**Aula XXVIII - Desenvolvimento Web III**

**Continuação de aplicação Spring Web - MVC**

**Configurando a tela de Cadastro**

**Testando tela de Cadastro**

**Criando domínio para manipulação do DAO**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*Fontes:*

[*https://spring.io/projects/spring-framework*](https://spring.io/projects/spring-framework)

[*https://blog.algaworks.com/spring-mvc/*](https://blog.algaworks.com/spring-mvc/)

[*https://www.devmedia.com.br/entendendo-anotacoes-em-java/26772*](https://www.devmedia.com.br/entendendo-anotacoes-em-java/26772)

[*https://www.devmedia.com.br/como-criar-anotacoes-em-java/32461*](https://www.devmedia.com.br/como-criar-anotacoes-em-java/32461)

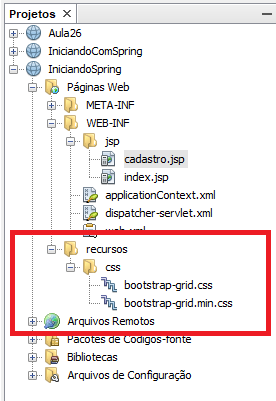
[*https://www.devmedia.com.br/conhecendo-o-mysql-front-parte-1/7021*](https://www.devmedia.com.br/conhecendo-o-mysql-front-parte-1/7021)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

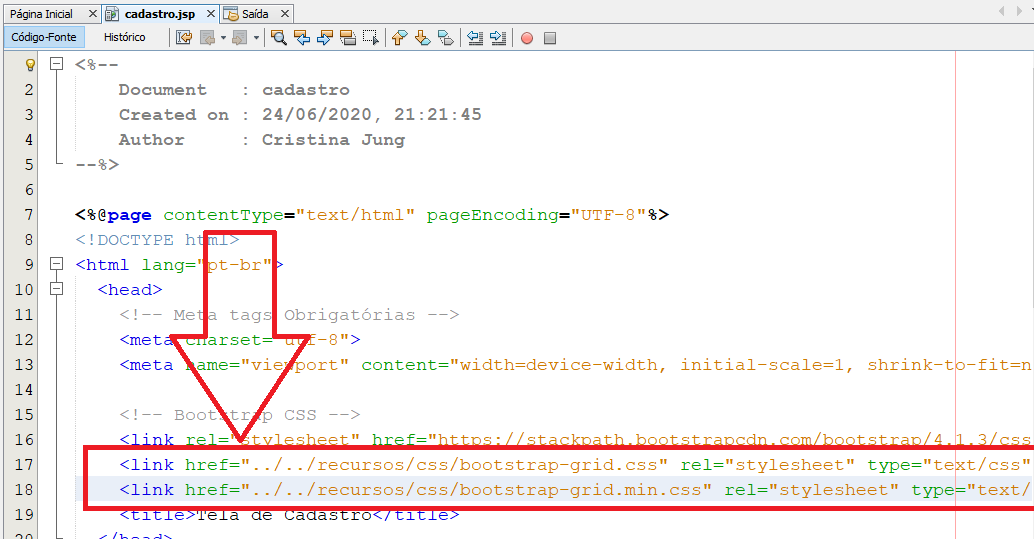
**Configurando a tela de cadastro**

Vamos começar a configurar o Framework que fará a parte do frontend da nossa API. Já sabemos que podemos trabalhar com os CDNs para o pré carregamento, ou mesmo trazer para o projeto os arquivos específicos.

Primeiro vamos configurar todos os nossos arquivos JSP com os CDNs do framework desejado. Neste tutorial eu vou usar o **Bootstrap**, então, vamos criar uma pasta na pasta **Páginas Web do nosso projeto**. Coloquei o nome de recurso e dentro dela, vou criar outra com o nome de css, dentro desta pasta, colocaremos os arquivos do Bootstrap. Para você fazer download do Bootstrap, basta clicar [neste link](https://getbootstrap.com.br/docs/4.1/getting-started/download/) que ele irá baixar no formato compactado - basta descompactar.



