**Anotações técnicas**

**Projeto Brewer**

**- Protocolo de comunicação HTTP**

**Request:  
Verbos empregados:** GET, POST, PUT  
**Headers:** Accept, Accept-language  
**Parametros:** Query String, Payload

**Response:**

**Code:** 200 OK, 404, 500

**Headers:** Content-Type

**Resposta:** Json

**Configuração eclipse**

Versão 4.5.2 (Mars.2)

View java

View server

Editors

Spelling desabilitar

Wrap

Fonts (14)

**Instalar tomcat no eclipse**

Utilizar a view server do eclipse add

**Maven**

Apartir da versão 4.5 do Eclipse o Maven 3.2 já vem embarcado

**Server**

Tomcat versão 8.35

Instalar Drivers de na pasta lib do tomcat:

**Driver:** Banco MySql drive mysql-connector-java-5.1.39-bin.jar

**Controlador de pool de conecção:** c3p0-0.9.5.jar

mchange-commons-java-0.2.11.jar

**Plugins do eclipse:**

Jrebel : evita o redeploy em projetos java e permite o update de códigos durante a excução da aplicação

**Frameworks diferença**

**Spring MVC** – é um Action Base

Front Controler (Spring MVC) -> Conttroller -> Model

Controller -> View -> Browser

Browser 1-> DispatcherServlet 2-> Handler Mapping (qual url?)

DispatcherServlet 3-> Controller 4-> DispatcherServlet

DispatcherServlet 5-> ViewResolver (qual view acessar?)

DispatcherServlet 6-> View 7-> Browser (renderiza)

Vantagem – baixo acoplamento com a view

**JSF** – é um Component Based

Front Controller (JSF) -> View -> Managed Bean -> Model

View - > Browser

Vantagem: Components ricos controlados por códigos java

O DispatcherServlet recebe a requisição, pergunta ao Handler mapping que informa onde esta os controller (configurado no WebConfig) que possui a url solicitada, o Controller é acessado e devolve os dados e o nome da view que deve ser renderizada, o viewResolver informa o caminho, onde esta a pagina que esta sendo solicitada ao DispatcherServlet cria a view e devolve ao browser o html.

**Metodo básico para chamada da view no controler quando a url** /cervejas/novo **for digitada no browser, o html (view) CadastroCerveja.html já deve estar criado na pasta cerveja**

@RequestMapping(/cervejas/novo)

public String novo(){

return “cerveja/CadastroCerveja”;

**Pom.xml do Maven plugin 3.2**

**Dependence mananger baixa as outras dependências de um projeto maven automaticamente, sem que seja necessário informar a versão.**

**Colocar no pom.xml**

<dependencyManagement>

<dependencies>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-framework-bom</artifactId>

<version>${spring-framework.version}</version>

<type>pom</type>

<scope>import</scope>

</dependency>

</dependencies>

</dependencyManagement>

POR EXEMPLO ELE VAI BAIXAR O PROJETO SPRING ABAIXO E TODAS AS OUTRAS DEPENDENCIAS DELE OBSERVE QUE NÃO É NECESSARIO COLOCAR A TAG <VERSION> :

<!-- Spring MVC -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-webmvc</artifactId>

<scope>compile</scope>

<exclusions>

<exclusion>

<groupId>commons-logging</groupId>

<artifactId>commons-logging</artifactId>

</exclusion>

</exclusions>

</dependency>

**A dependencia abaixo vai providencia as classes necessárias a as configuração do** [**DispatcherServlet**](eclipse-javadoc:%E2%98%82=brewer/C:%5C/Users%5C/Udinei%5C/.m2%5C/repository%5C/org%5C/springframework%5C/spring-webmvc%5C/4.3.0.RELEASE%5C/spring-webmvc-4.3.0.RELEASE.jar%3Corg.springframework.web.servlet.support(AbstractAnnotationConfigDispatcherServletInitializer.class%E2%98%83AbstractAnnotationConfigDispatcherServletInitializer%E2%98%82org.springframework.web.servlet.DispatcherServlet) **do spring**

<!-- Servlet API -->

<dependency>

<groupId>javax.servlet</groupId>

<artifactId>javax.servlet-api</artifactId>

<version>${servlet.version}</version>

<scope>provided</scope>

</dependency>

A tag abaixo informa ao maven que somente em tempo de projeto que será utilizado a dependência, o mesmo não deve ser empacotar quando for compilada para produção, pois a mesma sera fornecida pelo servidor

<scope>provided</scope>

Ja a tag abaixo informa que sempre vamos precisar dessa dependência inclusive no empacotamento.

<scope>compile</scope>

**Notas Controller:**

O método padrão para o controller é o método GET se não informado nada no atributo method=”” da tag <form> do html.

Ex:

<form></form> - será enviado uma chamada ao método GET

<form method=”POST”> </form> - será enviado uma chamada ao método POST

**Notas Thymeleaf:**

O thymelead vincula os atributos do objeto java a tags no html, que pode ser recuperado em métodos no controller, posso também receber os parâmetros sem ser objetos, strings por exemplo. Atributo name=”sku”

Pegando no controler novo(String sku){}

Ex:

**Html:**

<form method="POST">

<label for="sku">SKU</label>

<input type="text" id="sku" name="sku">

<br>

<input type="submit" value="Salvar">

</form>

O atributo name da tag html sera enviado para o controller (esse mapeamento é feito automaticamente pelo spring, pode ser passado um objeto também, que deve ter um atributo com o mesmo nome, do atributo na view html)

O pode ser acessado da seguinte no controller da forma abaixo:

**string**

public String cadastrar(String sku) {

**objeto**

public String cadastrar(Cerveja cerveja) {

cerveja.getSku();

**Controller:**

@RequestMapping(value = "/cervejas/novo", method = RequestMethod.POST)

public String cadastrar(Cerveja cerveja) {

System.out.println(">>> sku: " + cerveja.getSku());

return "cerveja/CadastroCerveja";

}

Caso não seja informado a view em return ele tentara retornar um html usando a url /cerveja/novo.html.

O objeto deve ter um construtor default, o spring cria um automaticamente.

**Classe java:**

public class Cerveja {

private String sku;

private String nome;

**Validando classe Venda com SpringValidator**

(springframework.validation.ValidationUtils)

@Component

**public** **class** VendaValidator **implements** Validator {

/\*\* Declara suporte de validação para a classe Venda \*/

@Override

**public** **boolean** supports(Class<?> clazz) {

**return** Venda.**class**.isAssignableFrom(clazz);

}

/\*\* target vira como objeto da classe que esta sendo validada \*/

@Override

**public** **void** validate(Object target, Errors errors) {

ValidationUtils.*rejectIfEmpty*(errors,"cliente.codigo", "", "Selecione um cliente na pesquisa rápida");

// convertendo target para Venda

Venda venda = (Venda) target;

validarSeInformouApenasHorarioEntrega(errors, venda);

validarSeInformouItens(errors, venda);

**if**(venda.getValorTotal().compareTo(BigDecimal.***ZERO***) < 0){

errors.reject("", "Valor total não pode ser negativo");

}

}

**private** **void** validarSeInformouItens(Errors errors, Venda venda) {

**if**(venda.getItens().isEmpty()){

errors.reject("", "Adicione pelo menos uma cerveja na venda");

}

}

**private** **void** validarSeInformouApenasHorarioEntrega(Errors errors, Venda venda) {

**if**(venda.getHorarioEntrega() != **null** && venda.getDataEntrega() == **null**){

errors.rejectValue("dataEntrega", "", "Informe uma data de entrega para um horário");

}

}

}

**Validação de campos da tela com Bean Validator**

**Na entidade**

@NotBlank // anotação do bean validation implementada pelo hibernate validator

private String nome;

**No controller** deve ser informado como parâmetro a anotação @**Valid** do beanValidation antes do nome da entidade, e o parâmetro **BindingResult** será o objeto que retornara o erro ocorrido na View. **Ex:**

@RequestMapping(value = "/cervejas/novo", method = RequestMethod.POST)

public String cadastrar(@Valid Cerveja cerveja, BindingResult result) {

if (result.hasErrors()) {

System.out.println(">>>> tem erro sim!");

}

System.out.println(">>> sku: " + cerveja.getSku());

return "redirect:/cervejas/novo";

}

**Retornando mensagem para a view em caso de erro**

Para tanto no método do Controller deve ser passado o parametro Model ex:

@RequestMapping(value = "/cervejas/novo", method = RequestMethod.POST)

public String cadastrar(@Valid Cerveja cerveja, BindingResult result, Model model, RedirectAttributes attributes) {

if (result.hasErrors()) {

System.out.println(">>>> tem erro sim!");

Model.addAttribute(“mensagem”, “Erro no formulario”); // msg com

return “cerveja/CadastroCerveja”; // forward, que exibe

diretamente a pagina sem

fazer uma nova requisicao

}

// redirect faz uma nova requisição para uma url e envia a mensagem nela

// com attributes.addFlashAttribute que cria uma sessão temporaria

attributes.addFlashAttribute("mensagem", "Cerveja salva com sucesso!");

System.out.println(">>> sku: " + cerveja.getSku());

return "redirect:/cervejas/novo";

}

**E no formulário implementado o namespace do thymeleaf**

xmls:th=”http://www.thymeleaf.org”

e a variável dentro do form

<span th:text=”${mensagem}”></span>

**Validação regra de negocio – Evitando entidades repetidas no banco, validando por atributo e capturando exception no controller**

**No controller – Ao Salvar**

@PostMapping({ "/novo", "{\\+d}" })

**public** ModelAndView salvar(@Valid Estilo estilo, BindingResult result, RedirectAttributes attributes){

**if**(result.hasErrors()){

**return** novo(estilo);

}

**try**{

cadastroEstiloService.salvar(estilo);

} **catch** (NomeEstiloJaCadastradoException e){

// atributo que deu problema

result.rejectValue("nome", e.getMessage(), e.getMessage());

**return** novo(estilo);

}

attributes.addFlashAttribute("mensagem", "Estilo salvo com sucesso!");

**return** **new** ModelAndView("redirect:/estilos/novo");

}

**No Service**

@Autowired

**private** Estilos estilos;

// Salva ou altera o objeto informado

@Transactional // controle de transação manual, necessario para evitar que o spring faça begin end nas consultas tambem

**public** Estilo salvar(Estilo estilo){

Optional<Estilo> estiloExistente = estilos.findByNomeIgnoreCase(estilo.getNome());

// objeto com o atributo pesquisado existe, e seu identificador é igual ao do objeto a salvar, entao sobe excecao

**if**(estiloExistente.isPresent() && !estiloExistente.get().equals(estilo)){

**throw** **new** NomeEstiloJaCadastradoException("Nome do estilo já cadastrado");

}

// salva um novo objeto, ou altera caso identificador do objeto ja existir

**return** estilos.saveAndFlush(estilo);

}

**No repository**

@Repository

**public** **interface** Estilos **extends** JpaRepository<Estilo, Long>, EstilosQueries {

**public** Optional<Estilo> findByNomeIgnoreCase(String nome);

}

**Na Classe Exception**

**public** **class** NomeEstiloJaCadastradoException **extends** RuntimeException {

**private** **static** **final** **long** ***serialVersionUID*** = 1L;

**public** NomeEstiloJaCadastradoException(String message){

**super**(message);

}

}

**Cadastro rápido utilizando View Modal com Json e javascript**

**Alert controlado por javascript**

**No html**

<div class=*"alert alert-danger hidden js-mensagem-cadastro-rapido-estilo"* role=*"alert"*> </div>

**No javascript**

// capturando div com a classe

**this**.containerMensagemErro = $('.js-mensagem-cadastro-rapido-estilo');

// adicionando escondendo a div com o alert

**this**.containerMensagemErro.addClass('hidden');

// removendo fazendo aparecer a div com o alert

**this**.containerMensagemErro.removeClass('hidden');

**Salvando via Ajax formato Json com javascript**

**Adicionar no maven**

var Brewer = Brewer || {};

Brewer.EstiloCadastroRapido = (function(){

function EstiloCadastroRapido(){

this.botaoSalvar = this.modal.find('.js-modal-cadastro-estilo-salvar-

btn');

}

EstiloCadastroRapido.prototype.iniciar = function (){

this.botaoSalvar.on('click', onBotaoSalvarClick.bind(this));

}

function onBotaoSalvarClick(){

var nomeEstilo = this.inputNomeEstilo.val().trim();

$.ajax({

url: this.url,

method:'POST',

contentType: 'application/json',

data: JSON.stringify({ nome: nomeEstilo }),

error: onErroSalvandoEstilo.bind(this),

success: onEstiloSalvo.bind(this)

});}

**Pesquisa via ajax utilizando javascript**

No controller para receber um parâmetro na url utilizar **@RequestParam** nos argumentos da função, um pra cada parâmetro como no exemplo abaixo. Vira a url **/brewer/cidades?estado=2** caso não seja fornecido parâmetro o valor default sera o valor -1, e para retornar como Json o método deve informar a anotation **@ResponseBody** junto ao tipo de retorno**.**

**No controller**

@RequestMapping(consumes = MediaType.***APPLICATION\_JSON\_VALUE***)

@Cacheable(value = "cidades", key="#codigoEstado") // cache de cidades, chave codigo do estado

**public** @ResponseBody List<Cidade> pesquisarPorCodigoEstado(

@RequestParam(name = "estado", defaultValue = "-1") Long codigoEstado){

**try** {

Thread.*sleep*(1000); // teste - aguarda 1 segundo antes de pesquisar

} **catch** (Exception e) {

// **TODO**: handle exception

}

**return** cidades.findByEstadoCodigo(codigoEstado);

}

**No Html –** data:url=”@{/cidades}” sera capturado na função javascript e usado para informar ao ajax a url do método a ser executado no controller

<div class=*"form-group col-sm-4"*>

<label for=*"cidade"*>Cidade</label>

<div class=*"bw-field-action"*>

<input type=*"hidden"* id=*"inputHiddenCidadeSelecionada"* th:value=*"\*{endereco.cidade} !=*

*null ? \*{endereco.cidade.codigo}"*/>

<select id=*"cidade"* class=*"form-control"* data:url=*"@{/cidades}"* th:field=*"\*{endereco.cidade}"* >

</select>

<div class=*"bw-field-action\_\_icon js-img-loading"* style="display: *none*">

<img th:src=*"@{/images/mini-loading.gif}"* >

</div>

</div>

</div>

**No Javascript**

**function** inicializarCidades(codigoEstado){

**if** (codigoEstado){

**var** resposta = $.ajax({

url: **this**.combo.data('url'),

method: 'GET',

contentType: 'application/json',

data: { 'estado': codigoEstado},

beforeSend: iniciarRequisicao.bind(**this**),

complete: finalizarRequisicao.bind(**this**)

});

resposta.done(onBuscarCidadesFinalizado.bind(**this**));

}**else** {

reset.call(**this**);

}

}

**Colocar um ícone de animação ao carregar objetos**

**No html**

<div class=*"bw-field-action"*>

… colocar aqui conteudo a ser apresentado junto com a animação...

<div class=*"bw-field-action\_\_icon js-img-loading"* style="display: *none*">

<img th:src=*"@{/images/mini-loading.gif}"* >

</div>

</div>

**No css**

/\* Field Action \*/

*.bw-field-action* {

display: *flex*;

}

*.bw-field-action\_\_icon* {

margin: *6px* *0* *0* *5px*

}

**Relacionamento ManyToMany**

Somente nas classes que se cria o atributo de relacionamento, nas tabelas não **Ex:** Em Usuário cria-se o atributo **grupos** que é a **lista** de grupos que usuario pertence, e em grupos cria-se o atributo **permissões** que é a lista de permissões dos usuários. Mas na tabela usuário não será criado o campo grupo e nem na tabela grupo o campo permissão, somente nas classes esses campos serão criados, para serem persistidos em uma tabela 1 para muitos que sera a tabela de relacionamento.

****

**SQL Criação das tabelas - relacionamento ManyToMany**

A tabela **usuario\_grupo** e **grupo\_permissão** usam como **primary key** chaves compostas pelos ids dos campos da tabela usuarioi, grupo e permissao, os ids das duas tabelas relacionandas.

**CREATE** **TABLE** usuario (

codigo BIGINT(20) **PRIMARY** **KEY** AUTO\_INCREMENT,

nome **VARCHAR**(50) **NOT** **NULL**,

email **VARCHAR**(50) **NOT** **NULL**,

senha **VARCHAR**(120) **NOT** **NULL**,

ativo **BOOLEAN** **DEFAULT** **true**,

data\_nascimento **DATE**

) ENGINE=InnoDB **DEFAULT** CHARSET=utf8;

**CREATE** **TABLE** grupo (

codigo BIGINT(20) **PRIMARY** **KEY**,

nome **VARCHAR**(50) **NOT** **NULL**

) ENGINE=InnoDB **DEFAULT** CHARSET=utf8;

**CREATE** **TABLE** permissao (

codigo BIGINT(20) **PRIMARY** **KEY**,

nome **VARCHAR**(50) **NOT** **NULL**

) ENGINE=InnoDB **DEFAULT** CHARSET=utf8;

**Criando tabelas ManyToMany**

**CREATE** **TABLE** usuario\_grupo (

codigo\_usuario BIGINT(20) **NOT** **NULL**,

codigo\_grupo BIGINT(20) **NOT** **NULL**,

**PRIMARY** **KEY** (codigo\_usuario, codigo\_grupo),

**FOREIGN** **KEY** (codigo\_usuario) **REFERENCES** usuario(codigo),

**FOREIGN** **KEY** (codigo\_grupo) **REFERENCES** grupo(codigo)

) ENGINE=InnoDB **DEFAULT** CHARSET=utf8;

**CREATE** **TABLE** grupo\_permissao (

codigo\_grupo BIGINT(20) **NOT** **NULL**,

codigo\_permissao BIGINT(20) **NOT** **NULL**,

**PRIMARY** **KEY** (codigo\_grupo, codigo\_permissao),

**FOREIGN** **KEY** (codigo\_grupo) **REFERENCES** grupo(codigo),

**FOREIGN** **KEY** (codigo\_permissao) **REFERENCES** permissao(codigo)

) ENGINE=InnoDB **DEFAULT** CHARSET=utf8;

**Na Entidade principal “classe Usuario”**

Relacionamento ManyToMany entre **Usuario e Grupo** serão persistidos na tabela **usuário\_grupo**

@Size(min=1, message = "Selecione pelo menos um grupo")

@ManyToMany // para este tipo de relacionamento, a melhor pratica é cria um metodo no UsuarioQueries e inicializar quando precisar

@JoinTable(name = "usuario\_grupo", joinColumns = @JoinColumn(name = "codigo\_usuario")

, inverseJoinColumns = @JoinColumn(name = "codigo\_grupo"))

**private** List<Grupo> grupos;

**Obs:** Não é necessário colocar **cascade** para salvar também grupos, porque o grupo já esta salvo no banco de dados, o objetivo aqui é salvar somente os ids do usuário e do grupo na tabela de relacionamento.

**Senha criptografando**

Adicionar no pom.xml Sspring Security

Na edição o usuário não é obrigado a informa a senha, somente quando for um usuário novo (exibe asterisco no label)

Criar classe **SecurityConfig.java** e criar nela o método abaixo

@Bean

**public** PasswordEncoder passwordEncoder(){

**return****new** BCryptPasswordEncoder();

}

**Adicionar a classe SecurityConfig.java na classe AppInitializer.java ex:**

@Override

**protected** Class<?>[] getRootConfigClasses() {

**return** **new** Class<?>[] {JPAConfig.**class**, ServiceConfig.**class**, SecurityConfig.**class**,

S3Config.**class**} ;

}

**No html –** para exibir asterisco no label somente ser for novo

<div class=*"form-group col-sm-3"* brewer:classforerror=*"senha"* th:classappend=*"${usuario.novo} ? bw-required"*>

**Na Entidade**

**public** **boolean** isNovo(){

**return** codigo == **null**;

}

**No Service**

@Autowired

**private** PasswordEncoder passwordEncoder;

**método salvar**

// Nota: O "campo senha" na edição sempre vira em branco na tela (Seguranca do Spring Security),

// Caso seja novo Usuario, ou Ja existente que tenha alterado, informado uma nova senha

// entao criptografa a nova senha informada e seta no cadastro do usuario

**if**(usuario.isNovo() || !StringUtils.*isEmpty*(usuario.getSenha())){

// criptogrando senha para novos usuarios

usuario.setSenha(**this**.passwordEncoder.encode(usuario.getSenha()));

// Usuario já cadastrado, não alterou a senha

// seta a mesma senha do usuario recuperada do banco (usuarioExistente), ja criptografada

}**else** **if**(StringUtils.*isEmpty*(usuario.getSenha())) {

usuario.setSenha(usuarioExistente.get().getSenha());

}

**Na classe relacionada “Grupo”**

**Relacionamento entre Grupo e Permissoes**

@ManyToMany

@JoinTable(name = "grupo\_permissao", joinColumns = @JoinColumn(name = "codigo\_grupo")

, inverseJoinColumns = @JoinColumn(name = "codigo\_permissao"))

**private** List<Permissao> permissoes;

**Usando observer para capturar exception e exibir Mensagens de Erro**

Ver classe:

ControllerAdviceExceptionHandler

**Métodos de Chamadas Callbacks do JPA**

@PrePersist

@PreUdate

@Postload

@Preremove

Etc...

