dentro daquela linha de comando.

7. `spring.datasource.username=Gf01191014`
`spring.datasource.password=#Gf37762756850`
comandos onde você define usuário e senha de acesso ao banco de dados. No caso, se quiserem testar no de vocês, é só trocar pelo seu.

> *Obs: Note que os comandos do azure acima possuem sempre um '#' antes de cada um, isto é porque é o metodo de adicionar comentários do tipo de arquivo properties. Quando você quiser testar servidor local, comente os comandos para o azure. Quando você quiser testar em nuvem, comente os comandos para servidor local*

8. `spring.freemarker.template-loader-path: classpath:/templates`
   `spring.freemarker.suffix: .ftl`
   `spring.freemarker.request-context-attribute: rc`
   Não sei explicar para vocês ao certo o que cada comando desses faz, mas sei que servem para configurar o freemarker no projeto. Por exemplo, o primeiro sabe que deve ser carregado os arquivos html que estão na página templates. O segundo, ele procura apenas por arquivos na extensão .ftl, ou seja, nossos arquivos não seriam mais .html e sim .ftl, mas isto não muda os comandos de HTML é só um jeito do programa saber o que são os arquivos freemarker. O ultimo comando não sei explicar realmente, mas se descobrir para o que é, aviso vocês.

Com isto explicado, vamos para a ultima parte de tudo

Para isto, precisaramos voltar uma pasta antes de resources e entrar em todas as pastas de java até sua última.

Inicialmente, só teremos um arquivo nesta pasta, que seria <nomedoprojeto>.java. Neste arquivo é onde fica nossa classe main(principal), ela que irá rodar todas as ações do sistema, por enquanto, e se Deus quiser, não precisaremos mexer nela.

Afins de teste e explicação, iremos criar duas classes, dentro desta pasta do projeto. Uma irá se chamar TabelaTeste e a outra ApplicationController.

Começando pela TabelaTeste...

```
@Table(name = "TBTESTE")
@Entity
@SequenceGenerator(name = "autoGen", sequenceName = "seqGe
n", allocationSize = 1)
public class TabelaTeste {

    @Id
    @GeneratedValue(generator = "autoGen", strategy = Gene
rationType.SEQUENCE)
    private Long idTeste;

    @Column(name = "teste")
    private String teste;
    }
```

Bom, o que nós fizemos aqui? Na verdade é bem simples, basicamente mapeamos/criamos uma tabela para o banco de dados. Quando iniciarmos a aplicação, nosso interpretador irá ler isso, e saber que deve gerar uma tabela ou reconhecer uma tabela que está dentro do banco de dados ao qual ele está conectado. Essa é uma tabela simples, com apenas duas colunas um Id e um campo teste.

Repare que existem alguns comandos que possuem um @, isto se chama anottation, é uma ferramenta do java que iremos utilizar bastante. Neste caso, a anottation está configurada junto ao spring, por isto temos alguns nomes conhecidos, como: *@Entity, @Table @Id e @Column*. Toda classe que for mapear uma tabela do BD tem a obrigatoriedade de possuir um @Entity em cima de si, mas tanto o @Table como o @SequenceGenerator são opcionais e estou utilizando para exemplificar melhor.

1. Começando pelo @Table, aí está como finalidade de procurar ou criar uma tabela chamada “TBTESTE”

2. @SequenceGenerator, é a regra do autoincrement do banco de dados. O campo name seria o campo do nome da sequência que será reconhecido pelo Java, o campo sequenceName será o campo do nome da sequência reconhecido pelo banco de dados, e por ultimo, o campo allocationSize será de quanto em quanto o Id irá aumentar, no caso, de um em um.

3. Dentro da nossa Classe, existe o @Id, que diz que o que vier

abaixo dele, será um campo chave primária no banco de dados.