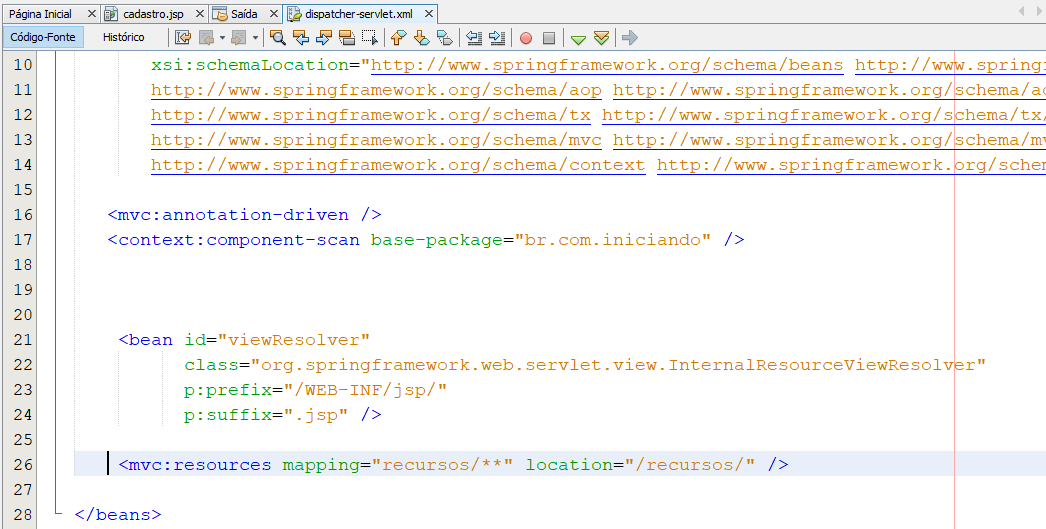
Também podemos arrastar os arquivos de CSS para o nosso JSP, mas sempre lembrando que só isso não basta!



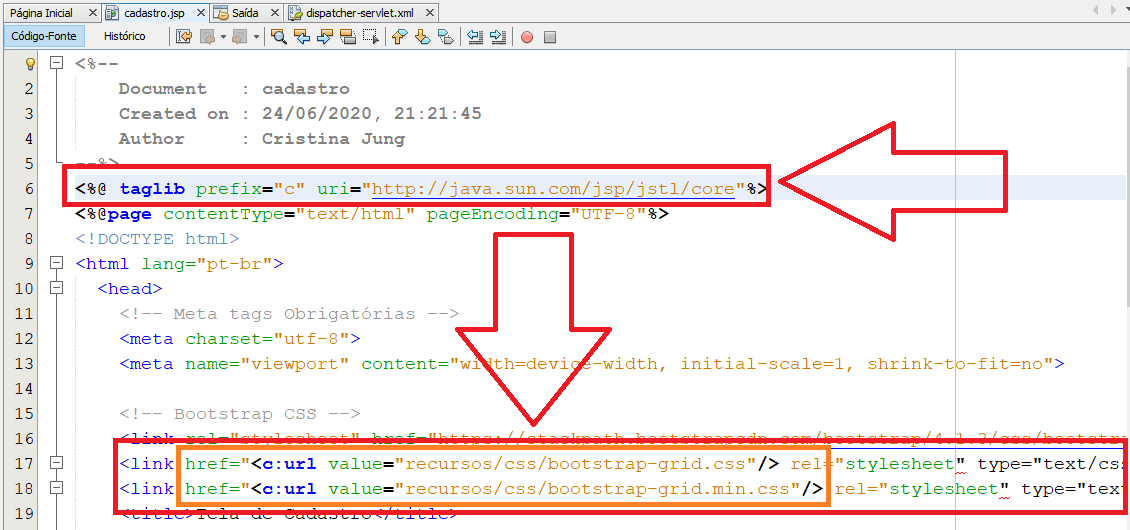
Agora é que vamos precisar da **biblioteca JSTL** para podermos setar corretamente para o Spring os arquivos de CSS do framework que estamos usando, nas suas pastas específicas!