**Na entidade**

Ex:

// transformando texto digitado no campo em maiusculo ao salvar ou // alterar

@PrePersist @PreUpdate

**private** **void** prePersisteUpdate(){

sku = sku.toUpperCase();

}

**Padrões javascript empregados – Autor Addy osmany**

* [JavaScript Namespacing Patterns](https://addyosmani.com/resources/essentialjsdesignpatterns/book/index.html#detailnamespacing)
* [Module Pattern](https://addyosmani.com/resources/essentialjsdesignpatterns/book/index.html#modulepatternjavascript)

**Extendo thymeleaf criando novo atributo html usando Dialect e Processadores do thymeleaf**

**Ver doc thymealeaf:**

**Processadores** são os que implementam os atributos ou tags: ex: th:each th:if etc..

**Dialect** é um nomespace com conjunto de processadores.

Ex: de dialects layout, docorate, brewer

xmlns:layout=*"http://www.ultraq.net.nz/thymeleaf/layout"*

layout:decorate=*"~{layout/LayoutPadrao}"*

xmlns:brewer=*"http://brewer.algaworks.co*

**criando um dialect**

**é necessário que o processador seja todo em minúsculo**

**CRIANDO CLASSE PROCESSOR**

public class ClassForErrorAttributeTagProcessor extends AbstractAttributeTagProcessor{

private static final String NOME\_ATRIBUTO = "classforerror";

private static final int PRECEDENCIA = 100;

public ClassForErrorAttributeTagProcessor(String dialectPrefix) {

super(TemplateMode.HTML, dialectPrefix, null, false, NOME\_ATRIBUTO, true, PRECEDENCIA, true);

}

@Override

protected void doProcess(ITemplateContext context, IProcessableElementTag tag, AttributeName attributeName,

String attributeValue , IElementTagStructureHandler structureHandler) {

boolean temErro = FieldUtils.hasErrors(context, attributeValue);

if(temErro){

String classesExistentes = tag.getAttributeValue("class");

structureHandler.setAttribute("class", classesExistentes + " has-error");

}

}

}

**CRIANDO CLASSE de DIALECT BREWERDIALECT**

**public class BrewerDialect extends AbstractProcessorDialect{**

**public BrewerDialect() {**

**super("AlgaWorks Brewer", "brewer", StandardDialect.PROCESSOR\_PRECEDENCE);**

**}**

**@Override**

**public Set<IProcessor> getProcessors(String dialectPrefix) {**

**final Set<IProcessor> processadores = new HashSet<>();**

**processadores.add(new ClassForErrorAttributeTagProcessor(dialectPrefix));**

**processadores.add(new MessageElementTagProcessor(dialectPrefix));**

**return processadores;**

**}**

**}**

**ADICIONAR NO WEBCONFIG NO METODO templateEngine(){...}**

@Bean

**public** TemplateEngine templateEngine(){

SpringTemplateEngine engine = **new** SpringTemplateEngine();

engine.setEnableSpringELCompiler(**true**);

engine.setTemplateResolver(templateResolver());

engine.addDialect(**new** LayoutDialect()); //para definição do layoutpadrao

engine.addDialect(**new** BrewerDialect());

engine.addDialect(**new** DataAttributeDialect());

engine.addDialect(**new** SpringSecurityDialect());

**return** engine;

}

**Extendendo o thymeleaf - Criando uma nova tag html**

**public** **class** MessageElementTagProcessor **extends** AbstractElementTagProcessor{

**private** **static** **final** String ***NOME\_TAG*** = "message";

**private** **static** **final** **int** ***PRECEDENCIA*** = 100;

**public** MessageElementTagProcessor(String dialectPrefix) {

**super**(TemplateMode.***HTML***, dialectPrefix, ***NOME\_TAG***, **true**, **null**, **false**,

***PRECEDENCIA***);

}

@Override

**protected** **void** doProcess(ITemplateContext context, IProcessableElementTag tag

,IElementTagStructureHandler structureHandler) {

IModelFactory modelFactory = context.getModelFactory();

IModel model = modelFactory.createModel();

model.add(modelFactory.createStandaloneElementTag("th:block", "th:replace", "fragments/MensagemSucesso :: mensagemSucesso"));

model.add(modelFactory.createStandaloneElementTag("th:block", "th:replace", "fragments/MensagensErroValidacao :: mensagensErroValidacao"));

// true para o thymeleaf processar false se fosse um html

structureHandler.replaceWith(model, **true**);

}

}

**ADICIONAR o processor criado acima na classe:**

**BrewerDialect.class**

**Mensagem de Erro e de Sucesso customizadas**

Faz uso de dois fragments (arquivos) html:

**Sucesso: MensagemSucesso.html**

<!DOCTYPE html>

<html lang="pt"

xmlns="http://www.w3.org/1999/xhtml"

xmlns:th="http://www.thymeleaf.org">

<div class="alert alert-success alert-dismissible" role="alert"

th:if="${not #strings.isEmpty(mensagem)}" th:fragment="mensagemSucesso">

<button type="button" class="close" data-dismiss="alert">

<span aria-hidden="true">&times;</span>

</button>

<span th:text="${mensagem}"></span>

</div>

</html>

**Erro: MensagensErroValidacao.html**

<!DOCTYPE html>

<html lang="pt"

xmlns="http://www.w3.org/1999/xhtml"

xmlns:th="http://www.thymeleaf.org">

<div class="alert alert-danger alert-dismissible" role="alert" th:if="${#fields.hasAnyErrors()}" th:fragment="mensagensErroValidacao">

<button type="button" class="close" data-dismiss="alert">

<span aria-hidden="true">&times;</span>

</button>

<th:block th:each="detailedError : ${#fields.detailedErrors()}">

<div><i class="fa fa-exclamation-circle"></i> [[${detailedError.message}]]</div>

</th:block>

</div>

</html>

**Redirect e Forward no controller**

**forward** - é comportamento default do spring, carrega a pagina e retorna código 200 Ok. não faz uma nova requisição e utiliza o objeto Model passado como parâmetro para enviar uma mensagem para mesma tela.

**redirect** - retorna um código 302 para o browser forçando-o a fazer uma nova requisição, para a url informada no Location do response. Pode ser informada qualquer nova url

**Acentuação no eclipse**

Opções > encodi (busca) > workspace

**Campo:** Text file encoding

Default(UTF-8) SELECIONAR

**Principais Tags do thymeleaf**

**${NomeVariavel}** **– usado para exibir o conteudo da variavel definida no controller**

**th:object="${cerveja} – usado action form para informar qual objeto sera processado**

**th:field="\*{sku}" - usado para exibir atributos do objeto informado na action form, faz o bind do objeto informado no form com o campos da tela**

**th:text="${mensagem} – usado para exibir no text do input o valor de uma variavel definida no controller**

**fields.detailedErrors – retorna true se tiver erros no objeto ou no formulário retorna uma lista**

**th:each="erro : lista" – percorre uma lista e exibe o conteúdo ex:**

**<div th:each="erro : ${#fields.detailedErrors('${cerveja.\*}')}">**

**<span th:text="${erro.message}"></span>**

**</div>**

Exibindo erros com thymeleaf

Atributo: **${#fields.detailedErrors('${cerveja.\*}')}** – tem erros no atributo informado no caso cerveja

**No html:**

**<div th:each="erro : ${#fields.detailedErrors('${cerveja.\*}')}">**

**<span th:text="${erro.message}"></span>**

**</div>**

**<form method="POST" th:object="${cerveja}">**

**<span th:text="${mensagem}">Erro!</span>**

**<label for="sku">SKU</label>**

**<input type="text" id="sku" th:field="\*{sku}">**

**....**

**</form>**

**No controller:**

**@Controller**

**public class CervejasController {**

**@RequestMapping("/cervejas/novo")**

**public String novo(Cerveja cerveja) {**

**return "cerveja/CadastroCerveja";**

**}**

**@RequestMapping(value = "/cervejas/novo", method = RequestMethod.POST)**

**public String cadastrar(@Valid Cerveja cerveja, BindingResult result, Model model, RedirectAttributes attributes) {**

**if (result.hasErrors()) {**

**return novo(cerveja);**

**}**

**// Salvar no banco de dados...**

**attributes.addFlashAttribute("mensagem", "Cerveja salva com sucesso!");**

**System.out.println(">>> sku: " + cerveja.getSku());**

**return "redirect:/cervejas/novo";**

**}**

**}**

**Na entidade:**

**@NotBlank(message = "A descrição é obrigatoria")**

**@Size(max = 50, message = "O tamanho da descrição deve estar entre 1 e 50")**

**private String descricao;**

**@NotNull(message = "Valor é Obrigatorio")**

**@DecimalMin("0.01")**

**@DecimalMax(value = "9999999.99", message = "O valor da cerveja deve ser menor que R$9.999.999,99")**

**private BigDecimal valor;**

**Pagina de template**

**Instalando o template de aplicação da Algaworks que já utiliza o bootstrap e vem semipronta e é grátis**

**1.Baixar e copiar para :**

**> src/main/resources**

**> static**

**> layout**

**2. Informar o spring onde procurar os recursos staticos da aplicação como imagens, javascript, sytylesheets etc ..**

**Configurar no WebConfig.java o metodo**

**@Override**

**public void addResourceHandlers(ResourceHandlerRegistry registry) {**

**registry.addResourceHandler("/\*\*").addResourceLocations("classpath:/static/");**

**}**

**3.alterar nas paginas html do template os caminhos para css, javascripts e imagens usando o nome da aplicação seguido do caminho no passo 1**

**Principais classes do bootstrap**

**Elementos de telas para entrada, saída, exibição**

**Logging ou log da aplicacao**

SLF4J – é facade para outras api de log, que facilita a troca de api de logs se for o caso. O spring já vem com o commons-logging, mas não é recomendado. Na aplicação esta usando o log4j com SLF4J.

Configuração – colocar dependencia no maven pom.xml

Aquivo de configuração – log4j2.xml

Nível

Loggers

**Exemplo de como fazer log de uma classe.**

**private** **static** **final** Logger ***logger*** = LoggerFactory.*getLogger*(FotoStorageLocal.**class**);

**Nos métodos:**

@Override

**public** **void** excluir(String foto) {

**try** {

Files.*deleteIfExists*(**this**.local.resolve(foto));

Files.*deleteIfExists*(**this**.local.resolve(***THUMBNAIL\_PREFIX*** + foto));

} **catch** (IOException e) {

***logger***.warn(String.*format*("Error apagando foto '%s'. Mensagem: %s", foto, e.getMessage()));

}

}

**private** **void** criarPastas() {

**try** {

Files.*createDirectories*(**this**.local);

**if**(***logger***.isDebugEnabled()){

***logger***.debug("Pastas criadas para salvar fotos");

***logger***.debug("Pastas default: "+ **this**.local.toAbsolutePath());

}

}**catch** (IOException e) {

**throw** **new** RuntimeException("Erro criando para para salvar foto", e);

}

}

**Criando Templates dinâmicos com javascript – handlebar**

Em http://handlebarsjs.com/ baixar javascript - handlebars.min.js

**No arquivo que sera o template Html**

<script id=*"foto-cerveja"* type=*"text/x-handlebars-template"*>

<div class="row js-foto-cerveja">

<div class="col-md-4"></div>

<div class="col-md-4 text-center">

<a href="#" class="thumbnail">

<img src="{{url}}"></img>

</a>

<button type="button" class="btn btn-link js-remove-foto"><span class="glyphicon glyphicon-remove"></span></button>

</div>

<div class="col-md-4"></div>

</div>

</script>

**No html que vai usar o template handlebar**

<script th:src=*"@{/javascripts/vendors/handlebars.min.js}"*></script>

<th:block th:replace=*"hbs/FotoCerveja"*></th:block>

**No javaScript**

**var** Brewer = Brewer || {};

Brewer.UploadFoto = (**function**(){

**function** UploadFoto(){ // funcao construtora inicializacao de variaveis

**this**.inputNomeFoto = $('input[name=foto]'); // acessando elementos hidden do html

**this**.inputContentType = $('input[name=contentType]');

**this**.novaFoto = $('input[name=novaFoto]');

**this**.inputUrlFoto = $('input[name=urlFoto]');

**this**.htmlFotoCervejaTemplate = $('#foto-cerveja').html(); // obtendo html template com codigo do handlebars

**this**.template = Handlebars.compile(**this**.htmlFotoCervejaTemplate); // compilando template handlebar

**this**.containerFotoCerveja = $('.js-container-foto-cerveja');

**this**.uploadDrop = $('#upload-drop');

}

UploadFoto.prototype.iniciar = **function** (){ // funcao da execucao do comportamento

**var** settings = {

type: 'json',

filelimit: 1, // quantidade de arquivos que sera enviados de uma vez

allow: '\*.(jpg|jpeg|png)', // tipos de arquivos que esta sendo enviado contentType

action: **this**.containerFotoCerveja.data('url-fotos'), // action recebe a url para onde sera enviada a requisicao

complete: onUploadCompleto.bind(**this**), // o atributo complete, recebera o retorno que vira do servidor

beforeSend: adicionarCsrfToken // antes de executar o envio do ajax

}

UIkit.uploadSelect($('#upload-select'), settings);

UIkit.uploadDrop(**this**.uploadDrop, settings);

**if**(**this**.inputNomeFoto.val()){

renderizarFoto.call(**this**, {

nome: **this**.inputNomeFoto.val(),

contentType: **this**.inputContentType.val(),

url: **this**.inputUrlFoto.val()});

}

}

**function** onUploadCompleto(resposta){ // funcoes auxiliares

**this**.novaFoto.val('true');

**this**.inputUrlFoto.val(resposta.url);

renderizarFoto.call(**this**,resposta);

}

**function** renderizarFoto(resposta){

**this**.inputNomeFoto.val(resposta.nome)

**this**.inputContentType.val(resposta.contentType);

**this**.uploadDrop.addClass('hidden'); // esconde a div upload-drop que contem o link de selecao da foto

**var** htmlFotoCerveja = **this**.template({url: resposta.url}); // passando valor para variavel {{url}} do handlebar

**this**.containerFotoCerveja.append(htmlFotoCerveja); // rederiza o template handlebars no lugar da div upload-drop

// dentro da div com a classe .js-container-foto-cerveja

$('.js-remove-foto').on('click', onRemoverFoto.bind(**this**));

}

**function** onRemoverFoto(){

$('.js-foto-cerveja').remove();

**this**.uploadDrop.removeClass('hidden') // exibe novamente a div upload-drop que contem o link de selecao da foto

**this**.inputNomeFoto.val('');

**this**.inputContentType.val('');

**this**.novaFoto.val('false');

}

**function** adicionarCsrfToken(xhr){

**var** token = $('input[name=\_csrf]').val();

**var** header = $('input[name=\_csrf\_header').val();

xhr.setRequestHeader(header, token);

}

**return** UploadFoto;

})();

$(**function**(){ // executa uma funcao "metodo" da classe Brewer

**var** uploadFoto = **new** Brewer.UploadFoto();

uploadFoto.iniciar();

});

**Handlebar – Na tela de vendas pesquisa rapida**

**No html cadastroVendas**

<!-- block de mensagens de erro inicia oculto, se houver erros sera exibido -->

<div class=*"alert alert-danger alert-dismissible hidden js-mensagem-erro"* role=*"alert"*>

<button type=*"button"* class=*"close"* data-dismiss=*"alert"* >

<span aria-hidden=*"true"*>&times;</span></button>

<div><i class=*"fa fa-exclamation-circle"*></i> Informe pelo menos 3 letras na pesquisa</div>

</div>

<!-- no container abaixo sera inserido o html de pesquisa rapida com handlebars -->

<div id=*"containerTabelaPesquisaRapidaclientes"*>

</div>

<!-- block html da pesquisa rapida que sera inserido na div containerTabelaPesquisaRapidaclientes -->

<th:block th:replace=*"hbs/TabelaPesquisaRapidaClientes"*></th:block>

**No Controller**

/\*\*Esse metodo sera um GET acessado via Ajax, contentType application/json, por isso deve retornar **@ResponseBody** para o javascript interpretar objetos json no browser \*/

@RequestMapping(consumes = { MediaType.***APPLICATION\_JSON\_VALUE*** })

**public** @ResponseBody List<Cliente> pesquisar(String nome){

validarTamanhoNome(nome);

**return** clientes.findByNomeStartingWithIgnoreCase(nome);

}

/\*\* Tratamento da quantidade minima de caracter que o usuario deve digitar no campo de pesquisa rapida de cliente \*/

**private** **void** validarTamanhoNome(String nome) {

**if**(StringUtils.*isEmpty*(nome) || nome.length() < 3){

**throw** **new** IllegalArgumentException();

}

}

/\*\* Trata a Exception IllegalArgumentException para esse controller quando a execao Exception for lancada evitando a exibição da pilha da execcao no log da aplicação, retorna badRequest Erro 400 (erro provocado pelo usuario) \*/

@ExceptionHandler(IllegalArgumentException.**class**)

**public** ResponseEntity<Void> tratarIllegalArgumentException(IllegalArgumentException e){

**return** ResponseEntity.*badRequest*().build();

}

**No Javascript**

Brewer = Brewer || {}

Brewer.PesquisaRapidaCliente = (**function** () {

**function** PesquisaRapidaCliente(){

**this**.pesquisaRapidaClientesModal = $('#pesquisaRapidaClientes');

**this**.nomeInput = $('#nomeClienteModal');

**this**.pesquisaRapidaBtn = $('.js-pesquisa-rapida-clientes-btn');

**this**.containerTabelaPesquisa = $('#containerTabelaPesquisaRapidaclientes');

**this**.htmlTabelaPesquisa = $('#tabela-pesquisa-rapida-cliente').html(); // pega pagina html que contem as tags do rendlebars

**this**.template = Handlebars.compile(**this**.htmlTabelaPesquisa); // compila html com as tags do rendlebars

**this**.mensagemErro = $('.js-mensagem-erro');

}

PesquisaRapidaCliente.prototype.iniciar = **function** () {

**this**.pesquisaRapidaBtn.on('click', onPesquisaRapidaClicado.bind(**this**));

**this**.pesquisaRapidaClientesModal.on('shown.bs.modal', onModalShow.bind(**this**));

}

**function** onModalShow(){

**this**.nomeInput.focus();

}

**function** onPesquisaRapidaClicado (event){

// desabilitar o evento padrao de submissao automatica do formulario via Ajax ao carregar a pagina

event.preventDefault();

$.ajax({

url: **this**.pesquisaRapidaClientesModal.find('form').attr('action'), // pegando url do formulario de pesquisa rapida

method: 'GET',

contentType: 'application/json',

data: {

nome: **this**.nomeInput.val() // "nome:" e o nome do parametro do metodo que recebera a informacao enviada

},

success: onPesquisaConcluida.bind(**this**),

error: onErroPesquisa.bind(**this**)

});

}

**function** onErroPesquisa(){

**this**.mensagemErro.removeClass('hidden');

}

**function** onPesquisaConcluida(resultado){

**this**.mensagemErro.addClass('hidden'); // esconde div de mensagem de erros

**var** html = **this**.template(resultado); // passa resultado "array de objetos" para a template do rendlebar fazer o parse retornar os dados e o html

**this**.containerTabelaPesquisa.html(html); // retorna ao browser dentro da div containerTabelaPesquisaRapidaclientes o html gerado pelo rendlebar

// Cria o objeto da classe TabelaClientePesquisaRapida, e passa o formulario modal como parametro

**var** tabelaClientePesquisaRapida = **new** Brewer.TabelaClientePesquisaRapida(**this**.pesquisaRapidaClientesModal);

tabelaClientePesquisaRapida.iniciar();

}

**return** PesquisaRapidaCliente;

}());

// obtem o cliente selecionado, seu nome e codigo e seta os campos nomeCliente e codigoCliente (hidden) do formulario de vendas

Brewer.TabelaClientePesquisaRapida = (**function**() {

**function** TabelaClientePesquisaRapida(modal){

**this**.modalCliente = modal;

**this**.cliente = $('.js-cliente-pesquisa-rapida');

}

TabelaClientePesquisaRapida.prototype.iniciar = **function**(){

**this**.cliente.on('click', onClienteSelecionado.bind(**this**)); // ao selecionar um cliente retornado na tabela

}

**function** onClienteSelecionado(evento){

**this**.modalCliente.modal('hide'); // remove o modal

**var** clienteSelecionado = $(evento.currentTarget); // retorna exatamento cliente que foi clicado na tabela

$('#nomeCliente').val(clienteSelecionado.data('nome')); // seta o nome no campo

$('#codigoCliente').val(clienteSelecionado.data('codigo')); // seta o campo "hidden" codigoCliente

}

**return** TabelaClientePesquisaRapida;

}());

$(**function**(){

**var** pesquisaRapidaCliente = **new** Brewer.PesquisaRapidaCliente();

pesquisaRapidaCliente.iniciar();

});

**No html com código handlebars – PesquisRapidaCliente.html**

<script id=*"tabela-pesquisa-rapida-cliente"* type=*"text/x-handlebars-template"*>

<div class="table-responsive bw-tabela-simples" style="margin-top: 15px">

<table class="table table-hover">

<thead>

<tr>

<th class="col-sm-2">Nome</th>

<th class="col-sm-2">Telefone</th>

</tr>

</thead>

<tbody>

<!-- inicio do codigo do Handlebars recebe uma lista -->

{#if .}} <!-- se algum objeto "." for passado -->

{{#each .}} <!-- para cada objeto passado na lista cria uma linha com codigo e nome -->

<tr data-codigo="{{codigo}}" data-nome="{{nome}}" class="js-cliente-pesquisa-rapida">

<td>{{nome}}</td>

<td>{{telefone}}</td>

</tr>

{{/each}}

{{else}} <!-- caso contrario exibe a linha com msg -->

<tr>

<td colspan="2">Nenhum cliente encontrado</td>

</tr>

{{/if}}

</tbody>

</table>

</div>

</script>

**AutoComplete EasyAutoComplete - Na tela de vendas**

Baixar e instalar

*easy-autocomplete.min.css, easy-autocomplete.themes.min.css*

*jquery.easy-autocomplete.min.js*

Campo de pesquisa dinâmico (autocomplete), conforme vai sendo digitado o conteúdo da pesquisa o componente vai retornando a o resultado da a consulta.

Sera utilizado o **EasyAutoComplete** PlugIn do JQuery, por permitir exibir imagens junto com os dados retornados.

**No html - cadastroVenda**

<link rel=*"stylesheet"* th:href=*"@{/stylesheets/vendors/easy-autocomplete.min.css}"*/>

<link rel=*"stylesheet"* th:href=*"@{/stylesheets/vendors/easy-autocomplete.themes.min.css}"*/>

<script th:src=*"@{/javascripts/vendors/jquery.easy-autocomplete.min.js}"*></script>

<th:block th:replace=*"hbs/TemplateAutocompleteCerveja"*></th:block>

**Venda - Mapeamento OneToMany**

**Entre Cerveja e ItemVenda **

**Na entidade Cerveja.java**

Nada, porque a navegação é de itemVenda para cerveja

**Na Entidade ItemVenda.java**

@ManyToOne

@JoinColumn(name = "codigo\_cerveja")

**private** Cerveja cerveja;

**Entre Venda e ItemVenda**

****

**Na Entidade Venda.java – mapeamento**

A navegação é bidirecional em ambas entidades tem que haver mapeamento

Na entidade venda - mapeamento

/\*\* orphanRemoval = true, para quando da edição ou exclusao de registros,

O hibernate remover os itens orfão da lista e também do banco. caso contrario sera

duplicado e inserido uma nova lista com os mesmos registros no banco

CascadeType.ALL para que o JPA ao salvar uma venda salve tambem os itens da venda\*/

@OneToMany(mappedBy = "venda", cascade = CascadeType.***ALL***, orphanRemoval = **true**)

**private** List<ItemVenda> itens = **new** ArrayList<>();

Na entidade ItemVenda.java - mapeamento

@ManyToOne

@JoinColumn(name = "codigo\_venda")

**private** Venda venda;

**Na Entidade Venda - setando a Venda nos Itens vendidos**

/\*\* seta a venda no itemVenda \*/

**public** **void** adicionarItens(List<ItemVenda> itens) {

**this**.itens = itens;

// para cada itemVenda seta o atributo codigo\_venda - this é a venda que esta sendo criada

**this**.itens.forEach(i -> i.setVenda(**this**));

}

**No controller Venda - validando venda**

**private** **void** validarVenda(Venda venda) {

// recupera os itens da venda da tabela de intens, e seta a venda nos itens, ao salvar a venda

venda.adicionarItens(tabelaItens.getItens(venda.getUuid()));

// calcula o valor total da venda

venda.calcularValorTotal();

}

**No controller Venda – buscando as cervejas por sku ou Nome**

/\*\* Recupera a cerveja ao digitar o sku ou nome \*/

@RequestMapping(consumes = MediaType.***APPLICATION\_JSON\_VALUE***)

**public** @ResponseBody List<CervejaDTO> pesquisar(String skuOuNome){

**return** cervejas.porSkuOuNome(skuOuNome);

}

**Div que trata busca da busca dos itens e adição no carrinho de compras**

<div class=*"tab-pane active"* id=*"cervejas"*>

<!-- input da busca e pesquisa dos itens de compras -->

<div class=*"row"*>

<div class=*"form-group col-lg-12"*>

<input type=*"text"* class=*"form-control js-sku-nome-cerveja-input"* id=*"cerveja"*

placeholder=*"Pesquise e adicione a cerveja pelo SKU ou nome"* autofocus=*"autofocus"*

data:url=*"@{/cervejas}"* th:disabled=*"${venda.salvarProibido}"*/>

</div>

</div>

<!-- div em que sera apresentada o carrinho de compras das cervejas -->

<div class=*"bw-tabela-cervejas js-tabela-cervejas-container"* data:valor=*"${valorTotalItens}"*>

<th:block th:replace=*"venda/TabelaItensVenda"*></th:block>

</div>

</div>

**No html – TabelaItensVenda (Carrinho de compras)**

<!DOCTYPE html>

<html xmlns=*"http://www.w3.org/1999/xhtml"*

xmlns:th=*"http://www.thymeleaf.org"*

xmlns:data=*"http://www.thymeleaf.org/extras/data"*>

<div class=*"bw-tabela-item js-tabela-item"* th:each=*"item : ${itens}"* data:valor-total=*"${valorTotal}"*>

<div class=*"bw-tabela-item\_\_coluna bw-tabela-item\_\_coluna--foto"*>

<img th:src=*"${item.cerveja.urlThumbnailFoto}"* class=*"img-responsive"*>

</div>

<div class=*"bw-tabela-item\_\_coluna bw-tabela-item\_\_coluna--detalhes"*>

<span class=*"bw-tabela-cerveja-nome"* th:text=*"${item.cerveja.nome}"*>Negra Modelo Long Neck 355ml

</span>

<span class=*"label label-default"* th:text=*"${item.cerveja.sku}"*>AA1111</span>

<span class=*"bw-tabela-cerveja-origem"* th:text=*"${item.cerveja.origem.descricao}"*>Importada</span>

</div>

<div class=*"bw-tabela-item\_\_coluna bw-tabela-item\_\_coluna--valores"*>

<div class=*"bw-tabela-cerveja-valor-pequeno"*>

<input type=*"text"* maxlength=*"3"* class=*"bw-tabela-cerveja-campo-quantidade js-tabela-cerveja-quantidade-item"* th:value=*"${item.quantidade}"* data:codigo-cerveja=*"${item.cerveja.codigo}"*

th:disabled=*"${venda != null ? venda.salvarProibido : false}"*>

<span>x R$[[${{item.valorUnitario}}]]</span>

</div>

<div class=*"bw-tabela-cerveja-valor-grande"*>R$[[${{item.valorTotal}}]]</div>

</div>

<div class=*"bw-tabela-item\_\_painel-exclusao"* th:if=*"${venda != null ? venda.salvarPermitido : false}"*>

<span class=*"bw-tabela-item\_\_titulo-exclusao"*>Excluir este item da venda?</span>

<button type=*"button"* class=*"btn btn-danger js-exclusao-item-btn"*

data:codigo-cerveja=*"${item.cerveja.codigo}"*>Sim, excluir!</button>

</div>

</div>

<div class=*"bw-tabela-cervejas\_\_vazio"* th:if=*"${itens.empty}"*>

<i class=*"glyphicon glyphicon-shopping-cart"*></i>

<span>Você ainda não adicionou nenhuma cerveja.</span>

</div>

</html>

**Ainda Tabela Itens de venda com junit**

Colocar dependência do **junit** no pom.xml e

**<scope>test<scope>**  - somente durante desenvolvimento que sera utilizado, ao empacotar pra produção não envia essas classes de testes

**Na classe ItemVenda**

@Entity

@Table(name = "item\_venda")

**public** **class** ItemVenda {

@Id

@GeneratedValue(strategy= GenerationType.***IDENTITY***)

**private** Long codigo;

**private** Integer quantidade;

@Column(name = "valor\_unitario")

**private** BigDecimal valorUnitario;

@ManyToOne

@JoinColumn(name = "codigo\_cerveja")

**private** Cerveja cerveja;

@ManyToOne

@JoinColumn(name = "codigo\_venda")

**private** Venda venda;

/\*\* calcula o valor total do item - valorUnitario x quantidade \*/

**public** BigDecimal getValorTotal(){

**return** valorUnitario.multiply(**new** BigDecimal(quantidade));

}

... get e set e hascode equals

**}**

**Nas classe TabelaItensVenda**

Classe auxiliar que representa a sessão da venda, fara o papel do carrinho de compra, onde será possível totalizar a venda, e informar todos os itens da venda

**Criar um objeto (TabelaItensVenda) por sessão de usuário**

Colocar no componente tabelaItensVenda a anotation **@SessionScope,** sendo assim para cada usuário logado sera criado uma sessão independente.