4. Logo abaixo, temos o @GeneratedValue que é a anotação que diz que o campo @Id recebrá alguma regra de auto increment, que no caso, seria o “autoGen”, como foi definido acima. O strategy, seria o tipo de geração de valores, no exemplo está como tipo de sequência, e será o que mais iremos utilizar.

5. Private Long idTeste, nada mais é do que a criação de uma variável privada (Coisas de POO, logo o Yoshi irá ensinar), do tipo Long (Iremos utilizar sempre este tipo quando um campo for chave primária) com o nome idTeste. Este campo no banco de dados irá aparecer como *id_teste*;

6. @Column, você diz ao java que o que vier abaixo dele vai ser uma nova coluna da tabela, no caso estou criando uma coluna que receberá o nome *teste* do banco de dados. E o mesmo esquema para a criação de variável, sendo a diferença de que será um campo String, ou seja, um campo de texto;

Com isto explicado, ainda dentro da sua classe TabelaTeste, clique em um campo vazio e aperte ALT + Insert.

1. Na janelinha que irá aparecer, selecione a opção de Getter e Setter;
2. E por fim, selecione todos os campos que irão aparecer, no caso idTeste e teste;

Perceba que foi gerado algumas linhas de código, estas linhas são os get e set dos atributos que acabamos de criar. É o mesmo conceito de POO que o Yoshi nos ensinou.
Com isto feito, terminamos a nossa classe TabelaTeste.

Sendo agora a ultima parte deste documento, abra a classe ApplicationController que pedi para que você criasse antes, caso não tenha criado, pode criar agora e utilize estes comandos.

```
@Controller
public class ApplicationController {



    @GetMapping("/teste")
    public String pageTeste(){
        return "teste";
    }

    @GetMapping("/")
    public String pageIndex(){
        return "index";
    }
```

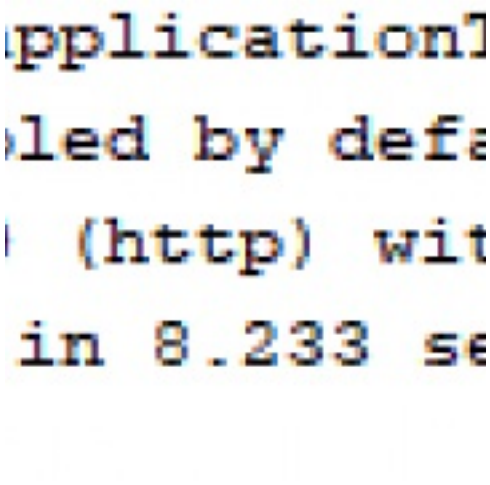
```
}
```

Tá, Matheus, o que acabamos de fazer? É bem simples, fizemos uma requisição, duas do tipo GET, isto vai ser utilizado quando você subir a aplicação.
Digamos que a url do servidor local seja:
localhost:8383
Caso você digite localhost:8383/teste
O interpretador irá cair neste primeiro GET e retonar para a página com o nome teste. Que seria na verdade, o teste.ftl.

O mesmo serve para a outra requisição, mas ela só irá ser ativada caso seja digitado na url - locahost:8383/ ou apenas localhost:8383. Isto fará com que vá para a página index do seu site.

Pode ser um pouco confuso de começo, só que mais para frente vai pegando o jeito. Então, aproveita que você está com o NetBeans ligado, e aperta no playzinho verde para iniciar a aplicação, vai aparacer uma janela pedindo para selecionar a classe principal, você só terá uma opção, pode marcá-la.
Caso não apareça essa janela, não tem problema, isso quer dizer que o NetBeans reconheceu a classe por você.

A aplicação não demora para subir, normalmente só na primeira vez, mas você irá saber que funcinou caso apareça está mensagem:



E para acessas o site digite localhost:8080 ou localhost:8080/teste

Caso não funcione, irá aparecer exit -0 ou algo assim, aí é só você me chamar que irei ajudar.

Para parar a aplicação clique no botão vermelho do console do NetBeans.

Por hoje é só, qualquer dúvida só vir falar comigo