Vamos acessar o nosso **dispacher-servlet** e vamos inserir outra notação para que o Spring localize a pasta recursos.

**<mvc:resources mapping="recursos/\*\*" location="/recursos/" />**

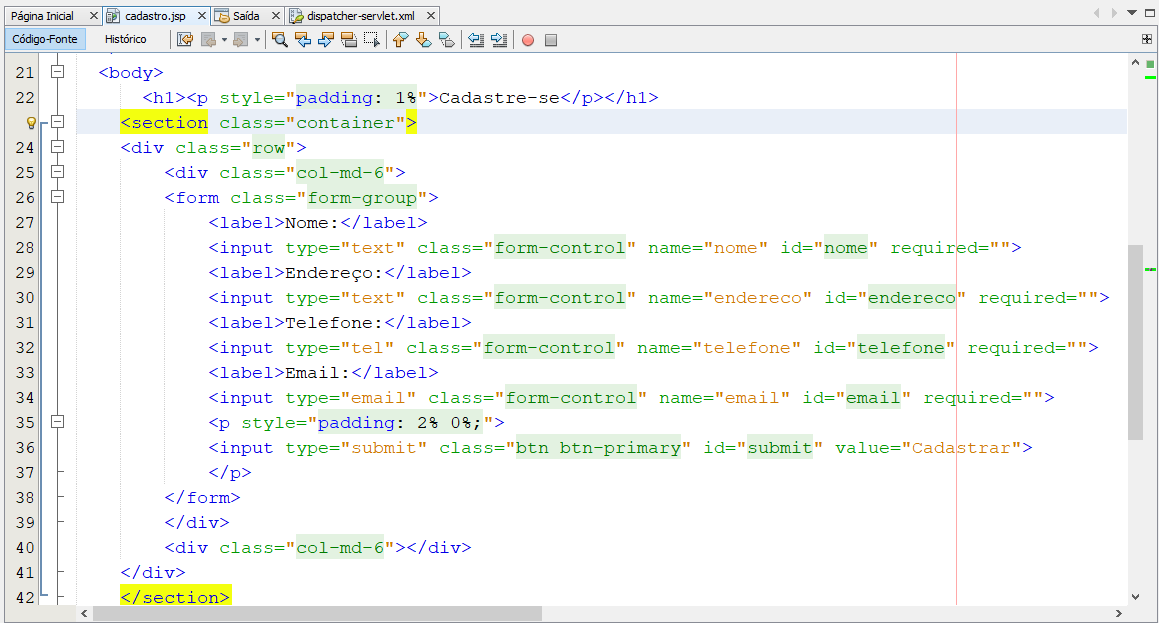
Mapeamento da nossa pasta recursos - rece os 2 **“\*”** para indicar a agregação de outras pastas que podem estar dentro de recursos.

Agora vamos ajustar o nosso cadastro.jsp! Observe o código! Iremos entrar com a tag lib(biblioteca e aqui falamos da biblioteca JSTL) e indicar o endereço correto para que encontre os nossos arquivos de CSS.