**Controlando sessão de usuário por visão**

Tendo em vista que o usuário pode abrir novas abas para atender outros pedidos. Sera criado um controle para evitar que ao inserir itens de venda em uma aba a mesma interfira na lista de itens de outra aba.

Para tanto sera gerado um identificador (uuid) para a classe TabelaItensVenda.

A classe TabelaItensSession que ira controlar a sessão do usuário, e conterá uma lista com varias tabelas, cada uma representando uma aba onde esta sendo realizado a venda.

Quando as sessão vão sendo expirada as tabela também vão sendo apagadas.

O uuid sera adicionado a venda, no momento da criação da venda no método “novo” no controller.

**No .HTML – TemplateAutocompleteCerveja com handlebars**

Esse template html/handlebars é utilizado na pesquisa da cerveja, na tela de venda, durante o cadastro e edição da venda

<script id=*"template-autocomplete-cerveja"* type=*"text/x-handlebars-template"*>

<div class="bw-tabela-item">

<div class="bw-tabela-item\_\_coluna bw-tabela-item\_\_coluna--foto">

<img src="{{urlThumbnailFoto}}" class="img-responsive" />

</div>

<div class="bw-tabela-item\_\_coluna bw-tabela-item\_\_coluna--detalhes">

<span class="bw-tabela-cerveja-nome">{{nome}}</span>

<span class="label label-default">{{sku}}</span>

<span class="bw-tabela-cerveja-origem">{{origem}}</span>

</div>

<div class="bw-tabela-item\_\_coluna bw-tabela-item\_\_coluna--valores">

<div class="bw-tabela-cerveja-valor-medio">R${{valorFormatado}}</div>

</div>

</div>

</script>

**Input do EasyAutoComplete**

<div class=*"row"*>

<div class=*"form-group col-lg-12"*>

<input type=*"text"* class=*"form-control js-sku-nome-cerveja-input"* id=*"cerveja"*

placeholder=*"Pesquise e adicione a cerveja pelo SKU ou nome"*

autofocus=*"autofocus"*

data:url=*"@{/cervejas}"* th:disabled=*"${venda.salvarProibido}"*/>

</div>

</div>

**Na Entidade VendaValidator.java – Validação**

**No javascritp**

**Biblioteca javascript para formatação de numeros**

**Numeral.js**

**numeral.min.js (configurados no LayoutPadrao.html)**

**pt-br.min.js**

**Funcionalidade:** Recebe um valor, e uma mascara que é aplicada e retorna uma string

**MaskMoney**

**Funcionalidade:** Enquando o usuario vai digitando um valor, vai aplicando uma mascara automaticamente e retorna uma string

**Jquery-masknumber.min.js**

Plugin do jquery para resolver problema de formatação de números no celular.

No celular no caso da digitação de campo numericos, o plugin JQueryMaskMoney não aplica a mascara no momento da digitação, mas somente ao sair do campo, o que pode tornar um inconveniente, para o usuario.

Jquery-masknumber.min.js é um plugin que resolve esse problema, foi desenvolvido por Normandes Junior da AlgaWorks baixar em:

<https://github.com/normandesjr/jquery-mask-number>

**Retornar dados parcialmente de uma tabela usando DTO**

**Na classe DTO - CervejaDTO**

/\*\* Essa classe foi criada para recuperar parcialmente os dados da tabela cerveja \*/

**public** **class** CervejaDTO {

**private** Long codigo;

**private** String sku;

**private** String nome;

**private** String origem;

**private** BigDecimal valor;

**private** String foto;

**private** String urlThumbnailFoto;

// A origem e um string que vem do banco mas sera convertido para um Enum pelo Spring automaticamente

**public** CervejaDTO(Long codigo, String sku, String nome, Origem origem, BigDecimal valor, String foto) {

**this**.codigo = codigo;

**this**.sku = sku;

**this**.nome = nome;

**this**.origem = origem.getDescricao(); // obtendo a descrição do Enum origem

**this**.valor = valor;

**this**.foto = StringUtils.*isEmpty*(foto) ? "cerveja-mock.png" : foto;

}

...

Get e sets

....

**Na consulta JPQL**

@Override

**public** List<CervejaDTO> porSkuOuNome(String skuOuNome) {

String queryJpql = "select new com.algaworks.brewer.dto.CervejaDTO(codigo, sku, nome, origem, valor, foto)"

+ "from Cerveja where lower(sku) like lower(:skuOuNome) or lower(nome) like lower(:skuOuNome)";

List<CervejaDTO> cervejasFiltradas = manager.createQuery(queryJpql, CervejaDTO.**class**)

.setParameter("skuOuNome", skuOuNome + "%")

.getResultList();

cervejasFiltradas.forEach(c -> c.setUrlThumbnailFoto(fotoStorage

.getUrl(FotoStorage.***THUMBNAIL\_PREFIX*** + c.getFoto())));

**return** cervejasFiltradas;

}

**IOC – injeção de dependência**

AplicationContext

Dependencia - spring-context

Beans - são classes anotadas com a anotação @Component

Autowired – injeta um objeto na classe ex:

@Component

public class Checkout {

@Autowired

private Impressora impressora;

public void finalizar() {

// regras do checkout

this.impressora.imprimir("compra10.csv");

}

}

Ex:

public class Main {

public static void main(String[] args) {

ApplicationContext context = new AnnotationConfigApplicationContext(AppConfig.class);

Checkout checkout = context.getBean(Checkout.class);

checkout.finalizar();

((ConfigurableApplicationContext) context).close();

}

}

@Configuration

public class AppConfig {

@Bean

public Impressora impressora() {

return new ImpressoraEpson();

}

@Bean

public Checkout checkout(Impressora impressora) {

return new Checkout(impressora);

}

}

**Escopo de beans – quando o objeto e criado e destruído o ciclo de vida do objeto Bean**

Padrão é singleton – entrega sempre o mesmo objeto bean para o aplicationContext – não usa anotation na classe bean

Com anotation usado na classe

@Scope - Escopo de sessão

@Scope(“prototype”) Escopo prototype Cria um novo objeto bean a cada vez que chamado pelo context

**JPA e Hibernate**

**Jpa** especificação java do java para incluir alterar inserir objetos no banco de dados para o

**HIBERNATE – ORM –** TRANSFORMA

Cada classe será uma tabela

Cada linha será um objeto na tabela

**Flyway –** Versionador de banco de dados, migrador, ajuda a evoluir e reverter versões do banco de dados, o hibernate somente gera não permite reverter etc.

Adicionar dependencia no pom.xml, informar também o driver de banco que ele vai usar

No pom

EX: do nome de uma migração usando a convenção do Flyway

V01\_\_o\_que\_ela\_faz\_no\_banco

Na criação de tabela usar ENGINE=InnoDB para garantir a consistência do banco, caso algum registro estiver sendo usado, o mesmo não poderá ser apagado da tabela.

Ex:

CREATE TABLE estilo (

codigo BIGINT(20) PRIMARY KEY AUTO\_INCREMENT,

nome VARCHAR(50) NOT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8;

CREATE TABLE cerveja (

codigo BIGINT(20) PRIMARY KEY AUTO\_INCREMENT,

sku VARCHAR(50) NOT NULL,

nome VARCHAR(80) NOT NULL,

descricao TEXT NOT NULL,

valor DECIMAL(10, 2) NOT NULL,

teor\_alcoolico DECIMAL(10, 2) NOT NULL,

comissao DECIMAL(10, 2) NOT NULL,

sabor VARCHAR(50) NOT NULL,

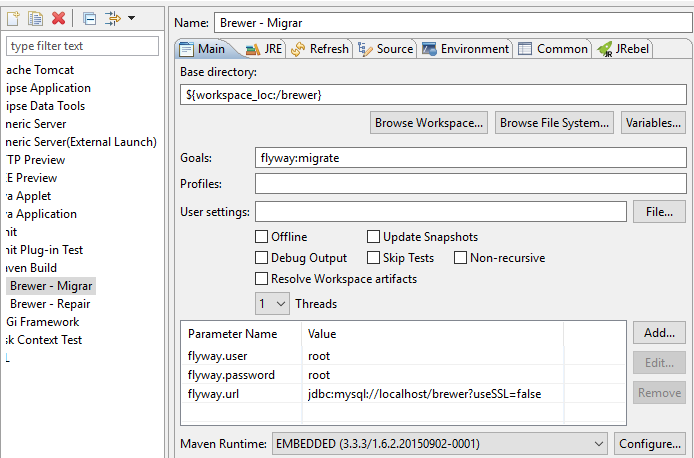
origem VARCHAR(50) NOT NULL,

codigo\_estilo BIGINT(20) NOT NULL,

FOREIGN KEY (codigo\_estilo) REFERENCES estilo(codigo)

) ENGINE=InnoDB DEFAULT CHARSET=utf8;

**Configurar o script maven para o flyway**



No banco o flyway cria a tabela schema\_version, para gerencia as versões das migrações.

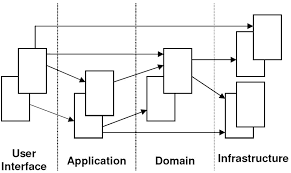
**JPQL – JAVA PERSISTENCE QUERY LANGUAGE**

Utilizado para pesquisar, update, delete utilizando string e objetos

**SPRING DATA AND JPAs**

**Spring Data** já implementa automaticamente métodos para acesso a dados no banco de dados.

**Fornece a abstração JpaRepository**



Para diversos métodos tais como CRUD (insert, delete, update, consultas basicas (retrieve))e pesquisas tais como findAll(), findOne, findId(), paginação etc... nte objetos

Com JpaRepository Pode ser criada varias consultas sem escreve-las usando apenas o nome do atributo e um verbo e métodos padrão de consultas sql.

Ex:

@Repository

public interface Estilos extends JpaRepository<Estilo, Long>, EstilosQueries {

public Optional<Estilo> findByNomeIgnoreCase(String nome);

}

Pode ser utilizado um Optional do java 8; para tratar a existência do objeto pesquisado, onde caso não exista evitar uma exception.

**Ex:**

Cerveja cerveja = cerveja.findById() // gerari uma uma exception caso findById retorne null

Optional<Cerveja> cerveja = cerveja.findById() // não retorna null e pode ser testado assim cerveja.isPresent();

**HIBERNATE CRITERIA**

Utilizado para pesquisar, update, delete e inserir utilizando somente objetos

**Mapeamento das entidade para o JPA**

Dependencias no pom.xml

– hibernate-entitymanager (já vem com o JPA)

- hibernate-java8 (implementa LocalDate do java8)

Implementar a interface serializable

Criar o atributo código

@Id

@GeneratedValue(strategy= GenerationType.IDENTITY)

private Long codigo;

Separar nomes compostos com andescor “\_” para serem criado no banco

@Column(name = “nome\_composto”)

private nomeComposto;

Enum utilizar STRING para salvar no banco o nome em vez do código

@Enumerated(EnumType.STRING)

private Origem origem;

Mapeamento ManyToOne entre Estilo e Cerveja

**Estilo**

@OneToMany(mappedBy = "estilo")

private List<Cerveja> cervejas;

**Cerveja**

@ManyToOne

@JoinColumn(name = "codigo\_estilo")

private Estilo estilo;

Gerar gets e sets,

Gerar hashCode e equals (somente do código)

**Mapeamento de Enum**

**Ex:**

public enum Origem {

NACIONAL("Nacional"),

INTERNACIONAL("Internacional");

private String descricao;

Origem(String descricao){

this.descricao = descricao;

}

public String getDescricao(){

return descricao;

}

}

**Ligando os campos da tela com o controller**

**No controller**

O método recebe o objeto processado pelo thymelead na tag <form> tanto para edição como exclusao

@RequestMapping("/nova")

**public** ModelAndView nova(Cerveja cerveja){

**No html**

**th:objetc** – envia e recebe o objeto para o método no controller

<form method=*"POST"* th:object=*"${cerveja}"* class=*"form-vertical js-form-loading"*

th:action=*"${cerveja.nova} ? @{/cervejas/nova} : @{/cervejas/{codigo}(codigo=${cerveja.codigo})}"* >

o código html acima trada edição e inclusão do objeto, se for enviado um código com um método POST, então é uma edição.

Se não se for enviado um post com somente uma url “/nova” então uma inserção

**th:action -** é utilizadopara o springSecurity adicionar segurança contra ataques csrf na pagina, o **csrf** é controlado nas configurações da classe SecurityConfig.java ex: .csrf().disabled();

**Bind (ligação) de simples atributos do form com o objeto no controller**

th:object =”${nomeObjeto}” – inseridos no form

th:field=”\*{nomeAtributoObjeto}” - inseridos nos campos do form

para controlar a edição o elemento input abaixo devera estar presente no form

<input type="hidden" th:field="\*{codigo}" />

**Bind de atributos (objetos) do form com o objeto no controller**

Deve ser criada uma classe conversora e declarada na classe webconfig

**Ex: classe conversora**

@Override

public Estilo convert(String codigo) {

if(!StringUtils.isEmpty(codigo)){

Estilo estilo = new Estilo();

estilo.setCodigo(Long.valueOf(codigo));

return estilo;

}

return null;

}

**Ex: declaração na classe WebConfig**

@Bean

public FormattingConversionService mvcConversionService (){

DefaultFormattingConversionService conversionService = new DefaultFormattingConversionService();

conversionService.addConverter(new EstiloConverter());

return conversionService;

}

**Template de exibição de mensagens na view**

<brewer:message/>

**Sinalizador visual de obrigatoriedade de preenchimento do campo**

*“bw-required"*

Para controlar a exibição do asterisco de obrigatoriede de preenchimento de campo é utilizado a classe css acima, no div do input.

**Ver Brewer.css**

*.bw-required* > *.control-label:after* {

content: *"\*"*;

}

**Atributo customizado de controle de exibição de erros ver dialect**

Sempre que for lançado algum erro pelo controller esse componente destaca em vermelho na tela o campo em que o erro aconteceu.

brewer:classforerror=*"nomeAtributoObjeto"*

**Transação**

Utilizar a anotation @EnableTransactionManagement

abaixo na classe de configuração do JPA para controlar manualmente a transação, caso contrario o spring fara controle de transação para consulta também o que não é necessário

***Ex:***

@EnableTransactionManagement

**public** **class** JPAConfig {

e utilizar a anotation @Transactional nos métodos de serviço que fazem uso do banco

***Ex:***

@Transactional

**public** **void** salvar(Cerveja cerveja){

**Formatação de valores**

**Na classe WebConfig.java**

**Afim de evitar erros de conversão de valores para o formato brasileiro Implementar formator especifico no método abaixo**

@Bean

public FormattingConversionService mvcConversionService (){

.....

//implementacao customizada para conversao de valores, em função da intercionalizacao

BigDecimalFormatter bigdecimalFormatter = new BigDecimalFormatter("#,##0.00");

conversionService.addFormatterForFieldType(BigDecimal.class, bigdecimalFormatter);

...

}

**Validando valores Mínimo e Máximo permitido em um campo da tela e entradas nulas**

@NotNull(message = "Valor é Obrigatorio")

@DecimalMin("0.01")

@DecimalMax(value = "9999999.99", message = "O valor da cerveja deve ser menor que R$9.999.999,99")

private BigDecimal valor;

@NotBlank(message = "A descrição é obrigatoria")

@Size(max = 50, message = "O tamanho da descrição deve estar entre 1 e 50")

private String descricao;

**Validando**

**JavaScript Mascara de valores Decimal e inteiro**

**No arquivo Brewer.js**

var Brewer = Brewer || {};

Brewer.MaskMoney = (function(){

function MaskMoney(){

this.decimal =$('.js-decimal');

this.plain =$('.js-plain');

}

MaskMoney.prototype.enable = function(){

//this.decimal.maskMoney({ decimal: ',' , thousands: '.' });

//this.plain.maskMoney({ precision:0, thousands: '.' });

this.decimal.maskNumber({ decimal: ',' , thousands: '.' });

this.plain.maskNumber({ integer: true, thousands: '.' });

}

return MaskMoney;

}());

$(function(){

var maskMoney = new Brewer.MaskMoney();

maskMoney.enable();

}());

**Customizando validação com anotation**

@Target({ ElementType.FIELD, ElementType.METHOD, ElementType.ANNOTATION\_TYPE})

@Retention(RetentionPolicy.RUNTIME)

@Constraint(validatedBy = {})

@Pattern(regexp = "([a-zA-Z]{2}\\d{4})?") //regex - [a-zA-Z] permite caracteres de a à z maiusculo e minusculo, até dois digitos e \\d{4} quatro digitos numericos e só valida se tiver preenchido o campo

public @interface SKU {

@OverridesAttribute(constraint = Pattern.class, name = "message")

String message() default "{com.algaworks.constraints.SKU.message}";

Class<?>[] groups() default {};

Class<? extends Payload>[] payload() default {};

}

**Carregando imagens e tratando com spring com UIKIT**

framework de javascript UIKIT Uma estrutura de front-end leve e modular para desenvolver interfaces web rápidas e poderosas.

**O destaque da seleção do campo quando a foto é passada por cima é ativada via o javascript abaixo:**

**Dependências:**

<script th:src=*"@{/javascripts/vendors/uikit.min.js}"*></script>

<script th:src=*"@{/javascripts/vendors/upload.min.js}"*></script

*<link rel="stylesheet" th:href="@{/stylesheets/vendors/upload.min.css}"/>*

**Html**

<div class=*"js-container-foto-cerveja"* th:attr=*"data-url-fotos=@{/fotos}"*>

<div id=*"upload-drop"* class=*"bw-upload"* >

<i class=*"glyphicon glyphicon-cloud-upload"*></i>

<span>Arraste a foto aqui ou</span>

<a class=*"bw-upload-form-file"*>Selecione <input id=*"upload-select"* type=*"file"*

accept=*".jpg,.jpeg,.png"*/>

</a>

</div>

</div>

**Enviando a foto para o servidor via Javascript and Ajax:**

**CSS:**

/\* Upload cadatro cerveja \*/

*.bw-upload* {

padding: *15px*;

border: *1px* *dashed* *#ddd*;

border-radius: *4px*;

background: *#fafafa*;

text-align: *center*;

}

/\* Upload cadastro Vendas \*/

*.bw-upload-vendas* {

padding: *15px*;

border: *1px* *dashed* *#ddd*;

border-radius: *4px*;

background: *#fafafa*;

text-align: *center*;

margin: *0px* *0px* *12px* *0px*;

}

*.bw-upload-form-file* {

display: *inline-block*;

vertical-align: *middle*;

position: *relative*;

overflow: *hidden*;

margin-bottom: *1px*;

}

*.bw-upload-form-file* > **input**[type=file] {

position: *absolute*;

top: *0*;

left: *0*;

z-index: *1*;

width: *100%*;

opacity: *0*;

cursor: *pointer*;

font-size: *500px*;

}

**No Controller**

/\*\* A vantagem ao anotar a classe com **@RestController** é que e a classe continua sendo tambem um **@Controller** e nao é necessario utilizar **@ResponseBoy** no retorno dos metodos para trabalhar com Ajax e Json, ex: public **@ResponseBoy** String upload (){..} - não seria necessario o **@ResponseBody**, já esta lá mas nao de forma explita \*/

@RestController

@RequestMapping("/fotos")

**public** **class** FotosController {

@Autowired

**private** FotoStorage fotoStorage;

/\*\* Para o servidor receber requisição tipo MultpartFile (arquivos) de ser configurado

na classe AppInitializer.java no metodo customizeRegistration

DeferredResult<FotoDTO> aguarda até que o retorno esteja disponivel

utilizado para melhorar a disponibilidade da aplicacao via Thread assincrona, liberando

o metodo para receber novas requisições enquanto processa outras requisoes anteriores

\*/

@PostMapping

**public** DeferredResult<FotoDTO> upload(@RequestParam("files[]") MultipartFile[] files){

DeferredResult<FotoDTO> resultado = **new** DeferredResult<>();

// cria thread para a classe FotoStorageRunnable que executa o processo de renomeacao salvamento da foto

Thread thread = **new** Thread(**new** FotoStorageRunnable(files, resultado, fotoStorage));

thread.start();

**return** resultado;

}

@GetMapping("/{nome:.\*}")