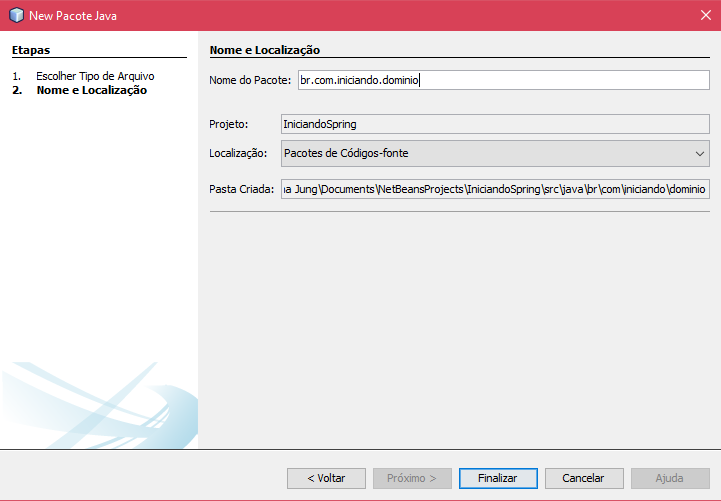


Agora vamos finalizar o nosso formulário! Observe na imagem abaixo o código do formulário dando atenção especial à formatação usando o [Sistema de Grid](https://getbootstrap.com.br/docs/4.1/layout/grid/) do Bootstrap!

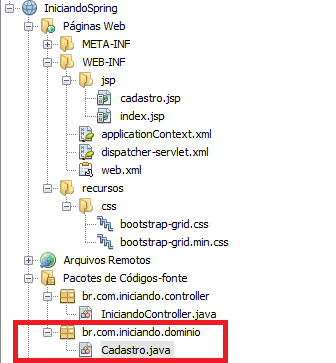
|  |
| --- |
| Lembre-se sempre!! Existem vários frameworks para frontend, eles estão listados bem como os seus links na aula XXIII. Você decide qual framework irá utilizar, explore, estude, pesquise sua documentação e veja qual framework é tendência de mercado e qual você se adapta mais! |



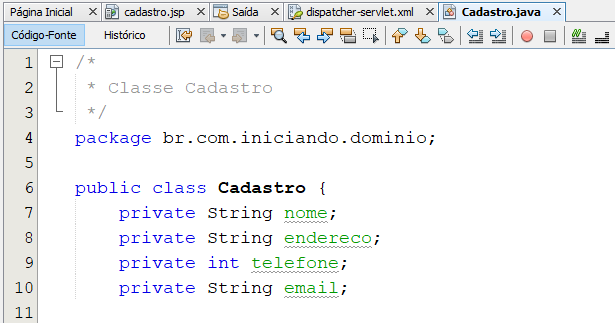
Perfeito! Agora que já montamos o nosso formulário, formatamos e testamos como ele fica no navegador, está na hora de fazermos o nosso Domínio para que possamos criar uma base de dados. Como estamos trabalhando com **padrão MVC e Orientado a Objeto**, vamos criar mais um pacote para código fonte com nome de **dominio**. Tudo que fará parte do banco irá dentro do domínio.



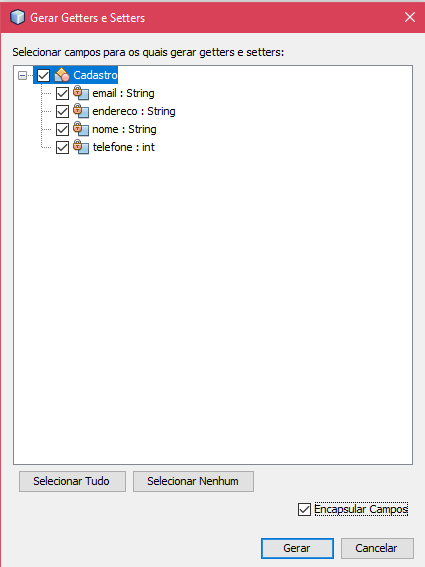
Próxima etapa, será criarmos a Classe Cadastro, dentro do pacote domínio. Clique com o botão direito do mouse sobre o pacote dominio/novo/Classe Java - nome da classe: Cadastro.



Perfeito! Agora vamos codificar a nossa classe de acordo com os campos que temos no nosso formulário.