**public** **byte**[] recuperar(@PathVariable String nome){

**return** fotoStorage.recuperar(nome);

}

}

**Filtro de pesquisa –**

**Criar uma classe .java com apenas os atributos que serviram de filtro**

**Para pesquisar a entidade**

**Deve ser usado o método GET e th:object=”nomeClasseFilter” no form ex:**

<form method=*"GET"* th:object=*"${cervejaFilter}"*>

**No HTML - Tabela de retorno da pesquisar**

**Se nenhum filtro for informado retorna toda a tabela, paginada 2 registro por pagina**

**Cabeçalho da tabela - colunas**

<div class=*"table-responsive bw-tabela-simples"*>

<table class=*"table table-hover"*>

<thead>

<tr>

<th class=*"table-cervejas-col-foto"*></th>

<th class=*"table-cervejas-col-sku"*>

<brewer:order page=*"${pagina}"* field=*"sku"* text=*"SKU"*/>

</th>

<th class=*"table-cervejas-col-nome"*>

<brewer:order page=*"${pagina}"* field=*"nome"* text=*"Nome"*/>

</th>

<th class=*"table-cervejas-col-estilo"*>Estilo</th>

<th class=*"table-cervejas-col-origem"*>Origem</th>

<th class=*"table-cervejas-col-valor"*>Valor</th>

<th class=*"table-cervejas-col-acoes"*></th>

</tr>

</thead>

**Dados da tabela**

<tbody>

<tr th:each=*"cerveja : ${pagina.conteudo}"*>

<td class=*"text-center"*>

<!-- <img th:src="@{/fotos/thumbnail.{foto}(foto=${cerveja.fotoOrMock})}" class="img-responsive"/> -->

<img th:src=*"${cerveja.urlThumbnailFoto}"* class=*"img-responsive"*/> </td>

<td class=*"text-center"* th:text=*"${cerveja.sku}"*>AA1234</td>

<td th:text=*"${cerveja.nome}"*>Cerveja Stella</td>

<td th:text=*"${cerveja.estilo.nome}"*>Lager</td>

<td th:text=*"${cerveja.origem.descricao}"*>Internacional</td>

<td class=*"text-right"* th:text=*"|R$ ${{cerveja.valor}}|"*>R$ 8.00</td>

<td class=*"text-center"*>

<a class=*"btn btn-link btn-xs js-tooltip"* title=*"Editar"*

th:href=*"@{/cervejas/{codigo}(codigo=${cerveja.codigo})}"*>

<i class=*"glyphicon glyphicon-pencil"*></i>

</a>

<a class=*"btn btn-link btn-xs js-tooltip js-exclusao-btn"* title=*"Excluir"* href=*"#"*

data:url=*"@{/cervejas/{codigo}(codigo=${cerveja.codigo})}"*

data:objeto=*"${cerveja.nome}"*>

<i class=*"glyphicon glyphicon-remove"*></i>

</a>

</td>

</tr>

<tr th:if=*"${pagina.vazia}"*>

<td colspan=*"7"*>Nenhuma cerveja encontrada</td>

</tr>

</tbody>

</table>

</div>

**Ordenação de colunas da tabela exibida**

Sera utilizado também pageable da integraçãoentre o **SpringMVC** e o **SpringData**, para fazer a ordenação dos campos

**Para Ordenação foi criado um componente**

**HTML – um fragments**

<!DOCTYPE html>

<html lang=*"pt"*

xmlns=*"http://www.w3.org/1999/xhtml"*

xmlns:th=*"http://www.thymeleaf.org"*>

<th:block th:fragment=*"order (pagina, propriedade, texto)"*>

<a th:href=*"${pagina.urlOrdenada(propriedade)}"* th:remove=*"${pagina.vazia} ? tag"*>

<span th:text=*"${texto}"*></span>

<span class=*"dropup"* th:remove=*"${pagina.descendente(propriedade)} ? tag"*

th:if=*"${pagina.ordenada(propriedade)}"*>

<span class=*"caret"*></span>

</span>

</a>

</th:block>

</html>

**Classe Java - TagProcessor**

**public** **class** OrderElementTagProcessor **extends** AbstractElementTagProcessor{

**private** **static** **final** String ***NOME\_TAG*** = "order";

**private** **static** **final** **int** ***PRECEDENCIA*** = 100;

**public** OrderElementTagProcessor(String dialectPrefix) {

**super**(TemplateMode.***HTML***, dialectPrefix, ***NOME\_TAG***, **true**, **null**, **false**, ***PRECEDENCIA***);

}

@Override

**protected** **void** doProcess(ITemplateContext context, IProcessableElementTag tag,

IElementTagStructureHandler structureHandler) {

IModelFactory modelFactory = context.getModelFactory();

IAttribute page = tag.getAttribute("page");

IAttribute field = tag.getAttribute("field");

IAttribute text = tag.getAttribute("text");

IModel model = modelFactory.createModel();

model.add(modelFactory.createStandaloneElementTag("th:block"

, "th:replace"

, String.*format*("fragments/Ordenacao :: order (%s,%s,'%s')", page.getValue(), field.getValue(),text.getValue())));

// true para o thymeleaf processar false se fosse um html

structureHandler.replaceWith(model, **true**);

}

}

**Registro da classe Java – TagProcessor no Dialiect**

**BrewerDialect**

**public** **class** BrewerDialect **extends** AbstractProcessorDialect{

**public** BrewerDialect() {

**super**("AlgaWorks Brewer", "brewer", StandardDialect.***PROCESSOR\_PRECEDENCE***);

}

@Override

**public** Set<IProcessor> getProcessors(String dialectPrefix) {

**final** Set<IProcessor> processadores = **new** HashSet<>();

processadores.add(**new** ClassForErrorAttributeTagProcessor(dialectPrefix));

processadores.add(**new** MessageElementTagProcessor(dialectPrefix));

processadores.add(**new** OrderElementTagProcessor(dialectPrefix));

processadores.add(**new** PaginacaoElementTagProcessor(dialectPrefix));

processadores.add(**new** MenuAttributeTagProcessor(dialectPrefix));

**return** processadores;

}

}

**Chamada na pagina html:**

<th class=*"table-cervejas-col-sku"*>

<brewer:order page=*"${pagina}"* field=*"sku"* text=*"SKU"*/>

</th>

**Na paginação é utilizado o componente do bootstrap**

E existe uma integração entre o **SpringMVC** e o **SpringData** para seu funcionamento, através do objeto Pageable, o **link** deve ter obrigatoriamente o atributo de nome **page**? e o parâmetro Pageable no método do controller.

E na classe **Webconfig** a anotation **@EnabledSprigDataWebSupport**

- A classe PageWapper foi criada para auxiliar na paginação evitando perder o filtro ao mudar de pagina.

- Para evitar peder o enconding de valores foi utilizado no método abaixo da classe PageWapper **.build(true).encode().** Que codifica encoda ao enviar a url ex:

**public** String urlParaPagina(**int** pagina) {**return** uriBuilder.replaceQueryParam("page",

pagina).build(**true**).encode().toUriString();

**Calculando o total de paginas a ser mostrada no componente boostrap**

Na consulta deve ser calculado o total de registro e retornado um Page

EX:

**private** Long total(CervejaFilter filtro){

Criteria criteria = manager.unwrap(Session.**class**).createCriteria(Cerveja.**class**);

adicionarFiltro(filtro, criteria);

criteria.setProjection(Projections.*rowCount*());

**return** (Long) criteria.uniqueResult();

}

**return** **new** PageImpl<>(criteria.list(), pageable, total(filtro));

**Exibe html de paginação usa dialect brewer:paginacao**

<div class=*"col-sm-12 text-center"* th:if=*"${not pagina.vazia}"*>

<brewer:paginacao page=*"${pagina}"* />

</div>

**Paginação no servidor**

Entrega a cada requisição uma quantidade de registros ao cliente, conforme o tamanho da paginação

**Paginação no cliente**

Toda a tabela é entregue em um única requisição ao cliente, epaginada no cliente

**Lançando Event ao salvar - listener**

**Na classe Listener**

|  |
| --- |
| @Component |
|  | public class CervejaListener { |
|  |  |
|  | @Autowired |
|  | private FotoStorage fotoStorage; |
|  |  |
|  | @EventListener(condition = "#evento.temFoto()") |
|  | public void cervejaSalva(CervejaSalvaEvent evento) { |
|  | fotoStorage.salvar(evento.getCerveja().getFoto()); |
|  | } |
|  |  |
|  | } |

**Na classeEvent**

|  |
| --- |
| public class CervejaSalvaEvent { |
|  |  |
|  | private Cerveja cerveja; |
|  |  |
|  | public CervejaSalvaEvent(Cerveja cerveja) { |
|  | this.cerveja = cerveja; |
|  | } |
|  |  |
|  | public Cerveja getCerveja() { |
|  | return cerveja; |
|  | } |
|  |  |
|  | public boolean temFoto() { |
|  | return !StringUtils.isEmpty(cerveja.getFoto()); |
|  | } |
|  |  |
|  | } |

**Na classe Service**

|  |
| --- |
| @Service |
|  | public class CadastroCervejaService { |
|  |  |
|  | @Autowired |
|  | private Cervejas cervejas; |
|  |  |
|  | @Autowired |
|  | private ApplicationEventPublisher publisher; |
|  |  |
|  | @Transactional |
|  | public void salvar(Cerveja cerveja) { |
|  | cervejas.save(cerveja); |
|  |  |
|  | publisher.publishEvent(new CervejaSalvaEvent(cerveja)); |
|  | } |
|  |  |
|  | } |

**Redimensionar imagem thumbnailator**

As fotos das cervejas possuem o tamanho 253x432 px e seram redimensionadas para 48x60px

Site para testar redimensionamento de imagens:

<http://scriptygoddess.com/resources/proportioncalc.htm>

Adicionar no pom.xml do maven

<!-- thumbnailator - Gerar Thumbnail -->

<dependency>

<groupId>net.coobird</groupId>

<artifactId>thumbnailator</artifactId>

<version>${thumbnailator.version}</version>

<scope>compile</scope>

</dependency>

**Resolvendo problema da consulta n+1**

Consultas realizadas pelo jpa com relacionamentos, geram novas consultas as entidades relacionadas, para evitar esse problemas deve-se

**ser feita uma pesquisa especifica**

**Utilizar:**

EntidadeCerveja extends JpaRepository<Moldel,id>, EntidadeXQuery

EntidadeImpl implements EntidadeXQuery(implementação da consulta)

**Consultando com Criteria do Hibernate**

**Nota:** Consultas com criteria, caso haja relacionamentos os mesmos serão feitos automaticamente pelo criteria

**Melhorando a performance do sistema com cache**

**Utilizando GuavaCacheManager**

**colocar dependencia no pom.xml**

**Em 3 passos:**

1 - utilizar a anotation **@Cacheable(“nomeCache”)** no método

**Ex:**

@Cacheable(value = "cidades") // todo o cache

@Cacheable(value = "cidades", key="#codigoEstado") // cache identificado pelo codigoEstado

**public** @ResponseBody List<Cidade> pesquisarPorCodigoEstado(

@RequestParam(name = "estado", defaultValue = "-1") Long codigoEstado){...}

**2 –** colocar a anotation **@EnableCaching** na classe WebConfig

**3** – adicionar o metodo cacheManager na classe WebConfig

**Ex:**

@Bean

**public** CacheManager cacheManager(){

CacheBuilder<Object, Object> cacheBuilder = CacheBuilder.*newBuilder*()

.maximumSize(3)

.expireAfterAccess(20, TimeUnit.***SECONDS***);

GuavaCacheManager cacheManager = **new** GuavaCacheManager();

cacheManager.setCacheBuilder(cacheBuilder);

**return** cacheManager;

}

**Invalidando o cache ao cadastrar um novo item que esta sendo cacheado**

- Para que o cache seja atualizado ao ser inserido um novo objeto no banco de dados, o método que salva o novo objeto deve invalidar o cache de busca anterior ao salvamento.

Colocar a anotation **@CacheEvict** no método que ira salvar no banco o objetos que estão sendo cacheados.

**Ex1:** Invalida todas as entradas no cache

@CacheEvict(value="cidades", allEntries=true)

**public** ModelAndView salvar(@Valid Cidade cidade, BindingResult result, RedirectAttributes attributes){..}

**Ex2:** Invalida somente a entrada especificada pela key

// condition – o cache somente sera executado se tiver estado

// selecionado sem essa condition acontecera um erro ao salvar

@CacheEvict(value="cidades", key = "#cidade.estado.codigo", condition = "#cidade.temEstado()")

**public** ModelAndView salvar(@Valid Cidade cidade, BindingResult result, RedirectAttributes attributes){...}

**Usando anotation em classes para validar atributos**

**Criação de anotation para comparação de atributos senha e confirmacaoSenha**

Usa no pom.xml commons.beansutils

Tendo em vista que o atributo **confirmacaoSenha** não existe na tabela usuário e é transiente.

**Na classe Entidade:**

@AtributoConfirmacao(atributo="senha", atributoConfirmacao="confirmacaoSenha", message = "Confirmação da senha não confere")

**Essa anotação somente pode ser usada sobe classe, para poder ter acesso a todos atributos da classe por foi anotada com @Target({ElementType}). Na classe Anotation, a regra de validação propriamente dita sera implementada pela classe definida na anotation @Constraint -**  AtributoConfirmacaoValidator.**class da classe AtributoConfirmacao abaixo:**

@Target({ ElementType.***TYPE***})

@Retention(RetentionPolicy.***RUNTIME***)

@Constraint(validatedBy = { AtributoConfirmacaoValidator.**class** })

**public** **@interface** AtributoConfirmacao {

@OverridesAttribute(constraint = Pattern.**class**, name = "message")

String message() **default** "Atributos não conferem";

Class<?>[] groups() **default** {};

Class<? **extends** Payload>[] payload() **default** {};

String atributo();

String atributoConfirmacao();

}

**Na classe AtributoConfirmacaoValidator,** Object passado como argumento de ConstraintValidator<AtributoConfirmacao, Object> indica que a anotação AtributoConfirmacao pode ser utilizada por qualquer classe.

**public** **class** AtributoConfirmacaoValidator **implements** ConstraintValidator<AtributoConfirmacao, Object>{

**private** String atributo;

**private** String atributoConfirmacao;

@Override

**public** **void** initialize(AtributoConfirmacao constraintAnotation) {

**this**.atributo = constraintAnotation.atributo();

**this**.atributoConfirmacao = constraintAnotation.atributoConfirmacao();

}

@Override

**public** **boolean** isValid(Object object, ConstraintValidatorContext context) {

**boolean** valido = **false**;

**try** {

Object valorAtributo = BeanUtils.*getProperty*(object, **this**.atributo);

Object valorAtributoConfirmacao = BeanUtils.*getProperty*(object, **this**.atributoConfirmacao);

valido = ambosSaoNull(valorAtributo, valorAtributoConfirmacao) ||

ambosSaoIguais(valorAtributo, valorAtributoConfirmacao);

} **catch** (Exception e) {

**throw** **new** RuntimeException("Erro recuperando valores de atributos", e);

}

**if**(!valido){

context.disableDefaultConstraintViolation();

String mensagem = context.getDefaultConstraintMessageTemplate();

ConstraintViolationBuilder violationBuilder =

context.buildConstraintViolationWithTemplate(mensagem);

violationBuilder.addPropertyNode(atributoConfirmacao).addConstraintViolation();

}

**return** valido;

}

**private** **boolean** ambosSaoIguais(Object valorAtributo, Object valorAtributoConfirmacao) {

**return** valorAtributo != **null** && valorAtributo.equals(valorAtributoConfirmacao);

}

**private** **boolean** ambosSaoNull(Object valorAtributo, Object valorAtributoConfirmacao) {

**return** valorAtributo == **null** && valorAtributoConfirmacao == **null**;

}

}

**Convertendo data tipo String para LocalDate do java 8**

Colocar na classe WebConfig.class no método mvcConversionService

@Bean

**public** FormattingConversionService mvcConversionService (){

...

//API de Datas do Java 8 - para utilizar o tipo LocalDate

DateTimeFormatterRegistrar datatTimeFormatter = **new** DateTimeFormatterRegistrar();

datatTimeFormatter.setDateFormatter(DateTimeFormatter.*ofPattern*("dd/MM/yyyy"));

datatTimeFormatter.setTimeFormatter(DateTimeFormatter.*ofPattern*("HH:mm"));

datatTimeFormatter.registerFormatters(conversionService);

**....**

**return** conversionService;

**}**

**Componente de datas bootstrap-date-picker**

**Validações:**

-Adicionar mascara de datas em campo data, com mask “00/00/0000”

-Não permitir entrada de textos

-Tratamento de mensagem em caso de datas incompletas ex: 01/02/00

Criar no WebConfig.java o método messageSource(){...}

**Adicionar os javascript na app:**

bootstrap-datepicker.min.js

bootstrap-datepicker.pt-BR.min.js

Foi criado um objeto MaskDate com javascript para controlar o date-picker

**No HTML**

<div class=*"form-group col-sm-3 bw-required"*>

<label for=*"dataNascimento"* >Data de Nascimento</label>

<input id=*"dataNascimento"* type=*"text"* class=*"form-control js-date"*

th:field=*"\*{dataNascimento}"* autocomplete=*"off"* />

</div>

**No JAVASCRIPT**

Brewer.MaskDate = (**function**(){

**function** MaskDate(){

**this**.inputDate = $('.js-date');

}

MaskDate.prototype.enable = **function**(){

**this**.inputDate.mask('00/00/0000');

**this**.inputDate.datepicker({

orientation : 'bottom',

language : 'pt-BR',

highlight : **false**,

autoclose : **true**

});

}

**return** MaskDate;

}());

$(**function**(){

---

**var** maskDate = **new** Brewer.MaskDate();

maskDate.enable();

---

});

**Componente visual swtich do bootstrap**

**Status ativo ou inativo**

**Adicionar os CSS na app, ver cadastro de usuario:**

*bootstrap-switch.min.js*

*bootstrap-switch.min.css*

**Carregando dados do banco no Checbox**

**No HTML**

<div>

<th:block th:each=*"grupo : ${grupos}"*>

<div class=*"checkbox checkbox-inline"*>

<input type=*"checkbox"* th:value=*"${grupo.codigo}"* th:field=*"\*{grupos}"*>

<label th:for=*"${#ids.prev('grupos')}"* th:text=*"${grupo.nome}"*>Administrador</label>

</div>

</th:block>

</div>

**No controller**

mv.addObject("grupos", grupos.findAll());

**No CSS - Customizado componente chekbox CSS**

Para que a seleção do check box funcione pelo CSS do bootstrap é necessário vir após o ckecked um label, mas o thymeleaf coloca um input hidden, antes do label para seu controle, dai não funciona a seleção do checkBox, para arrumar foi sobreposto a classe checkbox CSS do bootstrap pela classe do checkbox do brewer.css

*.checkbox* **input**[type="checkbox"]*:checked* + **input** + **label***::after* {

font-family: *'FontAwesome'*;

content: *"\f00c"*;

}

**Validar lista no html com bean validation**

Utilizar- @Size( min =1 , message “Selecione”)

Ex:

@Size(min=1, message = "Selecione pelo menos um grupo")

@ManyToMany

@JoinTable(name = "usuario\_grupo", joinColumns = @JoinColumn(name = "codigo\_usuario")

, inverseJoinColumns = @JoinColumn(name = "codigo\_grupo"))

**private** List<Grupo> grupos;

**Autenticação e Autorização**

Será Utilizado o spring Security

**Autenticação:** Processo de verificação das credenciais, para saber se o usuário a senha estão corretos, se realmente você é quem você esta falando que é. São validados em Banco de dados.

**Autorização:** Processo que permite ao usuário do sistema fazer somente aquilo que ele esta autorizada a fazer ou acessar.

**Spring Security**

Adicionar no pom.xml

Spring-security-web e spring-security-config

Interfere na acentuação UTF-8 por causa dos filtros

Para o resolver o problema, colocar na classe SecurityInitializer o método:

@Override

**protected** **void** beforeSpringSecurityFilterChain(ServletContext servletContext) {

//controla a sessao independente do usuario estar usando ou não por tempo determinado em segundos

//servletContext.getSessionCookieConfig().setMaxAge(20);

// nao mostra o sessionID na url, passa a ser controlado via cookie

servletContext.setSessionTrackingModes(EnumSet.*of*(SessionTrackingMode.***COOKIE***));

FilterRegistration.Dynamic characterEncodingFilter = servletContext.addFilter("encodingFilter", **new** CharacterEncodingFilter());

characterEncodingFilter.setInitParameter("encoding","UTF-8");

characterEncodingFilter.setInitParameter("forceEncoding","true");

characterEncodingFilter.addMappingForUrlPatterns(**null**, **false**, "/\*");

}

E caso tenha colocado na classe **AppInitializer** no método **getServletFilter** remover a configuração do método

**- Abilitar o Spring Security**

- Criar classe **SecurityInitializer.java**

**public** **class** SecurityInitializer **extends** AbstractSecurityWebApplicationInitializer{

@Override

**protected** **void** beforeSpringSecurityFilterChain(ServletContext servletContext) {

//controla a sessao independente do usuario estar usando ou não por tempo determinado em segundos

//servletContext.getSessionCookieConfig().setMaxAge(20);

// nao mostra o sessionID na url, passa a controlado via cookie

servletContext.setSessionTrackingModes(EnumSet.*of*(SessionTrackingMode.***COOKIE***));

FilterRegistration.Dynamic characterEncodingFilter = servletContext.addFilter("encodingFilter"

, **new** CharacterEncodingFilter());

characterEncodingFilter.setInitParameter("encoding","UTF-8");

characterEncodingFilter.setInitParameter("forceEncoding","true");

characterEncodingFilter.addMappingForUrlPatterns(**null**, **false**, "/\*");

}

}

}

-Colocar a anotation **@EnabledWebSecurity** na classe **SecurityConfig.java** e extender a classe **WebSecurityConfigureAdapter.**

Agora sempre que for solicitada uma pagina, e o usuário não estiver logado, ele sera redirecionada para uma pagina de login padrão do spring.

**Para desabilitar o Spring Security ler comentários do método** configure **na classe** SecurityConfig ex. abaixo:

@Override

**protected** **void** configure(HttpSecurity http) **throws** Exception {

/\*\* Nota: Para usar a sessao "Dev" COMENTAR a sessao "Prod1" dessa classe, a sessao "A" no arquivo BarraNavegacao,

\* e a sessao "A csrf" no arquivo LayoutPadrao.html e descomentar essa sessao "Dev" dessa classe.

\*/

// [ Sessao Dev ] - Desabilita o Spring Security para desenvolvimento (sem tela de login)

// http.authorizeRequests().anyRequest().permitAll().and().csrf().csrfTokenRepository(new CookieCsrfTokenRepository()).disable();

/\*\* Nota: Para usar a sessao "Prod1" DESCOMENTAR a sessao "Prod1" dessa classe, a sessao "A" no arquivo BarraNavegacao.html,

\* e a sessao "A csrf" no arquivo LayoutPadrao.html e comentar a sessao DEV dessa classe.

\*/

// [ Sessao Prod1 ] - Habilitar o Spring Security para produção (com tela de login)

http

.authorizeRequests() // abaixo do authorizeRequests incluir urls de acesso com permissão não basta estar autenticado

.antMatchers("/cidades/nova").hasRole("CADASTRAR\_CIDADE")

.antMatchers("/usuarios/\*\*").hasRole("CADASTRAR\_USUARIO")

.anyRequest().authenticated() // daqui pra abaixo incluir urls de acesso somente a usuarios autenticado

.and()

.formLogin()

.loginPage("/login")

.permitAll()

.and()

.logout().logoutRequestMatcher(**new** AntPathRequestMatcher("/logout"))

.and()

.exceptionHandling().accessDeniedPage("/403")

.and()

.sessionManagement()

.invalidSessionUrl("/login");

// O codigo abaixo controla a quantidade de acesso que um

// usuario pode ter ao mesmo tempo

/\*\*

.sessionManagement()

.maximumSessions(1)

.expiredUrl("/login");

\*/

}

**Expirando a sessão do usuário**

Caso o usuraio não estiver mais usando a aplicação a sua sessão deve expirar, isso é controlado no arquivo **web.xml** com a seguinte configurações.

<web-app xmlns=*"http://xmlns.jcp.org/xml/ns/javaee"* xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*xsi:schemaLocation=*"http://xmlns.jcp.org/xml/ns/javaee http://xmlns.jcp.org/xml/ns/javaee/web-app\_3\_1.xsd"*version=*"3.1"*>

<session-config>

<session-timeout>60</session-timeout>

</session-config>

</web-app>

E na classe SecurityConfig no método configure, o código abaixo para gerenciamento de sessão, que redireciona o usuário para a pagina de login

.sessionManagement()

.invalidSessionUrl("/login"); // sessao invalida, redireciona para a pagina de login

**Exbindo o nome do usuário logado na barra de navegação**

Adicionar no pom.xml

thymeleaf-extras-springsecurity4 - fornece alguns recursos a mais para trabalhar com segurança no Spring

como o objeto *authentication e principal por exemplo*

No HTML

<li>

<!-- sessao A -->

<p class=*"navbar-text"* th:text=*"${#authentication.principal.usuario.nome}"*>Usuário</p>

</li>

No WebConfig

No método templateEngine

@Bean

**public** TemplateEngine templateEngine(){

engine.addDialect(**new** SpringSecurityDialect());

**return** engine;

}

**Extendendo a classe User do Spring Security**

Isso é necessario para poder acessar o nome do usuario e outros atributos, em vez de somente o email e senha do usuário logado veja os código abaixo

Na nova classe que extende a classe User do Spring

**public** **class** UsuarioSistema **extends** User {

**private** **static** **final** **long** ***serialVersionUID*** = 1L;

**private** Usuario usuario;

**public** UsuarioSistema(Usuario usuario, Collection<? **extends** GrantedAuthority> authorities){

**super**(usuario.getEmail(), usuario.getSenha(), authorities);

**this**.usuario = usuario; }

**public** Usuario getUsuario() {

**return** usuario;

}

**}**

Na classe AppUserDetailsService no retorno do método loadUserByUsername em vez de retornar User retorna UsuarioSistema que extende de User que implementa UserDetails

// validando login e senha do usuario

@Override

**public** UserDetails loadUserByUsername(String email) **throws** UsernameNotFoundException {

// verifica que o usuario e senha existe cadastrado no BD

Optional<Usuario> usuarioOptional = usuarios.porEmailEAtivo(email);

// caso nao exista sobe exception

Usuario usuario = usuarioOptional.orElseThrow(() -> **new** UsernameNotFoundException("Usuário e/ou senha incorretos"));

// retorna usuario e suas permissoes do BD

**return** **new** UsuarioSistema(usuario, getPermissoes(usuario));

}

**Escondendo componentes de tela de usuários sem permissão de acesso**

Adicionar no pom.xml

thymeleaf-extras-springsecurity4 - fornece alguns recursos a mais para trabalhar com segurança no Spring

**No HTML**

<div class=*"col-xs-2"* sec:authorize-url=*"/cidades/nova"*>

<div class=*"aw-page-header-controls"*>

<a class=*"btn btn-default"* th:href=*"@{/cidades/nova}"*>

<i class=*"glyphicon glyphicon-plus-sign"*></i>

<span class=*"hidden-xs hidden-sm "*>Nova cidade</span>

</a>

</div>

</div>

Também pode ser escondido o componente por permissão EX:

**NO HTML**

<td class=*"text-center"* sec:authorize=*"hasRole('ROLE\_CADASTRAR\_CIDADE')"*>

<a class=*"btn btn-link btn-xs"* title=*"Editar"*

th:href=*"@{/cidades/{codigo}(codigo=${cidade.codigo})}"*>

<i class=*"glyphicon glyphicon-pencil"*></i>

</a>

<a class=*"btn btn-link btn-xs js-exclusao-btn"* title=*"Excluir"* href=*"#"*

data:url=*"@{/cidades/{codigo}(codigo=${cidade.codigo})}"*

data:objeto=*"${cidade.nome}"*>

<i class=*"glyphicon glyphicon-remove"*></i>

</a>

</td>

**Nota:** Para esconder os componentes pode ser usado um outro que da na mesma.

**Permissão de acesso refinado**

**Pagina de erro padrão - LayoutSimples.html**

<!DOCTYPE html>

<html lang=*"pt"*

xmlns=*"http://www.w3.org/1999/xhtml"*

xmlns:th=*"http://www.thymeleaf.org"*

xmlns:layout=*"http://www.ultraq.net.nz/thymeleaf/layout"*>

<head>

<meta charset=*"UTF-8"*/>

<meta http-equiv=*"X-UA-Compatible"* content=*"IE=edge"*/>

<meta name=*"viewport"* content=*"width=device-width, initial-scale=1"*/>

<title>Brewer</title>

<link rel=*"stylesheet"* type=*"text/css"* th:href=*"@{/layout/stylesheets/vendors.min.css}"* />

<link rel=*"stylesheet"* type=*"text/css"* th:href=*"@{/layout/stylesheets/algaworks.min.css}"* />

<!-- HTML5 shim and Respond.js for IE8 support of HTML5 elements and media queries -->

<!-- WARNING: Respond.js doesn't work if you view the page via file:// -->

<!--[if lt IE 9]>

<script src="https://oss.maxcdn.com/html5shiv/3.7.2/html5shiv.min.js"></script>

<script src="https://oss.maxcdn.com/respond/1.4.2/respond.min.js"></script>

<![endif]-->

</head>

<body class=*"aw-layout-simple-page"*>

<div class=*"aw-layout-simple-page\_\_container"*>

<th:block layout:fragment=*"conteudo"*></th:block>

</div>

</body>

</html>

**Implementando Erro 404 - Pagina não existe**

**Criar 404.html herda de LayoutSimples.html**

<!DOCTYPE html>

<html lang=*"pt"*

xmlns=*"http://www.w3.org/1999/xhtml"*

xmlns:th=*"http://www.thymeleaf.org"*

xmlns:layout=*"http://www.ultraq.net.nz/thymeleaf/layout"*

layout:decorate=*"~{layout/LayoutSimples}"*>

<th:block layout:fragment=*"conteudo"*>

<div class=*"aw-error-panel"*>

<h1 class=*"aw-error-panel\_\_code"*>

<i class=*"fa fa-location-arrow"*></i>404

</h1>

<h2 class=*"aw-error-panel\_\_title"*>Página não encontrada</h2>

<div class=*"aw-error-panel\_\_description"*>

Desculpe, a página que você tentou acessar não foi encontrada ou não existe.

Verifique se a URL esta correta ou entre em contato com a gente se precisar de ajuda.

<br/><br/>

<a href=*"javascript:history.back()"* class=*"btn btn-primary"*>

Voltar para onde estava

</a>

</div>

</div>

</th:block>

</html>

**No Webconfig inserir o xml abaixo**

<error-page>

<error-code>404</error-code>

<location>/404</location>

</error-page>

**No controller – ErrosController**

Utilizado para maper a url /404 (pagina não encontrada)e /500 (Erro no Servidor ) definida no WebConfig.xml

@Controller

**public** **class** ErrosController {

@GetMapping("/404")

**public** String paginaNaoEncontrada(){

**return** "404";

}

@RequestMapping("/500")

**public** String erroServidor() {

**return** "500";

}

}

**Implementando Erro 403 - Pagina acesso negado**

**Herda de LayoutSimples.html**

<!DOCTYPE html>

<html lang=*"pt"*

xmlns=*"http://www.w3.org/1999/xhtml"*

xmlns:th=*"http://www.thymeleaf.org"*

xmlns:layout=*"http://www.ultraq.net.nz/thymeleaf/layout"*

layout:decorate=*"~{layout/LayoutSimples}"*>

<th:block layout:fragment=*"conteudo"*>

<div class=*"aw-error-panel"*>

<h1 class=*"aw-error-panel\_\_code"*>

<i class=*"fa fa-lock"*></i>

</h1>

<h2 class=*"aw-error-panel\_\_title"*>Acesso negado</h2>

<div class=*"aw-error-panel\_\_description"*>

Desculpe, você não está autorizado a acessar a página que solicitou.

Se você acha que isso é um engano, entre em contato com a gente.

<br/><br/>

<a href=*"javascript:history.back()"* class=*"btn btn-primary"*>

Voltar para onde estava

</a>

</div>

</div>

</th:block>

</html>

**Implementando Erro 500 - Erro no Servidor**

**Herda de LayoutSimples.html**

**No webconfig**

<error-page>

<error-code>500</error-code>

<location>/500</location>

**No html**

<!DOCTYPE html>

<html lang=*"pt"*

xmlns:th=*"http://www.thymeleaf.org"*

xmlns:layout=*"http://www.ultraq.net.nz/thymeleaf/layout"*

layout:decorate=*"~{layout/LayoutSimples}"*>

<th:block layout:fragment=*"conteudo"*>

<div class=*"aw-error-panel"*>

<h1 class=*"aw-error-panel\_\_code"*>

<i class=*"fa fa-exclamation-triangle"*></i> 500

</h1>

<h2 class=*"aw-error-panel\_\_title"*>Erro no servidor</h2>

<div class=*"aw-error-panel\_\_description"*>

Desculpe, algo muito estranho aconteceu e não conseguimos processar a sua requisição.

Tente de novo ou entre em contato com a gente se precisar de ajuda.

<br/><br/>

<a href=*"javascript:history.back()"* class=*"btn btn-primary"*>Voltar para onde estava</a>

</div>

</div>

</th:block>

</html>

**Pagina de login**

**Utilizar o layout de login da Algaworks layout-bootstrap-atacama-1.0.0**

Pagina **login.html** dentro da pasta dist, copiar todo conteúdo e colar dentro do arquivo **login.html** da aplicação, para mais detalhes ver pagina login já customizada que esta na aplicação Brewer, Css e javascritpt.

- A anotação @AuthenticationPrincipal representa o usuario logado

- Caso o usuário solicitar uma pagina e não tiver logado o spring security ira carregar a pagina de login, e após o login ira redirecionar o usuário para a pagina solicitada antes do login

- Criar classe de controle SegurancaController.java ex: abaixo

@Controller

**public** **class** SegurancaController {

@GetMapping("/login")

**public** String login(@AuthenticationPrincipal User user){

**if**(user != **null**){

**return** "redirect:/cervejas";

}

**return** "Login";

}

@GetMapping("/403")

**public** String acessoNegado(){

**return** "403";

}

}

**Carregando usuário e senha do banco**

Na classe SecurityConfig injetar a interface UserDetailsService e implementar o metodo configure abaixo :

@Autowired

UserDetailsService userDetailsService;

@Override

**protected** **void** configure(AuthenticationManagerBuilder auth) **throws** Exception {

//Codigo para Autenticacao padrao em memoria, usar usuario e senha admin,admin que estao

configurados

//auth.inMemoryAuthentication()

//.withUser("admin").password("admin").roles("CADASTRO\_CLIENTE");

// passwordEncoder é quem faz a criptografia da senha para verificacao

auth.userDetailsService(userDetailsService).passwordEncoder(passwordEncoder());

}

Criar a implementação da interface UserDetailsService na classe AppUserDetailsService para carregar usuário e senha do banco.

@Autowired

**private** Usuarios usuarios;

// validando login e senha do usuario

@Override

**public** UserDetails loadUserByUsername(String email) **throws** UsernameNotFoundException {

// verifica que o usuario e senha existe cadastrado no BD

Optional<Usuario> usuarioOptional = usuarios.porEmailEAtivo(email);

// caso nao exista sobe exception

Usuario usuario = usuarioOptional.orElseThrow(() -> **new**

UsernameNotFoundException("Usuário e/ou senha incorretos"));

// retorna usuario e suas permissoes

**return** **new** UsuarioSistema(usuario, getPermissoes(usuario));

}

Método que controla acessos dentro da classe SecurityConfig

@Override

**protected** **void** configure(HttpSecurity http) **throws** Exception {

/\*\* Nota: Para usar a sessao "Dev" COMENTAR a sessao "Prod1" dessa classe, a sessao "A" no arquivo BarraNavegacao,

\* e a sessao "A csrf" no arquivo LayoutPadrao.html e descomentar essa sessao "Dev" dessa classe.

\*/

// [ Sessao Dev ] - Desabilita o Spring Security para desenvolvimento (sem tela de login)

// http.authorizeRequests().anyRequest().permitAll().and().csrf().csrfTokenRepository(new CookieCsrfTokenRepository()).disable();

/\*\* Nota: Para usar a sessao "Prod1" DESCOMENTAR a sessao "Prod1" dessa classe, a sessao "A" no arquivo BarraNavegacao.html,

\* e a sessao "A csrf" no arquivo LayoutPadrao.html e comentar a sessao DEV dessa classe.

\*/

// [ Sessao Prod1 ] - Habilitar o Spring Security para produção (com tela de login)

http

.authorizeRequests() // abaixo do authorizeRequests incluir urls que precisam de permissao e autenticacao

.antMatchers("/cidades/nova").hasRole("CADASTRAR\_CIDADE") // acessa somente url /cidades/nova

.antMatchers("/usuarios/\*\*").hasRole("CADASTRAR\_USUARIO") // acessa todas pastas e arquivos da url usuario pra frnte

.anyRequest().authenticated() // daqui pra baixo acessa tudo se tiver autenticado, urls acima somente se tiver permissao

.and()

.formLogin()

.loginPage("/login") // url da pagina de login, caso não tenha o spring gera uma padrao

.permitAll() // qualquer um pode acessar a pagina de login

.and()

.logout().logoutRequestMatcher(**new** AntPathRequestMatcher("/logout"))

.and()

.exceptionHandling().accessDeniedPage("/403")

.and()

.sessionManagement()

.invalidSessionUrl("/login");

// O codigo abaixo controla a quantidade de acesso que um

// usuario pode ter ao mesmo tempo

/\*\*

.sessionManagement()

.maximumSessions(1)

.expiredUrl("/login");

\*/

}

.anyRequest().denyAll() bloqueia tudo se não tiver permissão

**Pagina de acesso negado**

Ver pagina 403.html, quando o usuário não tiver acesso a pagina sera redirecionado para essa pagina.

Deve ser mapeado no controller e informado a url no método **configure** da classe **SecurityConfig**

.exceptionHandling().accessDeniedPage("/403")

**SQL para ver as permissões que o usuário tem**

select u.email usuario

,group\_concat(substring(p.nome, 6) order by p.nome separator ', ') permissao

from usuario u

, usuario\_grupo ug

, grupo g

, grupo\_permissao gp

, permissao p

where ug.codigo\_usuario = u.codigo

and ug.codigo\_grupo = g.codigo

and g.codigo = gp.codigo\_grupo

and gp.codigo\_permissao = p.codigo

**CSRF – Site Request Forgery**

Falsificacao de uma requisição entre sites

Para tanto se obtem a url do site original e se cria uma pagina html falsa com o <form> apontando para a url do site que esta logado com o usuário correto, campos hidden e um botão submit. Ao clicar no botão submit o formulário é enviado.

**Confirgurando o csrf na aplicação**

No spring security csrf já vem abilitado como default

**No html**

th:action do thymeleaf no <form> ativa o csrf na pagina, após ativação do spring security todas as paginas deveram conter o th:action caso contrario não serão acessivei e aparecera a mensagem:

**HTTP Status 405 – Request method ‘POST’ not supported**

No Webconfig

Nada a fazer já vem com implementação padrão mas pode ser desabilitado

No LayoutPadrao.html

<!-- Sessao csrf - Trata segurança contra ataques csrf, th:value="${\_csrf.headerName}

durante o upload de fotos envia para o servidor o token X-CSRF-TOKEN via javascript cerveja.upload-foto.js -->

<input type=*"hidden"* name=*"\_csrf"* th:value=*"${\_csrf.token}"* />

<input type=*"hidden"* name=*"\_csrf\_header"* th:value=*"${\_csrf.headerName}"* />

No Webconfig para desabilitar

.csrf().csrfTokenRepository(new CookieCsrfTokenRepository()).disable()

No javascript cerveja.upload-foto.js

.....

**var** settings = {

type: 'json',

filelimit: 1, // quantidade de arquivos que sera enviados de uma vez

allow: '\*.(jpg|jpeg|png)', // tipos de arquivos que esta sendo enviado contentType

action: **this**.containerFotoCerveja.data('url-fotos'), // a url de action a ser enviada, vem do atributo data-url-fotos adicionada na div

complete: onUploadCompleto.bind(**this**), // o atributo complete, recebera o retorno que vira do servidor, .bind(this) da acesso a todas var. declarada na funcao contrutora

beforeSend: adicionarCsrfToken // antes de executar o envio do ajax

}

...

**function** adicionarCsrfToken(xhr){

**var** token = $('input[name=\_csrf]').val();

**var** header = $('input[name=\_csrf\_header').val();

xhr.setRequestHeader(header, token);

}

**Confirgurando o csrf na aplicação**

No spring security

**Menu lateral implementação**

É um framents de código html inserido LayoutPadrao.html

A classe **is-active** do bootstrap, que destaca o link do menu ao ser selecionado, sera inserido por um TagProcessor

No HTML

<!DOCTYPE html>

<html lang=*"pt"* xmlns=*"http://www.w3.org/1999/xhtml"*

xmlns:th=*"http://www.thymeleaf.org"*

xmlns:brewer=*"http://brewer.algaworks.com"*>

<body>

<aside class=*"aw-layout-sidebar js-sidebar"* th:fragment=*"layout-sidebar"*>

<div class=*"aw-layout-sidebar\_\_content"*>

<nav class=*"aw-menu js-menu"*>

<ul class=*"aw-menu\_\_list"*>

<li class=*"aw-menu\_\_item"* brewer:menu=*"@{/}"*>

<a th:href=*"@{/}"*> <i class=*"fa fa-fw fa-home"*></i><span>Dashboard</span>

</a>

</li>

<li class=*"aw-menu\_\_item"*>

<a href=*"#"*>

<i class=*"fa fa-fw fa-credit-card"*></i><span>Vendas</span>

<i class=*"aw-menu\_\_navigation-icon fa"*></i>

</a>

<ul class=*"aw-menu\_\_list aw-menu\_\_list--sublist"*>

<li class=*"aw-menu\_\_item aw-menu\_\_item--link"* brewer:menu=*"@{/vendas.+}"*>

<a th:href=*"@{/vendas/nova}"*>Cadastro de venda</a>

</li>

</ul>

<ul class=*"aw-menu\_\_list aw-menu\_\_list--sublist"*>

<li class=*"aw-menu\_\_item aw-menu\_\_item--link"* brewer:menu=*"@{/vendas}"*>

<a th:href=*"@{/vendas}"*>Pesquisa venda</a>

</li>

</ul>

</li>

<li class=*"aw-menu\_\_item"*>

<a href=*"#"*>

<i class=*"fa fa-fw fa-truck"*></i><span>Estoque</span>

<i class=*"aw-menu\_\_navigation-icon fa"*></i>

</a>

<ul class=*"aw-menu\_\_list aw-menu\_\_list--sublist"*>

<li class=*"aw-menu\_\_item aw-menu\_\_item--link"* brewer:menu=*"@{/estilos.\*}"*>

<a th:href=*"@{/estilos}"*>Estilos</a>

</li>

<li class=*"aw-menu\_\_item aw-menu\_\_item--link"* brewer:menu=*"@{/cervejas.\*}"*>

<a th:href=*"@{/cervejas}"*>Cervejas</a>

</li>

</ul>

</li>

<li class=*"aw-menu\_\_item"*>

<a href=*"#"*>

<i class=*"fa fa-fw fa-database"*></i><span>Cadastros</span>

<i class=*"aw-menu\_\_navigation-icon fa"*></i>

</a>

<ul class=*"aw-menu\_\_list aw-menu\_\_list--sublist"*>

<li class=*"aw-menu\_\_item aw-menu\_\_item--link"* brewer:menu=*"@{/cidades.\*}"*>

<a th:href=*"@{/cidades}"*>Cidades</a>

</li>

<li class=*"aw-menu\_\_item aw-menu\_\_item--link"* brewer:menu=*"@{/clientes.\*}"*>

<a th:href=*"@{/clientes}"*>Clientes</a>

</li>

<li class=*"aw-menu\_\_item aw-menu\_\_item--link"* brewer:menu=*"@{/usuarios.\*}"*>

<a th:href=*"@{/usuarios/novo}"*>Usuário</a>

</li>

</ul>

</li>

<li class=*"aw-menu\_\_item"*>

<a href=*"#"*>

<i class=*"fa fa-fw fa-file-pdf-o"*></i><span>Relatórios</span>

<i class=*"aw-menu\_\_navigation-icon fa"*></i>

</a>

<ul class=*"aw-menu\_\_list aw-menu\_\_list--sublist"*>

<li class=*"aw-menu\_\_item aw-menu\_\_item--link"* brewer:menu=*"@{/relatorios.\*}"*>

<a th:href=*"@{/relatorios/vendasEmitidas}"*>Vendas emitidas</a>

</li>

</ul>

</li>

</ul>

</nav>

</div>

</aside>

</body>

</html>

**Na classe MenuAttributeTagProcessor tagprocessor**

**public** **class** MenuAttributeTagProcessor **extends** AbstractAttributeTagProcessor {

**private** **static** **final** String ***NOME\_ATRIBUTO*** = "menu";

**private** **static** **final** **int** ***PRECEDENCIA*** = 100;

**public** MenuAttributeTagProcessor(String dialectPrefix) {

**super**(TemplateMode.***HTML***, dialectPrefix, **null**, **false**, ***NOME\_ATRIBUTO***, **true**,

***PRECEDENCIA***, **true**);

}

@Override

**protected** **void** doProcess(ITemplateContext context, IProcessableElementTag tag,

AttributeName attributeName,

String attributeValue , IElementTagStructureHandler structureHandler) {

IEngineConfiguration configuration = context.getConfiguration();

IStandardExpressionParser parser =

StandardExpressions.*getExpressionParser*(configuration);

IStandardExpression expression = parser.parseExpression(context, attributeValue);

String menu = (String) expression.execute(context);

HttpServletRequest request = ((IWebContext) context).getRequest();

String uri = request.getRequestURI();

// a uri começa com o nome do menu

**if**(uri.matches(menu)){

String classesExistentes = tag.getAttributeValue("class");

// insere a classe is-active no item <li> do menu selecionado

structureHandler.setAttribute("class", classesExistentes + " is-active");

}

}

}

**Na classe BrewerDialect**

**public** **class** BrewerDialect **extends** AbstractProcessorDialect{

**public** BrewerDialect() {

**super**("AlgaWorks Brewer", "brewer", StandardDialect.***PROCESSOR\_PRECEDENCE***);

}

@Override

**public** Set<IProcessor> getProcessors(String dialectPrefix) {

**final** Set<IProcessor> processadores = **new** HashSet<>();

processadores.add(**new** MenuAttributeTagProcessor(dialectPrefix));

**return** processadores;

}

}

**SQL Consulta em tabelas de relacionamento manyToMany com Criteria**

**Na Entidade Usuario (principal)**

@Size(min=1, message = "Selecione pelo menos um grupo")

@ManyToMany // criar um metodo no UsuarioQueries e inicializar quando precisar

@JoinTable(name = "usuario\_grupo", joinColumns = @JoinColumn(name = "codigo\_usuario")

, inverseJoinColumns = @JoinColumn(name = "codigo\_grupo"))

**private** List<Grupo> grupos;

**Na Entidade Grupo (entidade Relacionadas)**

Não tem, pois o relacionamento é unidirecional

Parte de Usuarios para Grupo 1..\* (um usuário pode pertencer a muitos grupos**)**

**Na Entidade UsuarioGrupo ( entidade OneToMany)**

@Entity

@Table(name="usuario\_grupo")

**public** **class** UsuarioGrupo {

@EmbeddedId

**private** UsuarioGrupoId id;

...

get e set de usuarioGrupoId

...

hashCode e equals de usuarioGrupoId

}

**Na Classe UsuarioGrupoId**

A Classe **UsuarioGrupoId** é utilizada para criar o **Id** da entidade **UsuarioGrupo.**

A classe **UsuarioGrupoId** faz o relacionamento ManyToMany entre as tabelas **usuario** e **grupo** do BD com a tabela **usuario\_grupo,**  formando a chave composta (id da tabela UsuarioGrupo).