Vamos usar o ALT + INSERT para gerar os métodos getters e setters! Observe a imagem abaixo! **Muito importante observar que os nomes dos atributos na classe Cadastro, são os mesmos nomes dos atributos nomes nas tags do nosso formulário.**



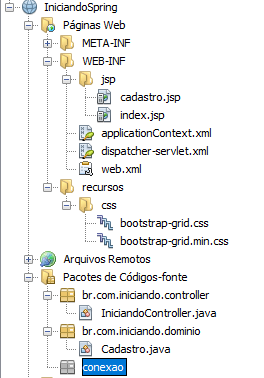
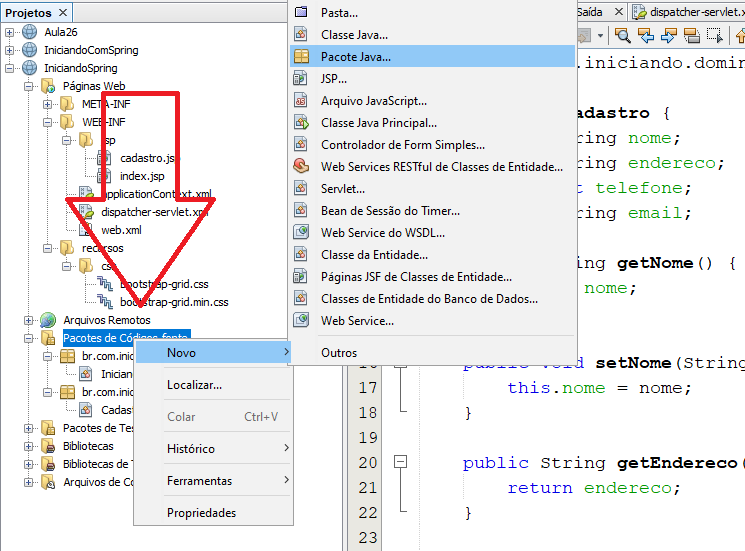
A partir de agora, iremos criar o nosso DAO!

**DAO!**

|  |
| --- |
| **Objeto de acesso a dados (acrônimo do inglês Data Access Object - DAO)**, é um padrão para aplicações que utilizam persistência de dados, onde tem a separação das regras de negócio das regras de acesso a banco de dados, **implementada com linguagens de programação orientadas a objetos (como por exemplo Java) e arquitetura MVC**, onde todas as funcionalidades de bancos de dados, tais como obter conexões, mapear objetos para tipos de dados SQL ou executar comandos SQL, devem ser feitas por classes DAO. |

Traduzindo! O DAO faz a persistência no BD para a nossa aplicação. Simplesmente quando a gente clicar em Cadastrar, lá no nosso controller, nós iremos chamar o DAO e o DAO irá colocar estas informações no nosso BD. A única funcionalidade dele é fazer a persistência com o BD e fará parte da camada Model da nossa arquitetura.

**Primeiro iremos criar o pacote “conexão”**



E vamos criar uma Classe Java com nome de Conexao. Não esqueça, a classe sempre começa com letra maiúscula



|  |
| --- |
| /\*  \* Classe de conexão.  \*/  package conexao;  import java.sql.Connection;  import java.sql.DriverManager;  import java.sql.SQLException;  import java.util.logging.Level;  public class Conexao {  //propriedade de conexão  private Connection conn;    private void conectar() {  System.out.println("Conectando ao Banco de Dados");    //comando try, vai tentar a conexão  try {  Class.forName("com.mysql.jdbc.Driver");  conn = DriverManager.getConnection("jdbc:mysql://localhost/**iniciandospring**", "root", "");  System.out.println("Conectado");  } catch (ClassNotFoundException e) {  System.out.println("Classe não encontrada, adicione os arquivos da biblioteca");  java.util.logging.Logger.getLogger(Conexao.class.getName()).log(Level.SEVERE, null, e);  } catch (SQLException e) {  System.out.println("e");  }  }    public Connection getConexao() {  conectar();  return conn;  }    } |

**Observação:** a **Classe Conexao** é uma classe padrão para estabelecer a conexão com o BD, a única coisa que muda será o nome do bando, que no caso acima está em vermelho. O **“root”** é o nome do usuário para conectar e as **“”**, definem que não configuramos a senha.