@Embeddable

**public** **class** UsuarioGrupoId **implements** Serializable {

**private** **static** **final** **long** ***serialVersionUID*** = 1L;

@ManyToOne

@JoinColumn(name = "codigo\_usuario")

**private** Usuario usuario;

@ManyToOne

@JoinColumn(name = "codigo\_grupo")

**private** Grupo grupo;

...

get e set de usuario e grupo

...

hashCode e equals de usuario e grupo

}

**Na classe UsuariosImpl método usado para filtrar os grupos usando Criteria do Hibernate**

**private** **void** adicionarFiltro(UsuarioFilter filtro, Criteria criteria) {

**if** (filtro != **null**) {

**if** (!StringUtils.*isEmpty*(filtro.getNome())) {

criteria.add(Restrictions.*ilike*("nome", filtro.getNome(), MatchMode.***ANYWHERE***));

}

**if** (!StringUtils.*isEmpty*(filtro.getEmail())) {

criteria.add(Restrictions.*ilike*("email", filtro.getEmail(), MatchMode.***START***));

}

//criteria.createAlias("grupos", "g", JoinType.LEFT\_OUTER\_JOIN);

**if** (filtro.getGrupos() != **null** && !filtro.getGrupos().isEmpty()) {

// lista que funcionara como a clasula and na consulta sql

List<Criterion> subqueries = **new** ArrayList<>();

// retorna para codigoGrupo o codiogo de cada grupo selecionado na tela

**for** (Long codigoGrupo : filtro.getGrupos().stream().mapToLong(Grupo::getCodigo).toArray()) {

// DetachedCriteria - faz uma consulta separada um "subselect" da criteria principal para a classe UsuarioGrupo

DetachedCriteria dc = DetachedCriteria.*forClass*(UsuarioGrupo.**class**);

dc.add(Restrictions.*eq*("id.grupo.codigo", codigoGrupo)); // filtra pelo codigo do grupo

dc.setProjection(Projections.*property*("id.usuario")); // Projections - retorna o codigo do usuario

subqueries.add(Subqueries.*propertyIn*("codigo", dc));

// a cada loop retorna o resultado do subselect

// u.codigo in (codigo) e adiciona na lista de subqueries

}

// cria um array de criterios, do tamanho da lista de subqueries

Criterion[] criterions = **new** Criterion[subqueries.size()];

// passando o resultado das subqueries para a queries principal - ex

criteria.add(Restrictions.*and*(subqueries.toArray(criterions))); // transforma subqueries em array, e aplica and nos resultados da subqueries

}

}

}

**Consulta SQL de usuario\_grupo implementada com Criteria**

Select \*

From usuario u

Inner join usuario\_grupo ug on u.codigo = ug.codigo\_usuario

Inner join grupo g on ug.codigo\_grupo = g.codigo

Where (

u.codigo in (select codigo\_usuario from usuario\_grupo where codigo\_grupo = 1)

and u.codigo in (select codigo\_usuario from usuario\_grupo where codigo\_grupo = 2))

**Seleção de 1 registro, todos ou mais de em**

**Em tabelas de dados**

**No html – Botões para Selecionar e Deselecionar Todos registros**

<div class=*"aw-datatable-toolbar"*>

<!-- data:url permite enviar uma url especifica, url informada em data:url pode ser acessada via javascript -->

<button class=*"btn btn-default btn-xs js-status-btn disabled"* data:status=*'ATIVAR'* data:url=*"@{/usuarios/status}"*>

<span>Ativar</span>

</button>

<button class=*"btn btn-default btn-xs js-status-btn disabled"* data:status=*'DESATIVAR'* data:url=*"@{/usuarios/status}"*>

<span>Desativar</span>

</button>

</div>

**No html – checkbox que seleciona todos checkbox**

<thead>

<tr>

<!-- primeira coluna checkbox de selecao seleciona todos registros-->

<th class=*"table-usuarios-col-status"*>

<div class=*"checkbox aw-checkbox-no-margin"*>

<input type=*"checkbox"* class=*"js-selecao-todos"* >

<label></label>

</div>

</th>

<th>Nome</th>

<th>E-mail</th>

<th>Grupo</th>

<th>Status</th>

<th class=*"table-col-acoes"*></th>

</tr>

</thead>

* **Javascript de seleção de todos checkbox**

**function** onSelecaoClicado(){

**var** selecaoCheckboxChecados = **this**.selecaoCheckbox.filter(':checked'); // filtra os componentes e retorna os componentes selecionados com filter

// se quantidade de checkbox selecionados for maior igual que a quantidade de checkbox que existem, seleciona checkbox selecionar todos

**this**.selecaoTodosCheckBox.prop('checked', selecaoCheckboxChecados.length >= **this**.selecaoCheckbox.length);

// desabilita os botoes Ativar/Desativar caso nenhum checkbox estiver selecionado

statusBotaoAcao.call(**this**, selecaoCheckboxChecados.length);

}

**Javascript desabilita botões ativar/desativar**

**function** statusBotaoAcao(ativar){

// remove a classe disabled dos botões caso algum checkbox esteja selecionado

ativar ? **this**.statusBtn.removeClass('disabled') : **this**.statusBtn.addClass('disabled');

}

**Não redenrizar o div (chebox) do usuário logado**

**No html**

<!-- th:if - nao rendereiza a selecao de ativar/desativar do proprio usuario logado (principal) -->

<div class=*"checkbox aw-checkbox-no-margin"* th:if=*"${#authentication.principal.usuario.codigo != usuario.codigo}"*>

<input type=*"checkbox"* class=*"js-selecao"* data:codigo=*"${usuario.codigo}"*>

<label></label>

</div>

</td>

**No HTML**

Colocando destaque no status do usuário

<td>

<th:block th:if=*"${usuario.ativo}"*>

<span class=*"label label-success"*>Ativo</span>

</th:block>

<th:block th:if=*"${not usuario.ativo}"*>

<span class=*"label label-danger"*>Inativo</span>

</th:block>

</td>

**No CSS**

Alinhando colunas do checkbox

*.table-usuarios-col-status* {

width: *30px*;

text-align: *center*;

}

**No controller**

/\*\* **@PutMapping** - Atualização de objetos. Aguarda retorno de uma view. Como o metodo não retornara uma view. Apos a execução **@ResponseStatus**(HttpStatus.OK) e utilizado para retornar codigo http para browser "codigo 200". A string status sera convertida automanticamento pelo spring para o Enum StatusUsuario \*/

@PutMapping("/status")

@ResponseStatus(HttpStatus.***OK***)

**public** **void** atualizarStatus(@RequestParam("codigos[]") Long[] codigos, @RequestParam("status")

StatusUsuario statusUsuario){

//Arrays.asList(codigos).forEach(System.out::println ); // print de array com java 8

cadastroUsuarioService.alterarStatus(codigos, statusUsuario);

}

**No Service – UsuarioService**

/\*\* Metodo chamado por outro metodo PUT do controller, que executara um UPDATE

automaticamente no banco de dados, sem ser solicitado explicitamente um Update em JPQL ou

SQL Nativa\*/

@Transactional

**public** **void** alterarStatus(Long[] codigos, StatusUsuario statusUsuario) {

statusUsuario.executar(codigos, usuarios);

}

**No Enum - StatusUsuario**

Empregado para adicionar novas regras, por exemplo caso outros status vem ser criados basta implementar aqui, e tudo já vai estar funcionando

**public** **enum** StatusUsuario {

***ATIVAR*** {

@Override

**public** **void** executar(Long[] codigos, Usuarios usuarios) {

usuarios.findByCodigoIn(codigos).forEach(u -> u.setAtivo(**true**));

}

},

***DESATIVAR*** {

@Override

**public** **void** executar(Long[] codigos, Usuarios usuarios) {

usuarios.findByCodigoIn(codigos).forEach(u -> u.setAtivo(**false**));

}

};

**public** **abstract** **void** executar(Long[] codigos, Usuarios usuarios);

}

**Na Classe AppInitializer**

/\*\* Nesse metodo pode ser implementado filtros de encoding para forcar UTF-8

\* HttpPutFormContentFilter - foi usado para permitir enviar parametros via PUT com Ajax ao controller \*/

@Override

**protected** Filter[] getServletFilters() {

HttpPutFormContentFilter httpPutFormContentFilter = **new** HttpPutFormContentFilter();

**return** **new** Filter[] { httpPutFormContentFilter };

}

**Paginação tela de pesquisa do Usuario**

**No controller**

**Na classe UsuariosImpl**

**Http Session – Escopo de sessão**

HTTP é stateless – não mantem estado, quem controla a sessão é servidor e não o protocolo HTTP. O servidor controla a sessão criando um objeto session\_id (Identificador de sessão) apartir da primeira requisição, e passa a enviar para o brower, que vai sempre retornar ao fazer novas requisições identificando assim a sessão do usuário. Não necessário criar, buscar, editar, o Spring já faz abstração do session\_id.

**JUnit Testes automatizados – Testes unitários**

**Enviando email**

Usando **JavaMailSender** e **Thymeleaf**

Antes do email ser enviado seu html sera processado pelo tymeleaf, para inserção dos dados no email.

Os dados são passados para o html através de acesso as variáveis inseridas no contexto ex:

Context context = **new** Context(**new** Locale("pt","BR"));

context.setVariable("venda", venda);

context.setVariable("logo","logo");

e serão recuperadas no html ex:

<p style="text-align: *right*">

<img src=*"../../static/images/logo-gray-brewer.png"* th:src=*"|cid:${logo}|"*>

</p>

<h1 style="font-size: *22px*" th:text=*"|Olá ${venda.cliente.nome}|"*>Olá Maria!</h1>

**Objeto do tymeleaf**

#numbers.formatDecimal – formata um numero passado no primeiro parâmetro, no segundo é informado o numero de dígitos da parte inteira, e no terceiro o numero de dígitos da casa decimais Ex:

<td style="width: *120px*; text-align: *right*" th:text=*"|R$${#numbers.formatDecimal(venda.valorTotalItens, 1, 2)}|"*>R$1.200,50</td>

**Objeto do tymeleaf** #objects.nullSafe

#objects.nullSafe – caso o primeiro parâmetro retornar null, sera utilizado no retorno o segundo parâmetro ex:

<td style="width: *100px*; text-align: *right*"><span th:text=*"|(R$${#objects.nullSafe(#numbers.formatDecimal(venda.valorDesconto, 1, 2), '0,00')})|"*>(R$5,00)</span></td>

**No HTML ResumoVenda.html - Template do email**

<!DOCTYPE html>

<html xmlns=*"http://www.w3.org/1999/xhtml"*

xmlns:th=*"http://www.thymeleaf.org"*>

<body style="font-family: *Arial*; font-size: *14px*;">

<p style="text-align: *right*">

<img src=*"../../static/images/logo-gray-brewer.png"* th:src=*"|cid:${logo}|"*>

</p>

<h1 style="font-size: *22px*" th:text=*"|Olá ${venda.cliente.nome}|"*>Olá Maria!</h1>

<p>

Estamos enviando para você a relação de itens da sua compra em nossa loja de cervejas.

</p>

<table style="width: *100%*; border-collapse: *collapse*; margin: *20px 0*">

<th:block th:each=*"item : ${venda.itens}"*>

<tr style="border-bottom: *1px solid #ccc*; border-top: *1px solid #ccc*">

<td style="width: *50px*; padding-bottom: *10px*; padding-top: *10px*">

<img style="width: *40px*" src=*"../../static/images/cerveja-mock.png"* th:src=*"|cid:${mockCerveja}|"* th:unless=*"${item.cerveja.temFoto()}"* />

<img style="width: *40px*" src=*"../../static/images/cerveja-mock.png"* th:src=*"|cid:foto-${item.cerveja.codigo}|"* th:if=*"${item.cerveja.temFoto()}"* />

</td>

<td style="vertical-align: *top*; padding-bottom: *10px*; padding-top: *10px*">

<div style="display: *block*; font-weight: *bold*; font-size: *16px*" th:text=*"${item.cerveja.nome}"*>Bhrama Lata 350 ml</div>

<div style="color: *#8e8e8e*" th:text=*"${item.cerveja.origem.descricao}"*>Nacional</div>

<!-- numbers.formatDecimal(item.valorUnitario, 1, 2) <<<< esse codigo "valorUnitario" numero a ser formatado, (1) numero de digitos da parte inteira, (2) numero de digitos da parte decimal-->

<div style="color: *#8e8e8e*" th:text=*"|${item.quantidade} x R$${#numbers.formatDecimal(item.valorUnitario, 1, 2)}|"*>1 x R$1,90</div>

</td>

<td style="vertical-align: *top*; text-align: *right*; padding-bottom: *10px*; padding-top: *10px*">

<span style="display: *block*; font-weight: *bold*; font-size: *16px*" th:text=*"|R$${#numbers.formatDecimal(item.valorTotal, 1, 2)}|"*>R$1,90</span>

</td>

</tr>

<th:block th:if=*"${false}"*>

<tr style="border-bottom: *1px solid #ccc*; border-top: *1px solid #ccc*">

<td style="width: *50px*; padding-bottom: *10px*; padding-top: *10px*">

<img style="width: *40px*" src=*"../../static/images/cerveja-mock.png"* />

</td>

<td style="vertical-align: *top*; padding-bottom: *10px*; padding-top: *10px*">

<div style="display: *block*; font-weight: *bold*; font-size: *16px*">Heineken</div>

<div style="color: *#8e8e8e*">Internacional</div>

<div style="color: *#8e8e8e*">450 x R$3,93</div>

</td>

<td style="vertical-align: *top*; text-align: *right*; padding-bottom: *10px*; padding-top: *10px*">

<span style="display: *block*; font-weight: *bold*; font-size: *16px*">R$1.768,50</span>

</td>

</tr>

</th:block>

</th:block>

</table>

<p style="font-size: *12px*">

\* O status da sua compra é: <strong style="color: *red*"><span th:text=*"${venda.status.descricao}"*>orçamento</span></strong>.

</p>

<table style="width: *100%*;">

<tr>

<td style="text-align: *right*; font-weight: *bold*" >Subtotal:</td>

<td style="width: *120px*; text-align: *right*" th:text=*"|R$${#numbers.formatDecimal(venda.valorTotalItens, 1, 2)}|"*>R$1.200,50</td>

</tr>

<tr>

<td style="text-align: *right*; font-weight: *bold*">Frete:</td>

<td style="width: *100px*; text-align: *right*">

<span th:text=*"|R$${#objects.nullSafe(#numbers.formatDecimal(venda.valorFrete, 1, 2), '0,00')}|"*>R$8,00</span>

</td>

</tr>

<tr>

<td style="text-align: *right*; font-weight: *bold*">Desconto:</td>

<td style="width: *100px*; text-align: *right*"><span th:text=*"|(R$${#objects.nullSafe(#numbers.formatDecimal(venda.valorDesconto, 1, 2), '0,00')})|"*>(R$5,00)</span></td>

</tr>

<tr>

<td style="text-align: *right*; font-weight: *bold*; font-size: *16px*">Total final:</td>

<td style="width: *100px*; text-align: *right*; font-weight: *bold*; font-size: *16px*">

<span th:text=*"|R$${#numbers.formatDecimal(venda.ValorTotal, 1, 2)}|"*>R$1.200,50</span>

</td>

</tr>

</table>

<p>Obrigado.</p>

<p>Equipe Brewer!</p>

</body>

</html>

**No javascript**

**Na classe Mailer.java**

**Classe que controla o preenchimento e envio do email**

**No html – cadastroCerveja – Exclusão generica**

Exibindo tooltip classe javascritp de exibição de mensagem em links ex:

<a class=*"btn btn-link btn-xs js-tooltip"* title=*"Editar"*

th:href=*"@{/cervejas/{codigo}(codigo=${cerveja.codigo})}"*>

<i class=*"glyphicon glyphicon-pencil"*></i>

</a>

Enviando parâmetros para o controller através de link ex:

th:href=*"@{/cervejas/{codigo}(codigo=${cerveja.codigo})}"*

Mesmo exemplo mas enviando via atributo do html5, que sera pego no js para enviar via ajax

data:url=*"@{/cervejas/{codigo}(codigo=${cerveja.codigo})}"*

**No Javascript brewer.dialogo-excluir – exclusao**

No alert de exclusão sera usado a biblioteca - T4t5.github.io/sweetalert

**Na classe CervejaController - exclusao**

// @ResponseBody permite retornar mensagem de erro ou de sucesso ao js

@DeleteMapping("/{codigo}")

**public** @ResponseBody ResponseEntity<?> excluir(@PathVariable("codigo") Cerveja cerveja){

**try** {

cadastroCervejaService.excluir(cerveja);

} **catch** (ImpossivelExcluirEntidadeException e) {

// retorna msg para browser que sera tratada no js

**return** ResponseEntity.*badRequest*().body(e.getMessage());

}

**return** ResponseEntity.*ok*().build(); //retorna 200 ao cliente browser - de tudo ocorreu como esperado

}

**Na classe cadastroCervejaService - exclusão**

@Transactional

**public** **void** excluir(Cerveja cerveja){

**try**{

String foto = cerveja.getFoto();

cervejas.delete(cerveja);

cervejas.flush(); // executa a exclusao no banco

fotoStorage.excluir(foto); // apagando a foto e thumbnail da foto

}**catch**(PersistenceException e){

**throw** **new** ImpossivelExcluirEntidadeException("Impossível apagar cerveja. Esta sendo usada em alguma venda.");

}

}

**Edição de registro**

**Na classe cadastroCervejaService - edição**

**No Html cadastroCerveja - edição**

Alteração para permitir a alteração de dados, usando a mesma tela de cadastro.

No titulo – Tornando mais amigavel

<div class=*"col-xs-10"*>

<h1 th:if=*"${cerveja.nova}"*>Cadastro de cerveja</h1>

<h1 th:unless=*"${cerveja.nova}"* th:text=*"|Edição da ${cerveja.nome}|"*>Cadastro de cerveja</h1>

</div>

No form

<form method=*"POST"* th:object=*"${cerveja}"* class=*"form-vertical js-form-loading"*

th:action=*"${cerveja.nova} ? @{/cervejas/nova} : @{/cervejas/{codigo}(codigo=${cerveja.codigo})}"* >

<input type=*"hidden"* th:field=*"\*{codigo}"* />

No link

<a class=*"btn btn-link btn-xs js-tooltip"* title=*"Editar"*

th:href=*"@{/cervejas/{codigo}(codigo=${cerveja.codigo})}"*>

<i class=*"glyphicon glyphicon-pencil"*></i>

</a>

**Na classe CervejaController - edição**

Carregando objeto na tela para edição, chama o método nova para auxiliar na exibição dos dados

@GetMapping("/{codigo}")

**public** ModelAndView editar(@PathVariable("codigo") Cerveja cerveja){

ModelAndView mv = nova(cerveja);

mv.addObject(cerveja);

**return** mv;

}

Método Nova – método auxiliar na edição

@RequestMapping("/nova")

**public** ModelAndView nova(Cerveja cerveja){

ModelAndView mv = **new** ModelAndView("cerveja/CadastroCerveja");

mv.addObject("sabores", Sabor.*values*());

mv.addObject("estilos", estilos.findAll());

mv.addObject("origens", Origem.*values*());

**return** mv;

}

Salvando objeto editado ou Novo

/\*\* Esse metodo trata requisicao de POST para mais de uma url.

\* a expressao regular "{\\d+}" permite qualquer digito na url ex: "cervejas/3"

\* alem da url "/nova"

\*/

@PostMapping(value = { "/nova", "{\\d+}" })

**public** ModelAndView salvar(@Valid Cerveja cerveja, BindingResult result, RedirectAttributes attributes){

**if**(result.hasErrors()){

**return** nova(cerveja); //coloca o objeto vindo da view na requisicao

}

cadastroCervejaService.salvar(cerveja);

attributes.addFlashAttribute("mensagem", "Cerveja salva com sucesso!"); // injeta msg pra view

**return** **new** ModelAndView("redirect:/cervejas/nova"); // carrega uma nova pagina com uma nova requisição

}

**Controle do Evento**

**Na classe CervejaController - edição**

Carregando objeto na tela para edição, chama o método nova para

**Usuario - Edição**

**Na classe UsuarioController - edição**

@GetMapping("/{codigo}")

**public** ModelAndView editar(@PathVariable Long codigo){

Usuario usuario = usuarios.buscarComGrupos(codigo); // Busca usuarios no BD com grupos

ModelAndView mv = novo(usuario); // chama tela de cadastro do usuario, preenchida com os dados do usuario, selecionado na pesquisa para edicao

mv.addObject(usuario);

**return** mv;

}

**Na classe Usuario - edição**

Relacionamentos @ManyToMany por padrão não são inicializados,

para este tipo de relacionamento, a melhor pratica é cria um metodo na UsuarioQueries e implementar no UsuariosImpl para inicializar o grupo quando precisar

@Size(min=1, message = "Selecione pelo menos um grupo")

@ManyToMany

@JoinTable(name = "usuario\_grupo", joinColumns = @JoinColumn(name = "codigo\_usuario")

, inverseJoinColumns = @JoinColumn(name = "codigo\_grupo"))

**private** List<Grupo> grupos;

**Na classe UsuariosImpl - edição**

@Transactional(readOnly = **true**) // readOnly informa que é somente pesquisa

@Override

**public** Usuario buscarComGrupos(Long codigo) {

Criteria criteria = manager.unwrap(Session.**class**).createCriteria(Usuario.**class**);

criteria.createAlias("grupos", "g", JoinType.***LEFT\_OUTER\_JOIN***); // Fazendo relacionamento com a tabela grupos

criteria.add(Restrictions.*eq*("codigo", codigo));

criteria.setResultTransformer(Criteria.***DISTINCT\_ROOT\_ENTITY***); // Agrupa por usuario e remove

**return** (Usuario) criteria.uniqueResult();

}

**Na classe CadastroUsuarioService - edição**

@Transactional

**public** **void** salvar(Usuario usuario){

Optional<Usuario> usuarioExistente = usuarios.findByEmail(usuario.getEmail());

// O email do usuario ja existe cadastrado e o usuario e diferente, Então envia msg com exception

**if**(usuarioExistente.isPresent() && !usuarioExistente.get().equals(usuario)){

**throw** **new** EmailJaCadastradoException("E-mail já cadastrado");

}

**if**(usuario.isNovo() && StringUtils.*isEmpty*(usuario.getSenha())){

**throw** **new** SenhaObrigatoriaUsuarioException("Senha é obrigatória para novo usuário");

}

// Nota: O "campo senha" na edição sempre vira em branco na tela (Seguranca do Spring Security),

// Se novo Usuario ou ja existente e que tenha alterado a senha, informado uma nova senha

// entao criptografa a nova senha informada e seta no cadastro do usuario

**if**(usuario.isNovo() || !StringUtils.*isEmpty*(usuario.getSenha())){

// criptogrando nova senha para novos usuarios ou ja cadastrados que alteraram a senha

usuario.setSenha(**this**.passwordEncoder.encode(usuario.getSenha()));

// caso contrario se Usuario já cadastrado, e não alterou a senha

// seta a mesma senha do usuario recuperada do banco, ja criptografada

}**else** **if**(StringUtils.*isEmpty*(usuario.getSenha())) {

usuario.setSenha(usuarioExistente.get().getSenha());

}

usuario.setConfirmacaoSenha(usuario.getSenha());

/\*\* Se o usuario não e novo e esta sendo editado, ele nao podera alterar seu proprio status

(nesse caso o status não e exibido na tela). Para evitar erro de nullPointer, ao salvar,

seta o campo status com o status do usuario retornado do banco \*/

**if**(!usuario.isNovo() && usuario.getAtivo() == **null**){

usuario.setAtivo(usuarioExistente.get().getAtivo());

}

usuarios.save(usuario);

}

**Na classe VendaController - edição**

Na classe VendaController ao recuperar os itens da venda para edição deve-se setar os itens da venda na tabela de itens, juntamente com o uuid da venda, caso o uuid da venda não exista deve ser gerado um.

/\*\*Gera um uuid de controle de sessão de visao do usuario, ou seja para cada nova aba aberta o usuario tera

um identificador de sessao \*/

**private** **void** setUuid(Venda venda) {

// se um uuid de sessao do usuario não foi criado quando da criacao ou edicao de uma venda

**if**(StringUtils.*isEmpty*(venda.getUuid())){

// entao gera aleatoreamente um uuid usando API UUID do javaUtil e seta na venda

venda.setUuid(UUID.*randomUUID*().toString()); //

}

}

**Na classe CadastroVendaService - edição**

Venda - edição

@Transactional

**public** Venda salvar(Venda venda){

**if**(venda.isSalvarProibido()){

**throw** **new** RuntimeException("Usuário tentando salvar uma venda proibida");

}

// se for uma nova venda

**if**(venda.isNova()){

// seta data e hora atual da venda

venda.setDataCriacao(LocalDateTime.*now*());

// se estiver editanto usa a data de criacao da venda recuperada do banco

}**else**{

Venda vendaExistente = vendas.findOne(venda.getCodigo());

venda.setDataCriacao(vendaExistente.getDataCriacao());

}

**if**(venda.getDataEntrega() != **null**){

// formando a DataHoraEntrega, caso não tenha horario entrega, passe o horario do sistema 12:00

venda.setDataHoraEntrega(LocalDateTime.*of*(venda.getDataEntrega()

, venda.getHorarioEntrega() != **null** ? venda.getHorarioEntrega() : LocalTime.***NOON***));

}

**return** vendas.saveAndFlush(venda);

}

**Na classe Venda - edição**

Evitando a duplicação de registros na classe ItemVenda - edição

/\*\* orphanRemoval = true, para quando da edição ou exclusao de registros,

O hibernate remover os itens orfão da lista e também do banco. caso contrario sera

duplicado e inserido uma nova lista com os mesmos registros no banco

CascadeType.ALL para que o JPA ao salvar uma venda salve tambem os itens da venda\*/

@OneToMany(mappedBy = "venda", cascade = CascadeType.***ALL***, orphanRemoval = **true**)

**private** List<ItemVenda> itens = **new** ArrayList<>();

**Na classe Venda - Tempo decorrido desde a criacao da venda**

**public** Long getDiasCriacao(){

LocalDate inicio = dataCriacao != **null** ? dataCriacao.toLocalDate() : LocalDate.*now*();

// ChronoUnit - conta quantos dias tem entre as datas inicio e a data atual do sitema

**return** ChronoUnit.***DAYS***.between(inicio, LocalDate.*now*());

}

**No HTML CadastroVenda - Tempo decorrido desde a criacao da venda**

<div class=*"col-sm-4"*>

<div class=*"aw-box"*>

<div class=*"aw-box\_\_icon"*>

<i class=*"fa fa-calendar fa-3x"*></i>

</div>

<div class=*"aw-box\_\_value"*>

<span class=*"js-tooltip"* th:if=*"${venda.diasCriacao == 0}"*>Hoje</span>

<span class=*"js-tooltip"* th:if=*"${venda.diasCriacao == 1}"*>Há 1 dia</span>

<span class=*"js-tooltip"* th:if=*"${venda.diasCriacao > 1 }"*>Há [[${venda.diasCriacao}]] dias</span>

</div>

<div class=*"aw-box\_\_title"*>Criação</div>

</div>

</div>

**No .HTML CadastroVenda – Cancelamento da venda**

No corpo do form

<input type=*"hidden"* th:field=*"\*{usuario}"*/>

No botao cancelar – exibe botão somente se for uma edicao

<button class=*"btn btn-danger js-submit-btn"* data:acao=*"cancelar"* th:unless=*"${venda.nova}"*>Cancelar</button>

No botao de pesquisa rapida – para desabilitar

<button class=*"btn btn-default js-tooltip"* type=*"button"*

title=*"Pesquisa avançada"* data-toggle=*"modal"*

data-target=*"#pesquisaRapidaClientes"*

th:disabled=*"${venda.salvarProibido}"*>

<i class=*"glyphicon glyphicon-search"*></i>

</button>

Nota: Quando uma venda for cancelada sera proibido editar qualquer um de seus campos e itens da venda

O botão venda devera aparecer somente na edição da venda e quando seu status for diferencte de cancelado.

**No .Java CadastroVendaController**

**– Cancelamento da venda**

@PostMapping(value = "/nova", params = "cancelar")

**public** ModelAndView cancelar(Venda venda, BindingResult result

,RedirectAttributes attributes, @AuthenticationPrincipal UsuarioSistema usuarioSistema){

**try** {

cadastroVendaService.cancelar(venda);

} **catch** (AccessDeniedException e) {

**return** **new** ModelAndView("/403"); //poderia enviar para outro pagina

}

//cadastroVendaService.cancelar(venda);

attributes.addFlashAttribute("mensagem", "Venda cancelada com sucesso");

**return** **new** ModelAndView("redirect:/vendas/" + venda.getCodigo());

}

**No .Java CadastroVendaService**

**– Cancelamento da venda**

/\*\* A anotacao "@PreAuthorize" permite fazer referencia ao objeto venda sendo passado no metodo.

"#venda.usuario == principal.usuario" - somente o usuario que criou a venda pode cancelar ou

ou quem tem o papel - "or hasRole('CANCELAR\_VENDA')". Esse controle faz parte do spring

security e e necessario ter anotation **@EnableGlobalMethodSecurity**(prePostEnabled = true)

no arquivo SecurityConfig do spring, somente usuario com a role CANCELAR\_VENDA pode cancelar uma venda \*/

@PreAuthorize("#venda.usuario == principal.usuario or hasRole('CANCELAR\_VENDA')")

@Transactional

**public** **void** cancelar(Venda venda) {

Venda vendaExistente = vendas.findOne(venda.getCodigo());

// retorna produto ao estoque

publisher.publishEvent(**new** VendaCanceladaEvent(vendaExistente));

vendaExistente.setStatus(StatusVenda.***CANCELADA***);

vendas.save(vendaExistente);

}

**No .Java SecurityConfig – Cancelamento da venda**

@EnableGlobalMethodSecurity(prePostEnabled = **true**) //permite que a anotacao PreAuthorize seja utilizada no service

**public** **class** SecurityConfig **extends** WebSecurityConfigurerAdapter {…}

**No .Java Venda – Cancelamento da venda**

**public** **boolean** isSalvarPermitido(){

**return** !status.equals(StatusVenda.***CANCELADA***);

}

**public** **boolean** isSalvarProibido(){

**return** !isSalvarPermitido();

}

**Estrutura de pasta da aplicação**

**O que - 2?**

Exibir a Pagina “/” ou Inicial da aplicação

**Onde - 2?**

No browser

**Dependência de código de terceiros - 2?**

Maven

CSS

JavaScript

Chart.js – www.chartjs.org

Download: <https://github.com/chartjs/Chart.js/releases/tag/v2.2.1>

Arquivo: Chart.min.js

Instalação: copiar colar na pasta x

Carregamento na app:

**Dependência de código template - 2?**

Template

**Implementar código - 2?**

**No .HTML dasbord.graficos.js - 2**

**No .CSS brewer.css - 2**

**No .Html Dasboard - 2**

**No .Java DasboardController - 2**

@Controller

**public** **class** DashboardController {

@Autowired

Vendas vendas;

@Autowired

Cervejas cervejas;

@Autowired

Clientes clientes;

@GetMapping("/")

**public** ModelAndView dashboard(){

ModelAndView mv = **new** ModelAndView("Dashboard");

mv.addObject("vendasNoAno", vendas.valorTotalNoAno());

mv.addObject("vendasNoMes", vendas.valorTotalNoMes());

mv.addObject("tickeMedio", vendas.valorTicketMedioNoAno());

mv.addObject("valorItensEstoque", cervejas.valorItensEstoque());

mv.addObject("totalClientes", clientes.count());

**return** mv;

}

}

**No .Java Vendas - 2 (interface)**

**public** **interface** Vendas **extends** JpaRepository<Venda, Long>, VendasQueries {

}

**No .Java VendasImpl - 2**

@Override

**public** BigDecimal valorTotalNoAno() {Optional<BigDecimal> optional = Optional.*ofNullable*(

manager.createQuery("select sum(valorTotal) from Venda where year(dataCriacao) = :ano and status = :status", BigDecimal.**class**)

.setParameter("ano", Year.*now*().getValue())

.setParameter("status", StatusVenda.***EMITIDA***)

.getSingleResult());

**return** optional.orElse(BigDecimal.***ZERO***);

}

**3.1 O Que ?**

Implementar consultas nativas com Java

**3.2 Onde ?**

Na tela dasboard do sistema, para exibir o gráfico de vendas dos últimos 6 mêses.

**3.3 Tem dependência de códigos de terceiros?**

Mave

.CSS

JS

**3.4 Codigos de configuração**

**3.4.1 No .Java JPAConfig**

Informando local onde se encotra as consultas nativas em sql

// gerencia entidades

@Bean

**public** EntityManagerFactory entityManagerFactory(DataSource dataSource, JpaVendorAdapter jpaVendorAdapter){

LocalContainerEntityManagerFactoryBean factory= **new** LocalContainerEntityManagerFactoryBean();

factory.setDataSource(dataSource);

factory.setJpaVendorAdapter(jpaVendorAdapter);

factory.setPackagesToScan(Cerveja.**class**.getPackage().getName()); // obtem o nome do pacote onde se encontram as entidades

factory.setMappingResources("sql/consultas-nativa.xml"); // local onde se encontra consultas nativas slq

factory.afterPropertiesSet();

**return** factory.getObject();

}

**3.5 Implementar código onde?**

**3.5.1 No .HTML Dasboard**

<div class=*"row"*>

<div class=*"col-sm-6"*>

<div class=*"aw-graph-box"*>

<div class=*"aw-graph-box\_\_header"*>

<h2 class=*"aw-graph-box\_\_title"*>Vendas por mês <small>últimos 6 meses</small></h2>

/div>

<div class=*"aw-graph-box\_\_content"*>

<!-- <div class="aw-graph-box\_\_no-data">

<i class="fa fa-line-chart fa-2x"></i>

<span>Não há dados</span>

</div> -->

<div>

<canvas id=*"graficoVendasPorMes"* height=*"160"*></canvas>

</div>

</div>

</div>

</div>

**3.5.2 No .Java VendasController**

/\*\*@ResponseBody abilita o retorno de um objeto no formato Json\*/

@GetMapping("/totalPorMes")

**public** @ResponseBody List<VendaMes> listarTotalPorMes(){

**return** vendas.totalPorMes();

}

Nota: para teste rápido digitar a url “<http://localhost:8080/brewer/vendas/totalPorMes>”

no browser e a lista de objetos: [{"mes":"2017/03","total":1},{"mes":"2017/02","total":0},{"mes":"2017/01","total":0},{"mes":"2016/12","total":0},{"mes":"2016/11","total":0},{"mes":"2016/10","total":0}]

sera retornada.

**3.5.3 No .XML consulta-nativas**

<?xml version=*"1.0"* encoding=*"UTF-8"*?>

<entity-mappings xmlns=*"http://xmlns.jcp.org/xml/ns/persistence/orm"*

xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*

xsi:schemaLocation=*"http://xmlns.jcp.org/xml/ns/persistence/orm http://xmlns.jcp.org/xml/ns/persistence/orm\_2\_1.xsd"*

version=*"2.1"*>

<named-native-query name=*"Vendas.totalPorMes"* result-set-mapping=*"totalPorMesMapping"*>

<query>

select date\_format(data\_criacao, '%Y/%m') mes

,count(\*) total

from venda

where data\_criacao > DATE\_SUB(NOW(), INTERVAL 6 MONTH)

and status = 'EMITIDA'

group by date\_format(data\_criacao, '%Y/%m')

order by date\_format(data\_criacao, '%Y/%m') desc

</query>

</named-native-query>

<sql-result-set-mapping name=*"totalPorMesMapping"*>

<constructor-result target-class=*"com.algaworks.brewer.dto.VendaMes"*>

<column name=*"mes"* class=*"java.lang.String"*/>

<column name=*"total"* class=*"java.lang.Integer"*/>

</constructor-result>

</sql-result-set-mapping>

</entity-mappings>

**3.5.4 No .Java VendaMes**

Nota: Deve ser criado um método construtor padrão e um método construtor com os retornos da consulta.

**public** **class** VendaMes {

**private** String mes;

**private** Integer total;

**public** VendaMes() {

}

**public** VendaMes(String mes, Integer total) {

**super**();

**this**.mes = mes;

**this**.total = total;

}

Gets e sets ....

}

**3.5.4 No .Java VendasImpl**

@SuppressWarnings("unchecked")

@Override

**public** List<VendaMes> totalPorMes() {

List<VendaMes> vendasMes = manager.createNamedQuery("Vendas.totalPorMes").getResultList();

// Meses que não tiveram vendas serão inseridos no grafico com valor igual a zero

InsereMesSemVendasComValorIgualZero(vendasMes);

**return** vendasMes;

}

**private** **void** InsereMesSemVendasComValorIgualZero(List<VendaMes> vendasMes) {

LocalDate hoje = LocalDate.*now*();

// verifica se tem algum mes faltante, caso tiver inseri o mes com valor 0

**for**(**int** i = 1; i <= 6; i++){

// iniciar com o ano e mes hoje, ex: 2017/02

String mesDaVenda = String.*format*("%d/%02d", hoje.getYear(), hoje.getMonthValue());

// na lista de vendas existe o Mes existe mesId

**boolean** possuiMes = vendasMes.stream().filter(v -> v.getMes().equals(mesDaVenda)).findAny().isPresent();

**if**(!possuiMes){

vendasMes.add(i -1, **new** VendaMes(mesDaVenda, 0));

}

// retrocede um mes, até 6 meses

hoje = hoje.minusMonths(1);

}

**if**(vendasMes.size() > 6){

vendasMes.remove(vendasMes.size()-1);

}

}

**3.5.5 No .JS dashboard.graficos**

**var** Brewer = Brewer || {};

Brewer.GraficoVendaPorMes = (**function**() {

**function** GraficoVendaPorMes() {

**this**.ctx = $('#graficoVendasPorMes')[0].getContext('2d');

**this**.totalItem = $('.js-total-item');

}

GraficoVendaPorMes.prototype.iniciar = **function**() {

onFormataTotalItens.call(**this**);

$.ajax({

url: 'vendas/totalPorMes',

method: 'GET',

success: onDadosRecebidos.bind(**this**)

});

}

**function** onDadosRecebidos(vendaMes) {

**var** meses = [];

**var** valores = [];

vendaMes.forEach(**function**(obj) {

meses.unshift(obj.mes); // unshift - insere no inicio da lista cada novo objeto

valores.unshift(obj.total);

});

//[{"mes":"2017/02","total":7},{"mes":"2017/01","total":5},{"mes":"2016/12","total":5},

//{"mes":"2016/11","total":2},{"mes":"2016/10","total":3},{"mes":"2016/09","total":7}]

**var** graficoVendasPorMes = **new** Chart(**this**.ctx, {

type: 'line',

data: {

labels: meses,

datasets: [{

label: 'Vendas por mês',

backgroundColor: "rgba(26,179,148,0.5)",

pointBorderColor: "rgba(26,179,148,1)",

pointBackgroundColor: "#fff",

data: valores

}]

},

});

}

**function** onFormataTotalItens(){

**this**.totalItem.html(Brewer.formataInteiroComPonto(**this**.totalItem.data('valor')));

}

**return** GraficoVendaPorMes;

}());

$(**function**() {

**var** graficoVendaPorMes = **new** Brewer.GraficoVendaPorMes();

graficoVendaPorMes.iniciar();

**var** graficoVendaPorOrigem = **new** Brewer.GraficoVendaPorOrigem();

graficoVendaPorOrigem.iniciar();

});

**4.1 O Que ?**

Internacionalização da aplicação, permite as mensagens da aplicação (textos da tela) ser exibida em vários idiomas, para tanto utiliza o arquivo messages.properties com default, e o arquivos internacionalizado basta mudar o idioma do browser que ele vai usar o arquivo de internacionalização que segue a convenção messsages\_2PrimeirasIniciaisDoIdioma\_2PrimeirasIniciaisDoPais.properties,

as iniciais ou siglas são as mesmas utilizadas no Aceept Headers dos browse.

**Request Headers:**

**Accept-Language:**

pt-BR,pt;q=0.8,en-US;q=0.6,en;q=0.4

Ver Internaciolização das mensagens de validação em 5.1

**4.2 Onde ?**

Nota: Afeta a aplicação toda.

**4.3 Tem dependência de códigos de terceiros?**

Mave

.CSS

JS

Thymeleaf -

**4.4 Codigos de configuração**

**4.4.1 No .java WebConfig**

**Nota:**remover o Bean (caso exista)

Public LocaleResolver localeResolver (){

Return new FixedLocaleResolver(new Locale(“pt”, “BR”))

}

**4.4.2 No Browser ajustar a linguagem**

Nota:

**4.5 Implementar código onde?**

**4.5.1 No .HTML CadastroVenda**

**Nota:** o # no th:text”#{...}” é necessário para o thymeleaf saber que deve buscar a chave da mensagem no arquivo messages.properties

<div class=*"col-xs-10"*>

<h1 th:if=*"${venda.nova}"* th:text=*"#{venda.cadastro.titulo}"*>Cadastro de venda</h1>

<h1 th:unless=*"${venda.nova}"* th:text=*"#{venda.edicao.titulo(${venda.codigo})}"*>Edição da venda nº [[${venda.codigo}]]</h1>

</div>

**4.5.2 No .properties messages**

venda.cadastro.titulo = Cadastro de venda

venda.edicao.titulo = Edi\u00e7\u00e3o da venda n\u00ba **{0}**

**4.5.3 No .properties messages\_en\_US**

venda.cadastro.titulo = Registration order

venda.edicao.titulo = Editing order number **{0}**

**4.1 O Que ?**

Formatação Numerica em função do locale informado no browser. Caso o locale seja Estados Unidos a formatação numérica para real deixara de funcionar o que não deveria, para resolver o problema sera criado um NumberFormater.

**4.2 Onde ?**

Nota:

**4.3 Tem dependência de códigos de terceiros?**

**4.4 Codigos de configuração**

**4.4.1 No .Java Webconfig**

**Metodo mvcConversionService**

@Bean

**public** FormattingConversionService mvcConversionService (){

DefaultFormattingConversionService conversionService = **new** DefaultFormattingConversionService();

…….

//implementacao padrao do spring

//NumberStyleFormatter bigdecimalFormatter = new NumberStyleFormatter("#,##0.00");

//implementacao customizada para conversao de valores, em função da intercionalizacao

BigDecimalFormatter bigdecimalFormatter = **new** BigDecimalFormatter("#,##0.00");

conversionService.addFormatterForFieldType(BigDecimal.**class**, bigdecimalFormatter);

//implementacao padrao do spring

//NumberStyleFormatter integerFormatter = new NumberStyleFormatter("#,##0");

BigDecimalFormatter integerFormatter = **new** BigDecimalFormatter("#,##0");

conversionService.addFormatterForFieldType(Integer.**class**, integerFormatter);

**}**

**4.5 Implementar código onde?**

**4.5.1 No .Java BigDecimalFormatter**

/\*\*

\* Essa classe faz a conversao de valores para o locale pt-BR, de um Bigdecimal ou de uma string evitando assim

\* o erro abaixo quando do uso de intercionalizacao

\* Failed to convert property value of type [java.lang.String] to required type [java.math.BigDecimal] for property valorFrete;

\* nested exception is java.lang.NumberFormatException

\*

\* **@author** Udinei

\*

\*/

**public** **class** BigDecimalFormatter **implements** Formatter<BigDecimal> {

**private** DecimalFormat decimalFormat;

**public** BigDecimalFormatter(String pattern){

NumberFormat format = NumberFormat.*getInstance*(**new** Locale("pt", "BR"));

decimalFormat = (DecimalFormat) format;

decimalFormat.setParseBigDecimal(**true**);

decimalFormat.applyPattern(pattern);

}

@Override

**public** String print(BigDecimal object, Locale locale) {

**return** decimalFormat.format(object);

}

@Override

**public** BigDecimal parse(String text, Locale locale) **throws** ParseException {

**return** (BigDecimal) decimalFormat.parse(text);

}

}

**5.1 O Que ?**

Customizar acesso e criação de Mensagens de validação, para diminuir código na entidade e permitir alterar as mensagem em um único arquivos.

**5.2 Onde ?**

No arquivos messages.properties, em vez de buscar as mensagem contidas nas anotation de valiação dos atributos das Entidades, ex:.

**Nota:**

Deve-se descobrir qual é a chave das anotation do Bean validator que esta dentro do arquivo .jar hibernate-validator-5.2.4 dentro do pacote validator dentro do arquivo ValidationMessages\_pt\_BR.properties

**5.3 Tem dependência de códigos de terceiros?**

Sim. Do arquivo hibernate-validator-5.2.4.jar

**5.4 Codigos de configuração**

**5.4.1 No .Java WebConfig**

Pra que as mensagens possam ser recuperadas do messages.properties

// tambem usado em: internacionalizacao, informa onde estão os arquivos de mensagens

@Bean

**public** MessageSource messageSource(){

ReloadableResourceBundleMessageSource bundle = **new**

ReloadableResourceBundleMessageSource();

bundle.setBasename("classpath:/messages");

bundle.setDefaultEncoding("UTF-8");

**return** bundle;

}

// usado em: intercionalizacao, informa onde estão as mensagens de validacao

@Bean

**public** LocalValidatorFactoryBean validator() {

LocalValidatorFactoryBean validatorFactoryBean = **new** LocalValidatorFactoryBean();

validatorFactoryBean.setValidationMessageSource(messageSource());

**return** validatorFactoryBean;

}

// usado em: internacionalizacao,

@Override

**public** Validator getValidator() {

**return** validator();

}

**5.5 Implementar código onde?**

**5.5.1 No .Java Cerveja (Entidade)**

@SKU

@NotBlank // anotação do bean validation implementada pelo hibernate validator

**private** String sku;

@NotBlank

**private** String nome;

**5.5.2 No .properties Messages**

org.hibernate.validator.constraints.NotBlank.message = **{0}** \u00e9 obrigat\u00f3rio

com.algaworks.constraints.SKU.message = **{0}** deve seguir o padrão XX9999

cerveja.sku = SKU

cerveja.nome = Nome

**5.5.3 No .Java SKU (classe)**

@Target({ ElementType.***FIELD***, ElementType.***METHOD***, ElementType.***ANNOTATION\_TYPE***})

@Retention(RetentionPolicy.***RUNTIME***)

@Constraint(validatedBy = {})

@Pattern(regexp = "([a-zA-Z]{2}\\d{4})?") // [a-zA-Z] permite caracteres de a à z maiusculo e minusculo,{2} até dois caracter. \\d Numeros até {4} quatro digitos numericos. "?" e só valida se o campo tiver preenchido

**public** **@interface** SKU {

@OverridesAttribute(constraint = Pattern.**class**, name = "message")

String message() **default** "{com.algaworks.constraints.SKU.message}";

Class<?>[] groups() **default** {};

Class<? **extends** Payload>[] payload() **default** {};

}

**5.5.3 No .HTML CadastroVenda**

**Nota:** o # no th:text”#{...}” é necessário para o thymeleaf saber que deve buscar a chave da mensagem no arquivo messages.properties

<div class=*"col-xs-10"*>

<h1 th:if=*"${venda.nova}"* th:text=*"#{venda.cadastro.titulo}"*>Cadastro de venda</h1>

<h1 th:unless=*"${venda.nova}"* th:text=*"#{venda.edicao.titulo(${venda.codigo})}"*>Edição da venda nº [[${venda.codigo}]]</h1>

</div>

**6.1 O Que ?**

Fazer deploy da aplicação no servidor de produção

**6.2 Onde ?**

No servidor de aplicação Linux ou Windows, na pasta lib do tomcat ou outro servidor de aplicação.

**6.3 Tem dependência?**

Sim.

Java Versão 1.8

Banco de dados MySql instalado

Tomcat instalado

Na pasta /lib deve conter os jars:

- c3p0-0.9.5.2.jar - cria pool de conexão

- mchange-commons-java-0.2.11.jar -

- mysql-connector-java-5.1.40-bin.jar – driver de conexao com bd myslql

Maven

.CSS

JS

937

**6.4 Codigos de configuração**

**6.1.1 No .XML pom**

<build>

<!-- finalName - define que o pacote de instalacao, contera somente o nome da aplicacao -->

<finalName>${artifactId}</finalName>

<plugins>

<plugin>

<artifactId>maven-compiler-plugin</artifactId>

<version>${maven-compiler-plugin.version}</version>

<configuration>

<source>${java.version}</source>

<target>${java.version}</target>

</configuration>

</plugin>

<plugin>

<groupId>org.flywaydb</groupId>

<artifactId>flyway-maven-plugin</artifactId>

<version>${flyway-maven-plugin.version}</version>

<configuration>

<driver>com.mysql.jdbc.Driver</driver>

</configuration>

</plugin>

</plugins>

</build>

**6.5 Implementar código, onde?**

Nota:

**6.5.1 No .HTML xxx**

Nota:

**7.1 O Que ?**

Criação de Profile para aplicação, para definir qual implementação sera usada, quando duas classes utilizam a mesma interface na aplicação

**7.2 Onde ?**

Nas classes que usam a anotation @Profile e @Component do Spring.

@Profile("local")

@Component

**public** **class** FotoStorageLocal **implements** FotoStorage {

**private** **static** **final** Logger ***logger*** = LoggerFactory.*getLogger*(FotoStorageLocal.**class**);

…

}

**7.3 Tem dependência de códigos de terceiros?**

S3 Amazon

Mave

.CSS

JS

**7.4 Codigos de configuração**

**7.4.1 No .Java AppInitializer**

Nota: No método onStartup é que definido qual profile sera usado (classes com a anotação @Profile). Quando a aplicação esta subindo o profile que esta definido nesse método é que sera usado o defaut

// Metodo utilizado na configuracao do profile de carregamento de imagens da nuvem ou local

@Override

**public** **void** onStartup(ServletContext servletContext) **throws** ServletException {

**super**.onStartup(servletContext);

servletContext.setInitParameter("spring.profiles.default","local"); // profile default é local, e sera mudado em tempo de execução para profile desejado

}

..

}

@Profile("prod")

@Component

**public** **class** FotoStorageS3 **implements** FotoStorage {

**private** **static** **final** Logger ***logger*** = LoggerFactory.*getLogger*(FotoStorageS3.**class**);

…

}

**7 .5 Implementar código onde?**

**7.5.1 No .java FotoStorageLocal (component)**

Nota:

**7.5.1 No .java FotoStorageS3 (component)**

Nota:

**7.5.1 No .java FotoStorage (interface)**

Nota:

**------------------------------------------------------------------------------------------------------------------------------------------------------------------------**

**8.1 O Que ?**

Acesso a banco de dados

**8.2 Onde ?**

No banco de dados Mysql Local ou remoto

**8.3 Tem dependência de códigos de terceiros?**

Sim

Maven Baixar dependencia no maven

**8.4 Codigos de configuração**

**8.4.1 No .XML context**

Criar em src/main/webapp/META-INF, caso não exista criar a pasta META-INF e o arquivo contexto.xml, inserir o código abaixo.

Instalar Drivers de na pasta lib do tomcat:

**Driver:** Banco MySql drive mysql-connector-java-5.1.39-bin.jar

**Controlador de pool de conecção:** c3p0-0.9.5.jar

mchange-commons-java-0.2.11.jar

<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE xml>

<Context>

<Resource name="jdbc/brewerDB" auth="Container" <!—nome datasource usado na classe -->

factory="org.apache.naming.factory.BeanFactory" <!—fabrica de bean -->

type="com.mchange.v2.c3p0.ComboPooledDataSource" <!—pool de conexao -->

driverClass="com.mysql.jdbc.Driver" <!—Driver de bd -->

jdbcUrl="jdbc:mysql://localhost/brewer?useSSL=false" <!— url de conexao -->

user="root"

password="root"

initialPoolSize="5" <!— inicio do tamanho do pool de conexao-->

minPoolSize="5" <!— tamanho do pool de conexao mínimo -->

maxPoolSize="5" <!— tamanho do pool de conexao maximo-->

/>

</Context>

**8.4.2 No .java JPAConfig**

Configurar na classe JPAConfig o DataSource

@EnableTransactionManagement

**public** **class** JPAConfig {

/\*\* configura fonte de dados (dataSource) acessa context.xml \*/

@Bean

**public** DataSource dataSource(){

JndiDataSourceLookup dataSourceLookup = **new** JndiDataSourceLookup();

dataSourceLookup.setResourceRef(**true**);

**return** dataSourceLookup.getDataSource("jdbc/brewerDB");

}

**8.4.3 No .java AppInitialize**

Na classe AppInitialize.java ativar as configurações

// configuração que envolve toda a aplicação servicos etc..

@Override

protected Class<?>[] getRootConfigClasses() {

return new Class<?>[] {JPAConfig.class, ServiceConfig.class, SecurityConfig.class, S3Config.class} ;

}

**8.5 Implementar código? Onde?**

Nota:

**8.5.1 No .HTML xxx**

Nota:

**9.1 O Que ?**

Executar SQL de criação da estrutura de Banco de dados, onde as tabelas, o usuario admin do sistema e os papeis já serão criados.

**9.2 Onde ?**

No banco de dados através do flyway

**9.3 Tem dependência de códigos de terceiros?**

Sim.

SQL

Flyway

**9.4 Codigos de configuração**

**4.4.1 No .Java xxx**

Nota:

**9.5 Implementar código? Onde?**

Nota:No arquivos de migração do flyway

**9.5.1 No .SQL xxx**

Nota:

**--------------------------------------------------------**

**10.1 O Que ?**

Enviar email ao cliente

**10.2 Onde ?**

Na tela de vendas, sera disparado o envio de email, com os itens orçados apartir da tela de venda, uma venda.

**10.3 Tem dependência de códigos de terceiros?**

Dependencia Maven:

<!-- Java Mail -->

<javax.mail.version>1.5.6</javax.mail.version>

<!-- Java Mail -->

<dependency>

<groupId>com.sun.mail</groupId>

<artifactId>javax.mail</artifactId>

<version>${javax.mail.version}</version>

<scope>compile</scope>

</dependency>

Sendgrid – servidor de email

Smtp: smtp.sendgrid.net

Porta de envio: 587

**10.4 Codigos de configuração**

**10.4.1 No .Java AppInitialize**

Adicionar no método getServletConfigClasses “MailConfig.class” para poder injetar o thymeleaf no html do email e usar seus recursos.

// configuração que envolve camada Web

@Override

**protected** Class<?>[] getServletConfigClasses() {

**return** **new** Class<?>[] { WebConfig.**class**, MailConfig.**class** };

}

**10.4.2 No .Java MailConfig**

/\*\* A primeira anotation **@PropertySource** tenta carregar o arquivo mail-ambiente.properties, "ambiente" e

\* uma variavel do tomcat, caso nao encontre essa variavel será usado no nome do arquivo

\* que e a string "local" que esta depois dos dois pontos do nome da variavel, e caso não encontre no

\* arquivo mail-local as configuração, vai executando na sequencia os outros anottation **@PropertySource.**

\* Caso as propriedades nos arquivos tiverem os mesmos nomes esses serão sobreescritos. Caso a propriedade

\* nao exista pode ser usada uma propriedade username por exemplo de um arquivo e password de outro.

\* ignoreResourceNotFound - caso não encontre o arquivo, sube a aplicacao sem erros

\* Uso de arquivo externo "brewer-mail.properties" evita que dados reais ou temporarios fiquem expostos

\* em producao ou no github.\*/

@Configuration

@ComponentScan(basePackageClasses = Mailer.**class**)

@PropertySource({ "classpath:env/mail-${ambiente:local}.properties" })

@PropertySource( value = { "file:${USERPROFILE}/.mail.brewer/brewer-mail.properties"}, ignoreResourceNotFound = **true**)

**public** **class** MailConfig {

// injeta objeto que acessa o classpath da aplicacao env

@Autowired

**private** Environment env;

@Bean

**public** JavaMailSender mailSender(){

JavaMailSenderImpl mailSender = **new** JavaMailSenderImpl();

mailSender.setHost("smtp.sendgrid.net"); // servidor de email sendgrid.net

mailSender.setPort(587);

mailSender.setUsername(env.getProperty("username.email"));

mailSender.setPassword(env.getProperty("password"));

System.***out***.println(">>>> username: " + env.getProperty("username.email"));

System.***out***.println(">>>> password: " + env.getProperty("password"));

Properties props = **new** Properties();

props.put("mail.transport.protocol", "smtp");

props.put("mail.smtp.auth",**true**);

props.put("mail.smtp.starttls.enable" ,**true**);

props.put("mail.debug", **false**);

props.put("mail.smtp.connectiontimeout", 10000); // miliseconds

mailSender.setJavaMailProperties(props);

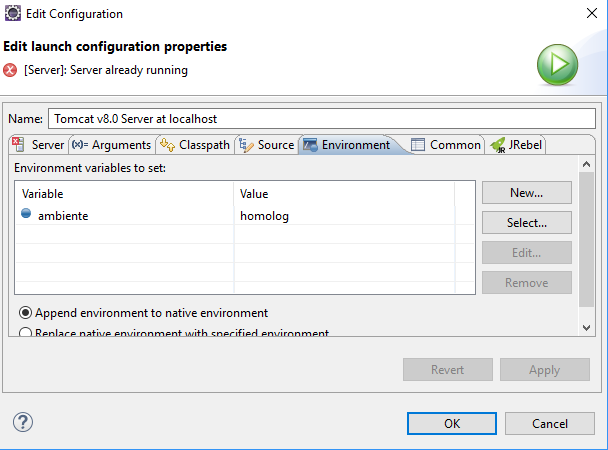
**return** mailSender;

}

}

**10.4.2 No .TomCat Server localhost**

Configurando a variável de ambiante para envio de email usando as configurações de homologação. E possível fazer também essa configuração no Tomcat externo.



**10.5 Implementar código? Onde?**

Usando chamadas assincronas para evitar o usuário ter que ficar esperando um retorno da aplicação ao enviar o email. Deve ser abilitado com a anottation @**EnableAsync** na classe WebConfig.java e a anottation @**Async** no método que sera assincrono:

@EnableAsync

**public** **class** WebConfig **extends** WebMvcConfigurerAdapter **implements** ApplicationContextAware {…}

**10.5.1 No .java Mailer**

Nota:

**10.5.2 No .java VendasController**

/\*\* Executa esse metodo se tiver na url o paramento enviarEmail \*/

@PostMapping(value="/nova", params="enviarEmail")

**public** ModelAndView enviarEmail(Venda venda, BindingResult result, RedirectAttributes attributes

, @AuthenticationPrincipal UsuarioSistema usuarioSistema){

validarVenda(venda);

// aplicando o validador "@Valid" apos inserir os itens

vendaValidator.validate(venda, result);

**if**(result.hasErrors()){

**return** nova(venda);

}

venda.setUsuario(usuarioSistema.getUsuario());

venda = cadastroVendaService.salvar(venda);

mailer.enviar(venda); // envia email de forma assincrona

attributes.addFlashAttribute("mensagem", String.*format*("Venda nº %d salva com sucesso e e-mail enviado", venda.getCodigo()));

**return** **new** ModelAndView("redirect:/vendas/nova");

}

**10.5.3 No .HTML ResumoVenda**

<!DOCTYPE html>

<html xmlns=*"http://www.w3.org/1999/xhtml"*

xmlns:th=*"http://www.thymeleaf.org"*>

<body style="font-family: *Arial*; font-size: *14px*;">

<p style="text-align: *right*">

<!-- cid: espaco de conteudo disponivel no email, tipo anexo -->

<img src=*"../../static/images/logo-gray-brewer.png"* th:src=*"|cid:${logo}|"*>

</p>

<h1 style="font-size: *22px*" th:text=*"|Olá ${venda.cliente.nome}|"*>Olá Maria!</h1>

<p>

Estamos enviando para você a relação de itens da sua compra em nossa loja de cervejas.

</p>

<table style="width: *100%*; border-collapse: *collapse*; margin: *20px 0*">

<th:block th:each=*"item : ${venda.itens}"*>

<tr style="border-bottom: *1px solid #ccc*; border-top: *1px solid #ccc*">

<td style="width: *50px*; padding-bottom: *10px*; padding-top: *10px*">

<img style="width: *40px*" src=*"../../static/images/cerveja-mock.png"* th:src=*"|cid:${mockCerveja}|"* th:unless=*"${item.cerveja.temFoto()}"* />

<img style="width: *40px*" src=*"../../static/images/cerveja-mock.png"* th:src=*"|cid:foto-${item.cerveja.codigo}|"* th:if=*"${item.cerveja.temFoto()}"* />

</td>

<td style="vertical-align: *top*; padding-bottom: *10px*; padding-top: *10px*">

<div style="display: *block*; font-weight: *bold*; font-size: *16px*" th:text=*"${item.cerveja.nome}"*>Bhrama Lata 350 ml</div>

<div style="color: *#8e8e8e*" th:text=*"${item.cerveja.origem.descricao}"*>Nacional</div>

<!-- numbers.formatDecimal(item.valorUnitario, 1, 2) <<<< esse codigo "valorUnitario" numero a ser formatado, (1) numero de digitos da parte inteira, (2) numero de digitos da parte decimal-->

<div style="color: *#8e8e8e*" th:text=*"|${item.quantidade} x R$${#numbers.formatDecimal(item.valorUnitario, 1, 2)}|"*>1 x R$1,90</div>

</td>

<td style="vertical-align: *top*; text-align: *right*; padding-bottom: *10px*; padding-top: *10px*">

<span style="display: *block*; font-weight: *bold*; font-size: *16px*" th:text=*"|R$${#numbers.formatDecimal(item.valorTotal, 1, 2)}|"*>R$1,90</span>

</td>

</tr>

<th:block th:if=*"${false}"*>

<tr style="border-bottom: *1px solid #ccc*; border-top: *1px solid #ccc*">

<td style="width: *50px*; padding-bottom: *10px*; padding-top: *10px*">

<img style="width: *40px*" src=*"../../static/images/cerveja-mock.png"* />

</td>

<td style="vertical-align: *top*; padding-bottom: *10px*; padding-top: *10px*">

<div style="display: *block*; font-weight: *bold*; font-size: *16px*">Heineken</div>

<div style="color: *#8e8e8e*">Internacional</div>

<div style="color: *#8e8e8e*">450 x R$3,93</div>

</td>

<td style="vertical-align: *top*; text-align: *right*; padding-bottom: *10px*; padding-top: *10px*">

<span style="display: *block*; font-weight: *bold*; font-size: *16px*">R$1.768,50</span>

</td>

</tr>

</th:block>

</th:block>

</table>

<p style="font-size: *12px*">

\* O status da sua compra é: <strong style="color: *red*"><span th:text=*"${venda.status.descricao}"*>orçamento</span></strong>.

</p>

<table style="width: *100%*;">

<tr>

<td style="text-align: *right*; font-weight: *bold*" >Subtotal:</td>

<td style="width: *120px*; text-align: *right*" th:text=*"|R$${#numbers.formatDecimal(venda.valorTotalItens, 1, 2)}|"*>R$1.200,50</td>

</tr>

<tr>

<td style="text-align: *right*; font-weight: *bold*">Frete:</td>

<td style="width: *100px*; text-align: *right*">

<span th:text=*"|R$${#objects.nullSafe(#numbers.formatDecimal(venda.valorFrete, 1, 2), '0,00')}|"*>R$8,00</span>

</td>

</tr>

<tr>

<td style="text-align: *right*; font-weight: *bold*">Desconto:</td>

<td style="width: *100px*; text-align: *right*"><span th:text=*"|(R$${#objects.nullSafe(#numbers.formatDecimal(venda.valorDesconto, 1, 2), '0,00')})|"*>(R$5,00)</span></td>

</tr>

<tr>

<td style="text-align: *right*; font-weight: *bold*; font-size: *16px*">Total final:</td>

<td style="width: *100px*; text-align: *right*; font-weight: *bold*; font-size: *16px*">

<span th:text=*"|R$${#numbers.formatDecimal(venda.ValorTotal, 1, 2)}|"*>R$1.200,50</span>

</td>

</tr>

</table>

<p>Obrigado.</p>

<p>Equipe Brewer!</p>

</body>

</html>

----------------------------------------------

**11.1 O Que ?**

Criação de relatórios em PDF, que serão emitidos no browser através do menu lateral da aplicação

**11.2 Onde ?**

Na IDE JasperReport Studio, utilizando arquivos .jrxml que serão compilados para .jasper, copiados para a aplicação e gerenciado para impressão pelo JasperResports.

**11.3 Tem dependência de códigos de terceiros?**

Sim

JasperResport

Maven

<!-- JasperReports -->

<jasperreports.version>6.3.0</jasperreports.version>

<!-- JasperReports Fonts -->

<jasperreports-fonts.version>6.0.0</jasperreports-fonts.version>

<!-- JasperReports -->

<dependency>

<groupId>net.sf.jasperreports</groupId>

<artifactId>jasperreports</artifactId>

<version>${jasperreports.version}</version>

<scope>compile</scope>

</dependency>

<!-- JasperReports - Fonts -->

<dependency>

<groupId>net.sf.jasperreports</groupId>

<artifactId>jasperreports-fonts</artifactId>

<version>${jasperreports-fonts.version}</version>

<scope>compile</scope>

</dependency>

JS

**11.4 Codigos de configuração**

Os relatórios são criados em .jrxml e compilados para .jasper no JasperReports Studio

Criar uma viewResolvar para tratar relatórios

**4.4.1 No .SQL arquivoX**

/\* O sql abaixo retorna as venda realizadas em um período, datainicial e data final\*/

SELECT v.data\_criacao AS data\_criacao

, v.valor\_total AS valor\_total

, c.nome AS nome\_cliente

, u.nome AS nome\_vendedor

, v.codigo AS codigo

FROM venda v INNER JOIN cliente c ON v.codigo\_cliente = c.codigo

INNER JOIN usuario u ON v.codigo\_usuario = u.codigo

WHERE v.status = 'EMITIDA'

AND v.data\_criacao BETWEEN '2016-02-01 00:00:00' AND '2016-09-08 23:59:59'

ORDER BY v.data\_criacao ASC

/\* O SQL abaixo gera inserts, com datas aleatórias e valores aleatorios no banco mysql - tabela venda - , unixtime utiliza o tempo em milisegundos \*/

insert into brewer.venda (data\_criacao, valor\_total, status, codigo\_cliente, codigo\_usuario)

values (

FROM\_UNIXTIME(ROUND((RAND() \* (1473292800 - 1454284800) + 1454284800))) /\*datas \*/

, round(rand() \* 10000, 2) /\* valores aleatorios \*/

, 'EMITIDA'

, round(rand() \* 7) + 1 /\* entre os cliente 1 e 7\*/

, round(rand() \* 2) + 1) /\* entre os usuario 1 e 2\*/

**Nota:** Em para obter uma data em milisengudo, usar

LocalDate date = LocalDate.of(2016, Month.FEBRUARY, 1)

Imprimindo: sysout(date.toEpochDay() \* 24 \* 60 \* 60)

**4.4.1 No .Java WebConfig**

Criar ViewResolver para tratar relatorios

/\*\* Esse metodo e ViewResolver para tratar relatorios .jasper \*/

@Bean

**public** ViewResolver JasperReportsViewResolver(DataSource datasource){

JasperReportsViewResolver resolver = **new** JasperReportsViewResolver();

resolver.setPrefix("classpath:/relatorios/");

resolver.setSuffix(".jasper");

resolver.setViewNames("relatorio\_\*");

resolver.setViewClass(JasperReportsMultiFormatView.**class**);

resolver.setJdbcDataSource(datasource);

resolver.setOrder(0);

**return** resolver;

}

**11.5 Implementar código? Onde?**

**11.5.1 No .HTML RelatorioVendasEmitidas**

<!DOCTYPE html>

<html xmlns=*"http://www.w3.org/1999/xhtml"*

xmlns:th=*"http://www.thymeleaf.org"*

xmlns:layout=*"http://www.ultraq.net.nz/thymeleaf/layout"*

layout:decorate=*"~{layout/LayoutPadrao}"*>

<head>

<title>Relatório - Vendas Emitidas</title>

</head>

<section layout:fragment=*"conteudo"*>

<div class=*"page-header"*>

<div class=*"container-fluid"*>

<h1>

Relatório de vendas emitidas

</h1>

</div>

</div>

<div class=*"container-fluid"*>

<form method=*"POST"* th:object=*"${periodoRelatorio}"* th:action=*"@{/relatorios/vendasEmitidas}"*

target=*"\_blank"* >

<th:block th:include=*"fragments/MensagensErroValidacao"*></th:block>

<div class=*"row"*>

<div class=*"form-group col-sm-12"*>

<label for=*"dataInicio"*>Data de criação</label>

<div class=*"form-inline"*>

<input type=*"text"* class=*"form-control aw-form-control-inline-sm js-date"*

id=*"dataInicio"* th:field=*"\*{dataInicio}"* autocomplete=*"off"*/>

<label for=*"a"* class=*"aw-form-label-between"*>a</label>

<input type=*"text"* class=*"form-control aw-form-control-inline-sm js-date"*

id=*"dataFim"* th:field=*"\*{dataFim}"* autocomplete=*"off"*/>

</div>

</div>

</div>

<button type=*"submit"* class=*"btn btn-primary"*>Emitir</button>

</form>

</div>

</section>

</html>

**11.5.2 No .java PeriodoRelatorio (DTO)**

**public** **class** PeriodoRelatorio {

**private** LocalDate dataInicio;

**private** LocalDate dataFim;

**gets e sets**

**....**

**}**

**11.5.3 No .java RelatoriosController**

@Controller

@RequestMapping("/relatorios")

**public** **class** RelatoriosController {

@GetMapping("/vendasEmitidas")

**public** ModelAndView relatoriosVendasEmitidas(){

ModelAndView mv = **new** ModelAndView("relatorio/RelatorioVendasEmitidas");

mv.addObject(**new** PeriodoRelatorio());

**return** mv;

}

@PostMapping("/vendasEmitidas")

**public** ModelAndView gerarRelatorioVendasEmitidas(PeriodoRelatorio periodoRelatorio){

Map<String, Object> parametros = **new** HashMap<>();

// converte LocalDateTime para Date

Date dataInicio = Date.*from*(LocalDateTime.*of*(periodoRelatorio.getDataInicio(), LocalTime.*of*(0, 0, 0))

.atZone(ZoneId.*systemDefault*()).toInstant());

Date dataFim = Date.*from*(LocalDateTime.*of*(periodoRelatorio.getDataFim(), LocalTime.*of*(23, 59, 59))

.atZone(ZoneId.*systemDefault*()).toInstant());

// passagem de parametros para o relatorio

parametros.put("format", "pdf");

parametros.put("data\_inicio", dataInicio);

parametros.put("data\_fim", dataFim);

**return** **new** ModelAndView("relatorio\_vendas\_emitidas", parametros);

}

}

**4.1 O Que ?**

Icones de links prontos

**4.2 Onde ?**

Utilizado em links nas paginas da aplicação

**4.3 Tem dependência de códigos de terceiros?**

Sim

Site de ícones do bootstrap: fontawesome.io

Nome da classe do ícone: fa-file-pdf-o

Mave

.CSS

JS

**4.4 Codigos de configuração**

**4.4.1 No .Java xxx**

Nota:

**4.5 Implementar código? Onde?**

Nota:

**4.5.1 No .HTML xxx**

Nota:

**---------------------------------------------------------------- Template -----------------------------------------------**

**4.1 O Que ?**

Nota:

**4.2 Onde ?**

Nota:

**4.3 Tem dependência de códigos de terceiros?**

Mave

.CSS

JS

**4.4 Codigos de configuração**

**4.4.1 No .Java xxx**

Nota:

**4.5 Implementar código? Onde?**

Nota:

**4.5.1 No .HTML xxx**

Nota: