1. MyInventoryApplication

**package** com.cts.casestudy.inventory;

**import** org.springframework.boot.SpringApplication;

**import** org.springframework.boot.autoconfigure.SpringBootApplication;

**import** org.springframework.context.annotation.Bean;

**import** org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;

@SpringBootApplication

**public** **class** MyInventoryApplication {

**public** **static** **void** main(String[] args) {

SpringApplication.*run*(MyInventoryApplication.**class**, args);

}

@Bean

**public** BCryptPasswordEncoder passwordEncoder() {

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

}

}

1. SecurityConfig

**package** com.cts.casestudy.inventory.auth;

**import** org.springframework.beans.factory.annotation.Autowired;

**import** org.springframework.context.annotation.Configuration;

**import** org.springframework.security.config.annotation.authentication.builders.AuthenticationManagerBuilder;

**import** org.springframework.security.config.annotation.method.configuration.EnableGlobalMethodSecurity;

**import** org.springframework.security.config.annotation.web.builders.HttpSecurity;

**import** org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;

**import** org.springframework.security.config.annotation.web.configuration.WebSecurityConfigurerAdapter;

**import** org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;

**import** com.cts.casestudy.inventory.service.UserDetailsServiceImpl;

@Configuration

@EnableWebSecurity

@EnableGlobalMethodSecurity(securedEnabled = **true**)

**public** **class** SecurityConfig **extends** WebSecurityConfigurerAdapter {

@Autowired

**private** UserDetailsServiceImpl userDetailsServiceImpl;

@Autowired

**private** AppAuthenticationEntryPoint appAuthenticationEntryPoint;

@Autowired

**private** BCryptPasswordEncoder passwordEncoder;

@Override

**protected** **void** configure(HttpSecurity http) **throws** Exception {

http.csrf().disable().authorizeRequests().antMatchers("/inventory/\*\*")

.hasAnyRole("ADMIN", "USER").and().httpBasic()

.realmName("MY APP REALM")

.authenticationEntryPoint(appAuthenticationEntryPoint);

}

@Autowired

**public** **void** configureGlobal(AuthenticationManagerBuilder auth)

**throws** Exception {

auth.userDetailsService(userDetailsServiceImpl).passwordEncoder(

passwordEncoder);

}

}

1. AppAuthenticationEntryPoint

**package** com.cts.casestudy.inventory.auth;

**import** java.io.IOException;

**import** javax.servlet.ServletException;

**import** javax.servlet.http.HttpServletRequest;

**import** javax.servlet.http.HttpServletResponse;

**import** org.springframework.security.core.AuthenticationException;

**import** org.springframework.security.web.authentication.www.BasicAuthenticationEntryPoint;

**import** org.springframework.stereotype.Component;

@Component

**public** **class** AppAuthenticationEntryPoint **extends** BasicAuthenticationEntryPoint {

@Override

**public** **void** commence(HttpServletRequest request,

HttpServletResponse response, AuthenticationException authException)

**throws** IOException, ServletException {

response.addHeader("WWW-Authenticate", "Basic realm=\""

+ getRealmName() + "\"");

response.sendError(HttpServletResponse.***SC\_UNAUTHORIZED***,

authException.getMessage());

}

@Override

**public** **void** afterPropertiesSet() **throws** Exception {

setRealmName("MY APP REALM");

}

}

1. InventoryController

**package** com.cts.casestudy.inventory.controller;

**import** java.util.List;

**import** org.springframework.beans.factory.annotation.Autowired;

**import** org.springframework.http.HttpHeaders;

**import** org.springframework.http.HttpStatus;

**import** org.springframework.http.ResponseEntity;

**import** org.springframework.stereotype.Controller;

**import** org.springframework.web.bind.annotation.DeleteMapping;

**import** org.springframework.web.bind.annotation.GetMapping;

**import** org.springframework.web.bind.annotation.PathVariable;

**import** org.springframework.web.bind.annotation.PostMapping;

**import** org.springframework.web.bind.annotation.PutMapping;

**import** org.springframework.web.bind.annotation.RequestBody;

**import** org.springframework.web.bind.annotation.RequestMapping;

**import** org.springframework.web.util.UriComponentsBuilder;

**import** com.cts.casestudy.inventory.entity.Product;

**import** com.cts.casestudy.inventory.exception.InventoryException;

**import** com.cts.casestudy.inventory.service.ProductService;

@Controller

@RequestMapping("inventory")

**public** **class** InventoryController {

@Autowired

**private** ProductService productService;

@GetMapping("product/{id}")

**public** ResponseEntity<Product> getProductById(@PathVariable("id") Integer id) {

Product product = productService.getProductById(id);

**return** **new** ResponseEntity<Product>(product, HttpStatus.***OK***);

}

@GetMapping("getAllproducts")

**public** ResponseEntity<List<Product>> getAllProducts() {

List<Product> list = productService.getAllProducts();

**return** **new** ResponseEntity<List<Product>>(list, HttpStatus.***OK***);

}

@GetMapping("search/{key}")

**public** ResponseEntity<?> searchProducts(@PathVariable("key") String key) {

List<Product> list;

**try** {

list = productService.searchProducts(key);

**return** **new** ResponseEntity<List<Product>>(list, HttpStatus.***OK***);

} **catch** (InventoryException e) {

**return** **new** ResponseEntity<String>("Item NotFound Exception",

HttpStatus.***OK***);

}

}

@PostMapping("product")

**public** ResponseEntity<Void> addProduct(@RequestBody Product product,

UriComponentsBuilder builder) {

**boolean** flag = productService.addProduct(product);

**if** (flag == **false**) {

**return** **new** ResponseEntity<Void>(HttpStatus.***CONFLICT***);

}

HttpHeaders headers = **new** HttpHeaders();

headers.setLocation(builder.path("/product/{id}")

.buildAndExpand(product.getId()).toUri());

**return** **new** ResponseEntity<Void>(headers, HttpStatus.***CREATED***);

}

@PutMapping("updateExpiry/{id}")

**public** ResponseEntity<?> updateProductExpiry(@PathVariable("id") Integer id) {

Product product = productService.updateProductExpiry(id);

**if** (product != **null**) {

**return** **new** ResponseEntity<Product>(product, HttpStatus.***OK***);

} **else** {

**return** **new** ResponseEntity<String>("Item NotFound",

HttpStatus.***NOT\_FOUND***);

}

}

@PutMapping("applayDiscounts")

**public** ResponseEntity<Void> applayDiscounts() {

productService.applayDiscounts();

**return** **new** ResponseEntity<Void>(HttpStatus.***OK***);

}

@DeleteMapping("delete/{id}")

**public** ResponseEntity<Void> delete(@PathVariable("id") Integer id) {

productService.deleteProduct(id);

**return** **new** ResponseEntity<Void>(HttpStatus.***NO\_CONTENT***);

}

@DeleteMapping("deleteExpiryProduct")

**public** ResponseEntity<Void> deleteExpiryProduct() {

productService.deleteExpiryProduct();

**return** **new** ResponseEntity<Void>(HttpStatus.***NO\_CONTENT***);

}

@GetMapping("sortProductsByExpiry")

**public** ResponseEntity<List<Product>> sortProducts() {

List<Product> list = productService.sortProducts();

**return** **new** ResponseEntity<List<Product>>(list, HttpStatus.***OK***);

}

}

1. InventoryWebController

**package** com.cts.casestudy.inventory.controller;

**import** java.util.Arrays;

**import** java.util.List;

**import** javax.servlet.http.HttpServletRequest;

**import** org.springframework.beans.factory.annotation.Autowired;

**import** org.springframework.beans.factory.annotation.Value;

**import** org.springframework.http.MediaType;

**import** org.springframework.http.ResponseEntity;

**import** org.springframework.stereotype.Controller;

**import** org.springframework.web.bind.annotation.ModelAttribute;

**import** org.springframework.web.bind.annotation.PathVariable;

**import** org.springframework.web.bind.annotation.RequestMapping;

**import** org.springframework.web.bind.annotation.RequestMethod;

**import** org.springframework.web.bind.annotation.RequestParam;

**import** org.springframework.web.servlet.ModelAndView;

**import** com.cts.casestudy.inventory.restclient.InventoryRestClient;

**import** com.cts.casestudy.inventory.service.ProductService;

**import** com.cts.casestudy.inventory.vo.Product;

@Controller

@RequestMapping("myinventory")

**public** **class** InventoryWebController {

@Autowired

**private** ProductService productService;

@Autowired

**private** InventoryRestClient inventoryRestClient;

@Value("${inventory.home.welcome}")

**private** String welcome;

@Value("${inventory.home.message}")

**private** String message;

@Value("${inventory.home.title}")

**private** String title;

@Value("${inventory.rest.url}")

**private** String restUrl;

@Value("${inventory.web.testing}")

**private** **boolean** testing;

@Value("${inventory.product.updateexpiry.success}")

**private** String updateExpirySuccess;

@Value("${inventory.product.deleteexpired.success}")

**private** String deleteExpirySuccess;

@Value("${inventory.product.applydiscount.success}")

**private** String applyDiscountSuccess;

@Value("${inventory.product.updateexpiry.failed}")

**private** String updateExpiryFailed;

@Value("${inventory.product.deleteexpired.failed}")

**private** String deleteExpiryFailed;

@Value("${inventory.product.applydiscount.failed}")

**private** String applyDiscountFailed;

@Value("${inventory.product.add.success}")

**private** String addSuccess;

@Value("${inventory.product.add.failed}")

**private** String addFailed;

@RequestMapping(value = { "products" }, method = RequestMethod.***GET***)

**public** ModelAndView getAllProducts(HttpServletRequest request) {

ModelAndView modelView = **new** ModelAndView("products");

// List<Product> products = productService.getAllProducts();

List<Product> products = Arrays.*asList*(inventoryRestClient

.getAllProducts(getRestUrl(request)));

modelView.addObject("productsList", products);

modelView.addObject("title", title);

modelView.addObject("message", message);

modelView.addObject("actionMessage", "");

**return** modelView;

}

@RequestMapping(value = { "sortProducts" }, method = RequestMethod.***GET***)

**public** ModelAndView sortProducts(HttpServletRequest request) {

ModelAndView modelView = **new** ModelAndView("products");

// List<Product> products = productService.getAllProducts();

List<Product> products = Arrays.*asList*(inventoryRestClient

.sortProductsByExpiry(getRestUrl(request)));

modelView.addObject("productsList", products);

modelView.addObject("title", title);

modelView.addObject("message", message);

modelView.addObject("actionMessage", "");

**return** modelView;

}

@RequestMapping(value = { "updateProductExpiry/{id}" }, method = RequestMethod.***POST***, produces = MediaType.***APPLICATION\_JSON\_VALUE***)

**public** ResponseEntity<?> updateProductExpiry(HttpServletRequest request,

@PathVariable("id") Integer id) {

// int productId = Integer.parseInt(id);

Product product = inventoryRestClient.updateProduct(

getRestUrl(request), id);

**return** ResponseEntity.*ok*(product);

}

@RequestMapping(value = { "applayDiscounts" }, method = RequestMethod.***POST***)

**public** ModelAndView applayDiscounts(HttpServletRequest request) {

ModelAndView modelView = **new** ModelAndView("products");

modelView.addObject("title", title);

modelView.addObject("message", message);

**boolean** success = inventoryRestClient

.applayDiscounts(getRestUrl(request));

**if** (success) {

modelView.addObject("actionMessage", applyDiscountSuccess);

} **else** {

modelView.addObject("actionMessage", applyDiscountFailed);

}

List<Product> products = Arrays.*asList*(inventoryRestClient

.getAllProducts(getRestUrl(request)));

modelView.addObject("productsList", products);

**return** modelView;

}

@RequestMapping(value = { "search" }, method = RequestMethod.***POST***)

**public** ModelAndView search(HttpServletRequest request,

@RequestParam String searchKey) {

ModelAndView modelView = **new** ModelAndView("search");

modelView.addObject("title", title);

modelView.addObject("message", "Search Products");

List<Product> products = **null**;

**try** {

products = Arrays.*asList*(inventoryRestClient.searchProducts(

getRestUrl(request), searchKey));

} **catch** (Exception e) {

}

modelView.addObject("actionMessage", "");

**if** (products == **null** || products.isEmpty()) {

modelView.addObject("actionMessage", "No Product Found");

}

modelView.addObject("productsList", products);

**return** modelView;

}

@RequestMapping(value = { "search" }, method = RequestMethod.***GET***)

**public** ModelAndView search(HttpServletRequest request) {

ModelAndView modelView = **new** ModelAndView("search");

modelView.addObject("title", title);

modelView.addObject("message", "Search Products");

modelView.addObject("actionMessage", "");

List<Product> products = Arrays.*asList*(inventoryRestClient

.getAllProducts(getRestUrl(request)));

modelView.addObject("productsList", products);

**return** modelView;

}

@RequestMapping(value = { "addProduct" }, method = RequestMethod.***GET***)

**public** ModelAndView addProduct(HttpServletRequest request) {

ModelAndView modelView = **new** ModelAndView("addProduct");

modelView.addObject("title", title);

modelView.addObject("message", "Create a new product");

modelView.addObject("actionMessage", "");

**return** modelView;

}

@RequestMapping(value = { "addProduct" }, method = RequestMethod.***POST***)

**public** ModelAndView addProduct(

HttpServletRequest request,

@ModelAttribute("addproductForm") com.cts.casestudy.inventory.entity.Product product) {

ModelAndView modelView = **new** ModelAndView("addProduct");

modelView.addObject("title", title);

modelView.addObject("message", "Create a new product");

product.setExpiryDate(**null**);

**boolean** success = inventoryRestClient.addProduct(getRestUrl(request),

product);

**if** (success) {

modelView.addObject("actionMessage", addSuccess);

} **else** {

modelView.addObject("actionMessage", addFailed);

}

**return** modelView;

}

@RequestMapping(value = { "deleteProduct/{id}" }, method = RequestMethod.***POST***)

**public** ModelAndView deleteProduct(HttpServletRequest request,

@PathVariable("id") Integer id) {

ModelAndView modelView = **new** ModelAndView("products");

modelView.addObject("title", title);

modelView.addObject("message", message);

**boolean** success = inventoryRestClient.deleteProducts(

getRestUrl(request), id);

**if** (success) {

modelView.addObject("actionMessage", "Successfully deleted the product");

} **else** {

modelView.addObject("actionMessage", "Failed to delete the product");

}

List<Product> products = Arrays.*asList*(inventoryRestClient

.getAllProducts(getRestUrl(request)));

modelView.addObject("productsList", products);

**return** modelView;

}

@RequestMapping(value = { "deleteExpired" }, method = RequestMethod.***POST***)

**public** ModelAndView deleteExpiredProducts(HttpServletRequest request) {

ModelAndView modelView = **new** ModelAndView("products");

modelView.addObject("title", title);

modelView.addObject("message", message);

**boolean** success = inventoryRestClient

.deleteExpiredProducts(getRestUrl(request));

**if** (success) {

modelView.addObject("actionMessage", deleteExpirySuccess);

} **else** {

modelView.addObject("actionMessage", deleteExpiryFailed);

}

List<Product> products = Arrays.*asList*(inventoryRestClient

.getAllProducts(getRestUrl(request)));

modelView.addObject("productsList", products);

**return** modelView;

}

**private** String getRestUrl(HttpServletRequest request) {

String restServiceUrl = "";

**if** (testing) {

String scheme = request.getScheme();

String host = request.getHeader("Host");

String contextPath = request.getContextPath();

restServiceUrl = scheme + "://" + host + contextPath;

} **else** {

restServiceUrl = restUrl;

}

**return** restServiceUrl;

}

}

1. UserController

**package** com.cts.casestudy.inventory.controller;

**import** java.util.Map;

**import** javax.servlet.http.HttpServletRequest;

**import** javax.servlet.http.HttpServletResponse;

**import** org.springframework.beans.factory.annotation.Autowired;

**import** org.springframework.beans.factory.annotation.Value;

**import** org.springframework.security.core.Authentication;

**import** org.springframework.security.core.context.SecurityContextHolder;

**import** org.springframework.security.web.authentication.logout.SecurityContextLogoutHandler;

**import** org.springframework.stereotype.Controller;

**import** org.springframework.ui.Model;

**import** org.springframework.validation.BindingResult;

**import** org.springframework.web.bind.annotation.ModelAttribute;

**import** org.springframework.web.bind.annotation.RequestMapping;

**import** org.springframework.web.bind.annotation.RequestMethod;

**import** org.springframework.web.bind.annotation.RequestParam;

**import** com.cts.casestudy.inventory.entity.User;

**import** com.cts.casestudy.inventory.service.SecurityService;

**import** com.cts.casestudy.inventory.service.UserService;

**import** com.cts.casestudy.inventory.validator.LoginValidator;

**import** com.cts.casestudy.inventory.validator.UserValidator;

@Controller

@RequestMapping("/")

**public** **class** UserController {

@Autowired

**private** UserService userService;

@Autowired

**private** SecurityService securityService;

@Autowired

**private** LoginValidator loginValidator;

@Autowired

**private** UserValidator userValidator;

@Value("${inventory.home.welcome}")

**private** String welcome;

@Value("${inventory.home.message}")

**private** String message;

@Value("${inventory.home.title}")

**private** String title;

@RequestMapping(value = "/login", method = RequestMethod.***GET***)

**public** String login(Model model,

@RequestParam(value = "error", required = **false**) String error,

@RequestParam(value = "logout", required = **false**) String logout) {

model.addAttribute("loginForm", **new** User());

model.addAttribute("error", "");

model.addAttribute("message", "");

**if** (error != **null**)

model.addAttribute("error",

"Your username and password is invalid.");

**if** (logout != **null**)

model.addAttribute("message",

"You have been logged out successfully.");

**return** "login";

}

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

**public** String login(@ModelAttribute("loginForm") User userForm,

BindingResult bindingResult, Model model) {

loginValidator.validate(userForm, bindingResult);

**if** (bindingResult.hasErrors()) {

**return** "redirect:/login?error";

}

securityService.login(userForm.getUsername(), userForm.getPassword());

**return** "redirect:/home";

}

@RequestMapping(value = "/logout", method = RequestMethod.***GET***)

**public** String logoutPage(HttpServletRequest request,

HttpServletResponse response) {

Authentication auth = SecurityContextHolder.*getContext*()

.getAuthentication();

**if** (auth != **null**) {

**new** SecurityContextLogoutHandler().logout(request, response, auth);

}

**return** "redirect:/login?logout";

}

@RequestMapping(value = {"/home" })

**public** String home(Map<String, Object> model) {

model.put("title", title);

model.put("message", welcome);

**return** "home";

}

@RequestMapping(value = "/registration", method = RequestMethod.***GET***)

**public** String registration(Model model) {

model.addAttribute("userForm", **new** User());

**return** "registration";

}

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

**public** String registration(@ModelAttribute("userForm") User userForm,

BindingResult bindingResult, Model model) {

userValidator.validate(userForm, bindingResult);

**if** (bindingResult.hasErrors()) {

**return** "registration";

}

userService.save(userForm);

securityService.login(userForm.getUsername(), userForm.getPassword());

**return** "redirect:/myinventory/home";

}

}

1. IinventoryDAO

**package** com.cts.casestudy.inventory.dao;

**import** java.util.List;

**import** com.cts.casestudy.inventory.entity.Product;

**import** com.cts.casestudy.inventory.exception.InventoryException;

**public** **interface** IinventoryDAO {

List<Product> getAllProducts();

Product getProductById(**int** articleId);

**void** addProduct(Product product);

**void** updateProduct(Product product);

**void** deleteProduct(Product product);

**boolean** productExists(String desc);

List<Product> searchProducts(String desc) **throws** InventoryException;

List<Product> sortProducts();

}

1. InventoryDAO

**package** com.cts.casestudy.inventory.dao;

**import** java.util.List;

**import** javax.persistence.EntityManager;

**import** javax.persistence.PersistenceContext;

**import** org.springframework.stereotype.Repository;

**import** org.springframework.transaction.annotation.Transactional;

**import** com.cts.casestudy.inventory.entity.Product;

**import** com.cts.casestudy.inventory.exception.InventoryException;

@Transactional

@Repository

**public** **class** InventoryDAO **implements** IinventoryDAO {

@PersistenceContext

**private** EntityManager entityManager;

@Override

**public** Product getProductById(**int** productId) {

**return** entityManager.find(Product.**class**, productId);

}

@SuppressWarnings("unchecked")

@Override

**public** List<Product> getAllProducts() {

String hql = "FROM Product as product ORDER BY product.id";

**return** (List<Product>) entityManager.createQuery(hql).getResultList();

}

@Override

**public** **void** addProduct(Product product) {

entityManager.persist(product);

}

@Override

**public** **void** updateProduct(Product product) {

entityManager.merge(product);

entityManager.flush();

}

@Override

**public** **void** deleteProduct(Product product) {

entityManager.remove(product);

}

**public** **boolean** productExists(String desc) {

String hql = "FROM Product as product WHERE product.description = ?";

**int** count = entityManager.createQuery(hql).setParameter(1, desc)

.getResultList().size();

**return** count > 0 ? **true** : **false**;

}

@SuppressWarnings("unchecked")

@Override

**public** List<Product> searchProducts(String searchKeyword)

**throws** InventoryException {

List<Product> products;

String hql = "FROM Product as product WHERE product.description like ?";

products = entityManager.createQuery(hql)

.setParameter(1, "%" + searchKeyword + "%").getResultList();

**if** (products == **null** || products.isEmpty()) {

**throw** **new** InventoryException("Item NotFound Exception");

}

**return** products;

}

@SuppressWarnings("unchecked")

@Override

**public** List<Product> sortProducts() {

String hql = "FROM Product as product ORDER BY product.expiryDate DESC";

**return** (List<Product>) entityManager.createQuery(hql).getResultList();

}

}

1. IUserDao

**package** com.cts.casestudy.inventory.dao;

**import** com.cts.casestudy.inventory.entity.User;

**public** **interface** IUserDao {

User getUserDetails(String userName);

}

1. UserDao

**package** com.cts.casestudy.inventory.dao;

**import** java.util.List;

**import** javax.persistence.EntityManager;

**import** javax.persistence.PersistenceContext;

**import** org.springframework.stereotype.Repository;

**import** org.springframework.transaction.annotation.Transactional;

**import** com.cts.casestudy.inventory.entity.User;

@Transactional

@Repository

**public** **class** UserDao **implements** IUserDao {

@PersistenceContext

**private** EntityManager entityManager;

@Override

**public** User getUserDetails(String userName) {

User user = **null**;

String hql = "FROM User as user WHERE user.userName = ?";

@SuppressWarnings("unchecked")

List<User> list = entityManager.createQuery(hql)

.setParameter(1, userName).getResultList();

**if** (list != **null** && !list.isEmpty()) {

user = list.get(0);

}

**return** user;

}

}

1. Product

**package** com.cts.casestudy.inventory.entity;

**import** java.io.Serializable;

**import** java.util.Date;

**import** javax.persistence.Column;

**import** javax.persistence.Entity;

**import** javax.persistence.GeneratedValue;

**import** javax.persistence.GenerationType;

**import** javax.persistence.Id;

**import** javax.persistence.Table;

**import** org.springframework.format.annotation.DateTimeFormat;

**import** com.cts.casestudy.inventory.util.JsonDateDeSerializer;

**import** com.cts.casestudy.inventory.util.JsonDateSerializer;

**import** com.fasterxml.jackson.databind.annotation.JsonDeserialize;

**import** com.fasterxml.jackson.databind.annotation.JsonSerialize;

@Entity

@Table(name = "products")

**public** **class** Product **implements** Serializable {

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

@Id

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

@Column(name = "id")

**private** **int** id;

@Column(name = "description")

**private** String description;

@Column(name = "weight")

**private** **int** weight;

@Column(name = "price")

**private** **double** price;

@Column(name = "manufacture\_date")

**private** Date manufactureDate;

@Column(name = "expire\_month")

**private** **int** expireMonth;

@Column(name = "expiry\_date")

@DateTimeFormat(pattern = "dd-MM-yyyy")

**private** Date expiryDate;

**public** Product() {

**super**();

}

**public** Product(String description, **int** weight, **double** price,

Date manufactureDate, **int** expireMonth) {

**super**();

**this**.description = description;

**this**.weight = weight;

**this**.price = price;

**this**.manufactureDate = manufactureDate;

**this**.expireMonth = expireMonth;

}

**public** **int** getId() {

**return** id;

}

**public** **void** setId(**int** id) {

**this**.id = id;

}

**public** String getDescription() {

**return** description;

}

**public** **void** setDescription(String description) {

**this**.description = description;

}

**public** **int** getWeight() {

**return** weight;

}

**public** **void** setWeight(**int** weight) {

**this**.weight = weight;

}

**public** **double** getPrice() {

**return** price;

}

**public** **void** setPrice(**double** price) {

**this**.price = price;

}

@JsonSerialize(using = JsonDateSerializer.**class**)

@DateTimeFormat(pattern = "dd-MM-yyyy")

@JsonDeserialize(using=JsonDateDeSerializer.**class**)

**public** Date getManufactureDate() {

**return** manufactureDate;

}

**public** **void** setManufactureDate(Date manufactureDate) {

**this**.manufactureDate = manufactureDate;

}

**public** **int** getExpireMonth() {

**return** expireMonth;

}

**public** **void** setExpireMonth(**int** expireMonth) {

**this**.expireMonth = expireMonth;

}

@JsonSerialize(using = JsonDateSerializer.**class**)

@JsonDeserialize(using=JsonDateDeSerializer.**class**)

**public** Date getExpiryDate() {

**return** expiryDate;

}

**public** **void** setExpiryDate(Date expiryDate) {

**this**.expiryDate = expiryDate;

}

@Override

**public** String toString() {

StringBuilder builder = **new** StringBuilder();

builder.append("Product [id=");

builder.append(id);

builder.append(", description=");

builder.append(description);

builder.append(", weight=");

builder.append(weight);

builder.append(", price=");

builder.append(price);

builder.append(", manufactureDate=");

builder.append(manufactureDate);

builder.append(", expireMonth=");

builder.append(expireMonth);

builder.append(", expiryDate=");

builder.append(expiryDate);

builder.append("]");

**return** builder.toString();

}

}

1. Role

**package** com.cts.casestudy.inventory.entity;

**import** java.util.Set;

**import** javax.persistence.Column;

**import** javax.persistence.Entity;

**import** javax.persistence.GeneratedValue;

**import** javax.persistence.GenerationType;

**import** javax.persistence.Id;

**import** javax.persistence.ManyToMany;

**import** javax.persistence.Table;

@Entity

@Table(name = "role")

**public** **class** Role {

**private** **int** id;

**private** String roleName;

**private** Set<User> users;

@Id

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

@Column(name = "id")

**public** **int** getId() {

**return** id;

}

**public** **void** setId(**int** id) {

**this**.id = id;

}

@Column(name = "rolename")

**public** String getRoleName() {

**return** roleName;

}

**public** **void** setRoleName(String roleName) {

**this**.roleName = roleName;

}

@ManyToMany(mappedBy = "roles")

**public** Set<User> getUsers() {

**return** users;

}

**public** **void** setUsers(Set<User> users) {

**this**.users = users;

}

}

1. User

**package** com.cts.casestudy.inventory.entity;

**import** java.io.Serializable;

**import** java.util.Set;

**import** javax.persistence.Column;

**import** javax.persistence.Entity;

**import** javax.persistence.FetchType;

**import** javax.persistence.Id;

**import** javax.persistence.JoinColumn;

**import** javax.persistence.JoinTable;

**import** javax.persistence.ManyToMany;

**import** javax.persistence.Table;

@Entity

@Table(name = "users")

**public** **class** User **implements** Serializable {

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

@Id

@Column(name = "username")

**private** String username;

@Column(name = "password")

**private** String password;

@Column(name = "full\_name")

**private** String fullName;

@Column(name = "country")

**private** String country;

@Column(name = "enabled")

**private** **short** enabled;

@ManyToMany(fetch = FetchType.***EAGER***)

@JoinTable(name = "user\_roles", joinColumns = @JoinColumn(name = "username"), inverseJoinColumns = @JoinColumn(name = "role\_id"))

**private** Set<Role> roles;

**public** String getUsername() {

**return** username;

}

**public** **void** setUsername(String userName) {

**this**.username = userName;

}

**public** String getPassword() {

**return** password;

}

**public** **void** setPassword(String password) {

**this**.password = password;

}

**public** String getFullName() {

**return** fullName;

}

**public** **void** setFullName(String fullName) {

**this**.fullName = fullName;

}

**public** String getCountry() {

**return** country;

}

**public** **void** setCountry(String country) {

**this**.country = country;

}

**public** **short** getEnabled() {

**return** enabled;

}

**public** **void** setEnabled(**short** enabled) {

**this**.enabled = enabled;

}

**public** Set<Role> getRoles() {

**return** roles;

}

**public** **void** setRoles(Set<Role> roles) {

**this**.roles = roles;

}

}

1. InventoryException

**package** com.cts.casestudy.inventory.exception;

**public** **class** InventoryException **extends** Exception {

/\*\*

\*

\*/

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

**public** InventoryException() {

**super**();

}

**public** InventoryException(String arg0, Throwable arg1, **boolean** arg2,

**boolean** arg3) {

**super**(arg0, arg1, arg2, arg3);

}

**public** InventoryException(String arg0, Throwable arg1) {

**super**(arg0, arg1);

}

**public** InventoryException(String arg0) {

**super**(arg0);

}

**public** InventoryException(Throwable arg0) {

**super**(arg0);

}

}

1. RoleRepository

**package** com.cts.casestudy.inventory.repository;

**import** org.springframework.data.jpa.repository.JpaRepository;

**import** com.cts.casestudy.inventory.entity.Role;

**public** **interface** RoleRepository **extends** JpaRepository<Role, Long>{

}

1. UserRepository

**package** com.cts.casestudy.inventory.repository;

**import** org.springframework.data.jpa.repository.JpaRepository;

**import** com.cts.casestudy.inventory.entity.User;

**public** **interface** UserRepository **extends** JpaRepository<User, Long> {

User findByUsername(String username);

}

1. InventoryRestClient

**package** com.cts.casestudy.inventory.restclient;

**import** org.apache.tomcat.util.codec.binary.Base64;

**import** org.springframework.beans.factory.annotation.Autowired;

**import** org.springframework.http.HttpEntity;

**import** org.springframework.http.HttpHeaders;

**import** org.springframework.http.HttpMethod;

**import** org.springframework.http.MediaType;

**import** org.springframework.http.ResponseEntity;

**import** org.springframework.security.core.userdetails.UserDetails;

**import** org.springframework.stereotype.Component;

**import** org.springframework.web.client.RestTemplate;

**import** com.cts.casestudy.inventory.entity.User;

**import** com.cts.casestudy.inventory.service.SecurityService;

**import** com.cts.casestudy.inventory.service.UserService;

**import** com.cts.casestudy.inventory.vo.Product;

@Component

**public** **class** InventoryRestClient {

@Autowired

**private** SecurityService securityService;

@Autowired

**private** UserService userService;

**private** HttpHeaders restAuth() {

UserDetails userDetails = securityService.findLoggedInUser();

User user = userService.findByUsername(userDetails.getUsername());

String plainClientCredentials = userDetails.getUsername() + ":" + user.getPassword();

String base64ClientCredentials = **new** String(

Base64.*encodeBase64*(plainClientCredentials.getBytes()));

HttpHeaders headers = **new** HttpHeaders();

headers.add("Authorization", "Basic " + base64ClientCredentials);

**return** headers;

}

**public** Product getProductById(String restUrl) {

// HttpHeaders headers = restAuth();

HttpHeaders headers = restAuth();

RestTemplate restTemplate = **new** RestTemplate();

headers.setContentType(MediaType.***APPLICATION\_JSON***);

String url = restUrl + "/inventory/product/{id}";

HttpEntity<String> requestEntity = **new** HttpEntity<String>(headers);

ResponseEntity<Product> responseEntity = restTemplate.exchange(url,

HttpMethod.***GET***, requestEntity, Product.**class**, 1);

**return** responseEntity.getBody();

}

**public** Product[] searchProducts(String restUrl, String searchKey) {

//HttpHeaders headers = new HttpHeaders();

HttpHeaders headers = restAuth();

RestTemplate restTemplate = **new** RestTemplate();

headers.setContentType(MediaType.***APPLICATION\_JSON***);

String url = restUrl + "/inventory/search/" + searchKey;

HttpEntity<String> requestEntity = **new** HttpEntity<String>(headers);

ResponseEntity<Product[]> responseEntity = restTemplate.exchange(url,

HttpMethod.***GET***, requestEntity, Product[].**class**);

Product[] products = responseEntity.getBody();

**return** products;

}

**public** **boolean** addProduct(String restUrl,

com.cts.casestudy.inventory.entity.Product product) {

**try** {

HttpHeaders headers = restAuth();

RestTemplate restTemplate = **new** RestTemplate();

headers.setContentType(MediaType.***APPLICATION\_JSON***);

String url = restUrl + "/inventory/product";

HttpEntity<com.cts.casestudy.inventory.entity.Product> requestEntity = **new** HttpEntity<com.cts.casestudy.inventory.entity.Product>(

product, headers);

ResponseEntity<Void> out = restTemplate.exchange(url,

HttpMethod.***POST***, requestEntity, Void.**class**);

**if** (out.getStatusCode().is2xxSuccessful()) {

**return** **true**;

}

} **catch** (Exception e) {

}

**return** **false**;

}

**public** Product[] getAllProducts(String restUrl) {

//HttpHeaders headers = restAuth();

HttpHeaders headers = restAuth();

RestTemplate restTemplate = **new** RestTemplate();

headers.setContentType(MediaType.***APPLICATION\_JSON***);

String url = restUrl + "/inventory/getAllproducts";

HttpEntity<String> requestEntity = **new** HttpEntity<String>(headers);

ResponseEntity<Product[]> responseEntity = restTemplate.exchange(url,

HttpMethod.***GET***, requestEntity, Product[].**class**);

Product[] products = responseEntity.getBody();

**return** products;

}

**public** Product[] sortProductsByExpiry(String restUrl) {

HttpHeaders headers = restAuth();

RestTemplate restTemplate = **new** RestTemplate();

headers.setContentType(MediaType.***APPLICATION\_JSON***);

String url = restUrl + "/inventory/sortProductsByExpiry";

HttpEntity<String> requestEntity = **new** HttpEntity<String>(headers);

ResponseEntity<Product[]> responseEntity = restTemplate.exchange(url,

HttpMethod.***GET***, requestEntity, Product[].**class**);

Product[] products = responseEntity.getBody();

**return** products;

}

**public** Product updateProduct(String restUrl, Integer id) {

HttpHeaders headers = restAuth();

RestTemplate restTemplate = **new** RestTemplate();

headers.setContentType(MediaType.***APPLICATION\_JSON***);

String url = restUrl + "/inventory/updateExpiry/" + id;

HttpEntity<String> requestEntity = **new** HttpEntity<String>(headers);

ResponseEntity<Product> response = restTemplate.exchange(url,

HttpMethod.***PUT***, requestEntity, Product.**class**, 3);

**if** (response.getStatusCode().is2xxSuccessful()) {

**return** response.getBody();

} **else** {

**return** **null**;

}

}

**public** **boolean** deleteProducts(String restUrl, Integer id) {

**try** {

HttpHeaders headers = restAuth();

RestTemplate restTemplate = **new** RestTemplate();

headers.setContentType(MediaType.***APPLICATION\_JSON***);

String url = restUrl + "/inventory/delete/" + id;

HttpEntity<Product> requestEntity = **new** HttpEntity<Product>(headers);

ResponseEntity<Void> out = restTemplate.exchange(url,

HttpMethod.***DELETE***, requestEntity, Void.**class**, 3);

**if** (out.getStatusCode().is2xxSuccessful()) {

**return** **true**;

}

} **catch** (Exception e) {

// **TODO**: handle exception

}

**return** **false**;

}

**public** **boolean** deleteExpiredProducts(String restUrl) {

**try** {

HttpHeaders headers = restAuth();

RestTemplate restTemplate = **new** RestTemplate();

headers.setContentType(MediaType.***APPLICATION\_JSON***);

String url = restUrl + "/inventory/deleteExpiryProduct";

HttpEntity<Product> requestEntity = **new** HttpEntity<Product>(headers);

ResponseEntity<Void> out = restTemplate.exchange(url,

HttpMethod.***DELETE***, requestEntity, Void.**class**, 3);

**if** (out.getStatusCode().is2xxSuccessful()) {

**return** **true**;

}

} **catch** (Exception e) {

// **TODO**: handle exception

}

**return** **false**;

}

**public** **boolean** applayDiscounts(String restUrl) {

RestTemplate restTemplate = **new** RestTemplate();

**try** {

HttpHeaders headers = restAuth();

HttpEntity<String> requestEntity = **new** HttpEntity<String>(headers);

String url = restUrl + "/inventory/applayDiscounts";

ResponseEntity<Void> out = restTemplate.exchange(url,

HttpMethod.***PUT***, requestEntity, Void.**class**, 3);

**if** (out.getStatusCode().is2xxSuccessful()) {

**return** **true**;

}

} **catch** (Exception e) {

}

**return** **false**;

}

}

1. ProductService

**package** com.cts.casestudy.inventory.service;

**import** java.util.List;

**import** org.springframework.security.access.annotation.Secured;

**import** com.cts.casestudy.inventory.entity.Product;

**import** com.cts.casestudy.inventory.exception.InventoryException;

**public** **interface** ProductService {

@Secured ({"ROLE\_ADMIN", "ROLE\_USER"})

List<Product> getAllProducts();

@Secured ({"ROLE\_ADMIN", "ROLE\_USER"})

Product getProductById(**int** productId);

@Secured ({"ROLE\_ADMIN"})

**boolean** addProduct(Product product);

@Secured ({"ROLE\_ADMIN"})

Product updateProductExpiry(**int** productId);

@Secured ({"ROLE\_ADMIN"})

**void** deleteProduct(**int** productId);

@Secured ({"ROLE\_ADMIN"})

**void** deleteExpiryProduct();

@Secured ({"ROLE\_ADMIN"})

**void** applayDiscounts();

@Secured ({"ROLE\_ADMIN", "ROLE\_USER"})

List<Product> searchProducts(String desc) **throws** InventoryException;

@Secured ({"ROLE\_ADMIN", "ROLE\_USER"})

List<Product> sortProducts();

}

1. ProductServiceImpl

**package** com.cts.casestudy.inventory.service;

**import** java.util.Calendar;

**import** java.util.Date;

**import** java.util.List;

**import** org.springframework.beans.factory.annotation.Autowired;

**import** org.springframework.stereotype.Service;

**import** com.cts.casestudy.inventory.dao.IinventoryDAO;

**import** com.cts.casestudy.inventory.entity.Product;

**import** com.cts.casestudy.inventory.exception.InventoryException;

@Service

**public** **class** ProductServiceImpl **implements** ProductService {

@Autowired

**private** IinventoryDAO productDAO;

@Override

**public** Product getProductById(**int** productId) {

Product obj = productDAO.getProductById(productId);

**return** obj;

}

@Override

**public** List<Product> getAllProducts() {

**return** productDAO.getAllProducts();

}

@Override

**public** **synchronized** **boolean** addProduct(Product product) {

**if** (productDAO.productExists(product.getDescription())) {

**return** **false**;

} **else** {

productDAO.addProduct(product);

**return** **true**;

}

}

@Override

**public** Product updateProductExpiry(**int** productId) {

Product product = productDAO.getProductById(productId);

**if** (product == **null**) {

**return** **null**;

}

Date manufactureDate = product.getManufactureDate();

**int** month = product.getExpireMonth();

Calendar myCal = Calendar.*getInstance*();

myCal.setTime(manufactureDate);

myCal.add(Calendar.***MONTH***, +month);

Date expiryDate = myCal.getTime();

product.setExpiryDate(expiryDate);

productDAO.updateProduct(product);

**return** product;

}

@Override

**public** **void** deleteProduct(**int** productId) {

Product product = productDAO.getProductById(productId);

**if** (product != **null**) {

productDAO.deleteProduct(product);

}

}

@Override

**public** **void** deleteExpiryProduct() {

List<Product> products = productDAO.getAllProducts();

**for** (Product product : products) {

Date now = **new** Date();

Date expiryDate = product.getExpiryDate();

**if** (now.after(expiryDate)) {

productDAO.deleteProduct(product);

}

}

}

@Override

**public** **void** applayDiscounts() {

List<Product> products = productDAO.getAllProducts();

**for** (Product product : products) {

Date now = **new** Date();

Date expiryDate = product.getExpiryDate();

Calendar startDate = Calendar.*getInstance*();

startDate.setTime(now);

Calendar endDate = Calendar.*getInstance*();

endDate.setTime(expiryDate);

**int** diffYear = endDate.get(Calendar.***YEAR***)

- startDate.get(Calendar.***YEAR***);

**int** diffMonth = diffYear \* 12 + endDate.get(Calendar.***MONTH***)

- startDate.get(Calendar.***MONTH***);

**if** (diffMonth <= 6) {

**double** exsitingPrice = product.getPrice();

**double** newPrice = (exsitingPrice \* 20) / 100;

product.setPrice(newPrice);

productDAO.updateProduct(product);

}

}

}

@Override

**public** List<Product> searchProducts(String desc) **throws** InventoryException {

**try** {

**return** productDAO.searchProducts(desc);

} **catch** (InventoryException ie) {

**throw** ie;

} **catch** (Exception e) {

**throw** **new** InventoryException("Item NotFound Exception");

}

}

@Override

**public** List<Product> sortProducts() {

**return** productDAO.sortProducts();

}

}

1. SecurityService

**package** com.cts.casestudy.inventory.service;

**import** org.springframework.security.core.userdetails.UserDetails;

**public** **interface** SecurityService {

UserDetails findLoggedInUser();

String findLoggedInUsername();

**void** login(String username, String password);

}

1. SecurityServiceImpl

**package** com.cts.casestudy.inventory.service;

**import** org.springframework.beans.factory.annotation.Autowired;

**import** org.springframework.security.authentication.AuthenticationManager;

**import** org.springframework.security.authentication.UsernamePasswordAuthenticationToken;

**import** org.springframework.security.core.context.SecurityContextHolder;

**import** org.springframework.security.core.userdetails.UserDetails;

**import** org.springframework.security.core.userdetails.UserDetailsService;

**import** org.springframework.stereotype.Service;

@Service

**public** **class** SecurityServiceImpl **implements** SecurityService {

@Autowired

**private** AuthenticationManager authenticationManager;

@Autowired

**private** UserDetailsService userDetailsService;

@Override

**public** UserDetails findLoggedInUser() {

Object userDetails = SecurityContextHolder.*getContext*()

.getAuthentication().getPrincipal();

**if** (userDetails **instanceof** UserDetails) {

**return** (UserDetails) userDetails;

}

**return** **null**;

}

@Override

**public** String findLoggedInUsername() {

Object userDetails = SecurityContextHolder.*getContext*()

.getAuthentication().getPrincipal();

**if** (userDetails **instanceof** UserDetails) {

**return** ((UserDetails) userDetails).getUsername();

}

**return** **null**;

}

@Override

**public** **void** login(String username, String password) {

UserDetails userDetails = userDetailsService

.loadUserByUsername(username);

UsernamePasswordAuthenticationToken usernamePasswordAuthenticationToken = **new** UsernamePasswordAuthenticationToken(

userDetails, password, userDetails.getAuthorities());

authenticationManager.authenticate(usernamePasswordAuthenticationToken);

**if** (usernamePasswordAuthenticationToken.isAuthenticated()) {

SecurityContextHolder.*getContext*().setAuthentication(

usernamePasswordAuthenticationToken);

}

}

}

1. UserDetailsServiceImpl

**package** com.cts.casestudy.inventory.service;

**import** java.util.HashSet;

**import** java.util.Set;

**import** org.springframework.beans.factory.annotation.Autowired;

**import** org.springframework.security.core.GrantedAuthority;

**import** org.springframework.security.core.authority.SimpleGrantedAuthority;

**import** org.springframework.security.core.userdetails.UserDetails;

**import** org.springframework.security.core.userdetails.UserDetailsService;

**import** org.springframework.security.core.userdetails.UsernameNotFoundException;

**import** org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;

**import** org.springframework.stereotype.Service;

**import** org.springframework.transaction.annotation.Transactional;

**import** com.cts.casestudy.inventory.entity.Role;

**import** com.cts.casestudy.inventory.entity.User;

**import** com.cts.casestudy.inventory.repository.UserRepository;

@Service

**public** **class** UserDetailsServiceImpl **implements** UserDetailsService {

@Autowired

**private** UserRepository userRepository;

@Autowired

**private** BCryptPasswordEncoder passwordEncoder;

@Override

@Transactional(readOnly = **true**)

**public** UserDetails loadUserByUsername(String username)

**throws** UsernameNotFoundException {

User user = userRepository.findByUsername(username);

Set<GrantedAuthority> grantedAuthorities = **new** HashSet<>();

**for** (Role role : user.getRoles()) {

grantedAuthorities.add(**new** SimpleGrantedAuthority(role

.getRoleName()));

}

**return** **new** org.springframework.security.core.userdetails.User(

user.getUsername(), passwordEncoder.encode(user.getPassword()), grantedAuthorities);

}

}

1. UserService

**package** com.cts.casestudy.inventory.service;

**import** com.cts.casestudy.inventory.entity.User;

**public** **interface** UserService {

**public** **void** save(User user);

**public** User findByUsername(String username);

}

1. UserServiceImpl

**package** com.cts.casestudy.inventory.service;

**import** java.util.HashSet;

**import** org.springframework.beans.factory.annotation.Autowired;

**import** org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;

**import** org.springframework.stereotype.Service;

**import** com.cts.casestudy.inventory.entity.User;

**import** com.cts.casestudy.inventory.repository.RoleRepository;

**import** com.cts.casestudy.inventory.repository.UserRepository;

@Service

**public** **class** UserServiceImpl **implements** UserService {

@Autowired

**private** UserRepository userRepository;

@Autowired

**private** RoleRepository roleRepository;

@Autowired

**private** BCryptPasswordEncoder passwordEncoder;

@Override

**public** **void** save(User user) {

user.setPassword(passwordEncoder.encode(user.getPassword()));

user.setRoles(**new** HashSet<>(roleRepository.findAll()));

userRepository.save(user);

}

@Override

**public** User findByUsername(String username) {

**return** userRepository.findByUsername(username);

}

}

1. JsonDateDeSerializer

**package** com.cts.casestudy.inventory.util;

**import** java.io.IOException;

**import** java.text.ParseException;

**import** java.text.SimpleDateFormat;

**import** java.util.Date;

**import** com.fasterxml.jackson.core.JsonParser;

**import** com.fasterxml.jackson.core.JsonProcessingException;

**import** com.fasterxml.jackson.databind.DeserializationContext;

**import** com.fasterxml.jackson.databind.JsonDeserializer;

**public** **class** JsonDateDeSerializer **extends** JsonDeserializer<Date> {

**private** SimpleDateFormat dateFormat = **new** SimpleDateFormat(

"dd-MM-yyyy");

@Override

**public** Date deserialize(JsonParser paramJsonParser,

DeserializationContext paramDeserializationContext)

**throws** IOException, JsonProcessingException {

String str = paramJsonParser.getText().trim();

**try** {

**return** dateFormat.parse(str);

} **catch** (ParseException e) {

}

**return** paramDeserializationContext.parseDate(str);

}

}

1. JsonDateSerializer

**package** com.cts.casestudy.inventory.util;

**import** java.io.IOException;

**import** java.text.SimpleDateFormat;

**import** java.util.Date;

**import** org.springframework.stereotype.Component;

**import** com.fasterxml.jackson.core.JsonGenerator;

**import** com.fasterxml.jackson.core.JsonProcessingException;

**import** com.fasterxml.jackson.databind.JsonSerializer;

**import** com.fasterxml.jackson.databind.SerializerProvider;

@Component

**public** **class** JsonDateSerializer **extends** JsonSerializer<Date> {

**private** **static** **final** SimpleDateFormat ***dateFormat*** = **new** SimpleDateFormat(

"dd-MM-yyyy");

@Override

**public** **void** serialize(Date date, JsonGenerator jsonGenerator,

SerializerProvider arg2) **throws** IOException,

JsonProcessingException {

**if** (date != **null**) {

String formattedDate = ***dateFormat***.format(date);

jsonGenerator.writeString(formattedDate);

}

}

}

1. LoginValidator

**package** com.cts.casestudy.inventory.validator;

**import** org.springframework.beans.factory.annotation.Autowired;

**import** org.springframework.stereotype.Component;

**import** org.springframework.validation.Errors;

**import** org.springframework.validation.ValidationUtils;

**import** org.springframework.validation.Validator;

**import** com.cts.casestudy.inventory.entity.User;

**import** com.cts.casestudy.inventory.service.UserService;

@Component

**public** **class** LoginValidator **implements** Validator {

@Autowired

**private** UserService userService;

@Override

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

**return** User.**class**.equals(aClass);

}

@Override

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

User user = (User) obj;

ValidationUtils.*rejectIfEmptyOrWhitespace*(errors, "username",

"user.not.empty");

ValidationUtils.*rejectIfEmptyOrWhitespace*(errors, "password",

"password.not.empty");

**if** (userService.findByUsername(user.getUsername()) == **null**) {

errors.rejectValue("username", "incorrect.credentials");

}

}

}

1. UserValidator

**package** com.cts.casestudy.inventory.validator;

**import** org.springframework.beans.factory.annotation.Autowired;

**import** org.springframework.stereotype.Component;

**import** org.springframework.validation.Errors;

**import** org.springframework.validation.ValidationUtils;

**import** org.springframework.validation.Validator;

**import** com.cts.casestudy.inventory.entity.User;

**import** com.cts.casestudy.inventory.service.UserService;

@Component

**public** **class** UserValidator **implements** Validator {

@Autowired

**private** UserService userService;

@Override

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

**return** User.**class**.equals(aClass);

}

@Override

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

User user = (User) obj;

ValidationUtils.*rejectIfEmptyOrWhitespace*(errors, "username",

"user.not.empty");

ValidationUtils.*rejectIfEmptyOrWhitespace*(errors, "password",

"password.not.empty");

**if** (userService.findByUsername(user.getUsername()) != **null**) {

errors.rejectValue("username", "duplicate.username");

}

}

}

1. Product

**package** com.cts.casestudy.inventory.vo;

**import** java.io.Serializable;

**public** **class** Product **implements** Serializable {

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

**private** **int** id;

**private** String description;

**private** **int** weight;

**private** **double** price;

**private** String manufactureDate;

**private** **int** expireMonth;

**private** String expiryDate;

**public** Product() {

**super**();

}

**public** Product(String description, **int** weight, **double** price,

String manufactureDate, **int** expireMonth) {

**super**();

**this**.description = description;

**this**.weight = weight;

**this**.price = price;

**this**.manufactureDate = manufactureDate;

**this**.expireMonth = expireMonth;

}

**public** **int** getId() {

**return** id;

}

**public** **void** setId(**int** id) {

**this**.id = id;

}

**public** String getDescription() {

**return** description;

}

**public** **void** setDescription(String description) {

**this**.description = description;

}

**public** **int** getWeight() {

**return** weight;

}

**public** **void** setWeight(**int** weight) {

**this**.weight = weight;

}

**public** **double** getPrice() {

**return** price;

}

**public** **void** setPrice(**double** price) {

**this**.price = price;

}

**public** String getManufactureDate() {

**return** manufactureDate;

}

**public** **void** setManufactureDate(String manufactureDate) {

**this**.manufactureDate = manufactureDate;

}

**public** **int** getExpireMonth() {

**return** expireMonth;

}

**public** **void** setExpireMonth(**int** expireMonth) {

**this**.expireMonth = expireMonth;

}

**public** String getExpiryDate() {

**if**(expiryDate == **null**){

expiryDate = "";

}

**return** expiryDate;

}

**public** **void** setExpiryDate(String expiryDate) {

**this**.expiryDate = expiryDate;

}

@Override

**public** String toString() {

StringBuilder builder = **new** StringBuilder();

builder.append("Product [id=");

builder.append(id);

builder.append(", description=");

builder.append(description);

builder.append(", weight=");

builder.append(weight);

builder.append(", price=");

builder.append(price);

builder.append(", manufactureDate=");

builder.append(manufactureDate);

builder.append(", expireMonth=");

builder.append(expireMonth);

builder.append(", expiryDate=");

builder.append(expiryDate);

builder.append("]");

**return** builder.toString();

}

}

1. InventoryControllerTestConfig

**package** com.cts.casestudy.inventory.controller;

**import** org.springframework.context.annotation.Bean;

**import** com.cts.casestudy.inventory.dao.IinventoryDAO;

**import** com.cts.casestudy.inventory.dao.InventoryDAO;

**import** com.cts.casestudy.inventory.service.ProductService;

**import** com.cts.casestudy.inventory.service.ProductServiceImpl;

**public** **class** InventoryControllerTestConfig {

@Bean

ProductService getProductService() {

**return** **new** ProductServiceImpl();

}

@Bean

IinventoryDAO getProductDAO() {

**return** **new** InventoryDAO();

}

/\*

\* @Bean public RestTemplate restTemplate() { return new RestTemplate(); }

\*/

}

1. InventoryControllerTest

**package** com.cts.casestudy.inventory.controller;

**import** **static** org.mockito.Mockito.*when*;

**import** **static** org.springframework.test.web.servlet.request.MockMvcRequestBuilders.*get*;

**import** **static** org.springframework.test.web.servlet.result.MockMvcResultMatchers.*jsonPath*;

**import** java.util.Date;

**import** org.junit.Before;

**import** org.junit.Test;

**import** org.junit.runner.RunWith;

**import** org.mockito.InjectMocks;

**import** org.mockito.Mock;

**import** org.springframework.beans.factory.annotation.Autowired;

**import** org.springframework.boot.test.context.SpringBootTest;

**import** org.springframework.test.context.ContextConfiguration;

**import** org.springframework.test.context.junit4.SpringRunner;

**import** org.springframework.test.web.servlet.MockMvc;

**import** org.springframework.test.web.servlet.MvcResult;

**import** org.springframework.test.web.servlet.request.MockMvcRequestBuilders;

**import** org.springframework.test.web.servlet.setup.MockMvcBuilders;

**import** org.springframework.web.context.WebApplicationContext;

**import** com.cts.casestudy.inventory.dao.IinventoryDAO;

**import** com.cts.casestudy.inventory.entity.Product;

**import** com.cts.casestudy.inventory.service.ProductService;

@RunWith(SpringRunner.**class**)

@SpringBootTest

// @ContextConfiguration(classes = { InventoryControllerTestConfig.class })

@ContextConfiguration(classes = { InventoryControllerTestConfig.**class** })

// @WebMvcTest(value = InventoryController.class, secure = false)

**public** **class** InventoryControllerTest {

/\*

\* @Autowired private RestTemplate restTemplate;

\*/

@Autowired

**private** WebApplicationContext webApplicationContext;

**private** MockMvc mockMvc;

@Autowired

**private** ProductService productService;

@Mock

**private** IinventoryDAO productDAO;

/\*

\* @Mock private IinventoryDAO productDAO;

\*/

/\*@InjectMocks

InventoryController inventoryController;\*/

@Before

**public** **void** setup() {

mockMvc = MockMvcBuilders.*webAppContextSetup*(webApplicationContext)

.build();

}

/\*

\* @Autowired private MockMvc mockMvc;

\*

\* @MockBean private IProductService productService;

\*/

@Test

**public** **void** testGetProductById() **throws** Exception {

Product reqProduct = **new** Product("Test Product", 200, 12.5, **new** Date(),

3);

/\* given(this.productService.getProductById(1)).willReturn(reqProduct); \*/

// when(this.productDAO.getProductById(1)).thenReturn(reqProduct);

*when*(**this**.productService.getProductById(1)).thenReturn(reqProduct);

MvcResult result = mockMvc.perform(

MockMvcRequestBuilders.*get*("/product/1")).andReturn();

mockMvc.perform(*get*("/product/1"))

// .andExpect(status().isOk())

/\*

\* .andExpect(

\* content().contentType("application/json;charset=UTF-8"))

\*/

.andExpect(*jsonPath*("$.description").value("Test Product"))

.andExpect(*jsonPath*("$.weight").value(200))

.andExpect(*jsonPath*("$.price").value("12.5"))

.andExpect(*jsonPath*("$.expireMonth").value(3));

}

/\*

\* @Test public void testGetAllProducts() { fail("Not yet implemented"); }

\*

\* @Test public void testSearchProducts() { fail("Not yet implemented"); }

\*

\* @Test public void testAddProduct() { fail("Not yet implemented"); }

\*

\* @Test public void testUpdateProductExpiry() {

\* fail("Not yet implemented"); }

\*

\* @Test public void testApplayDiscounts() { fail("Not yet implemented"); }

\*

\* @Test public void testDelete() { fail("Not yet implemented"); }

\*

\* @Test public void testDeleteExpiryProduct() {

\* fail("Not yet implemented"); }

\*

\* @Test public void testSortProducts() { fail("Not yet implemented"); }

\*/

}

1. header.html

<!DOCTYPE HTML>

<head>

<title>{{title}}</title>

<meta http-equiv=*"Content-Type"* content=*"text/html; charset=UTF-8"* />

<link rel=*"stylesheet"* type=*"text/css"* href=*"webjars/bootstrap/3.3.7/css/bootstrap.min.css"* />

<link rel=*"stylesheet"* href=*"/css/main.css?version=1"* />

<script type=*"text/javascript"* src=*"/webjars/bootstrap/3.3.7/js/bootstrap.min.js"*></script>

<script type=*"text/javascript"* src=*"/webjars/jquery/2.2.4/jquery.min.js"*></script>

</head>

<body>

<nav class=*"navbar navbar-inverse"*>

<div class=*"container"*>

<div class=*"navbar-header"*>

<a class=*"navbar-brand"* href=*"/home"*>Home</a> |

<a class=*"navbar-brand"* href=*"/myinventory/products"*>Products</a> |

<a class=*"navbar-brand"* href=*"/myinventory/search"*>Search</a> |

<a class=*"navbar-brand"* href=*"/myinventory/addProduct"*>AddProduct</a> |

<a class=*"navbar-brand"* href=*"/logout"*>Logout</a>

</div>

</div>

</nav>

<!-- this is header -->

1. footer.html

<!-- this is footer -->

</body>

</html>

1. products.html

{{>layout/header}}

<div class=*"container"*>

<div class=*"starter-template"*>

<h1>{{title}}</h1>

</div>

<div class=*"starter-template"*>

<h2>{{message}}</h2>

<h4 style="color: *red*;">{{actionMessage}}</h4>

<form action=*""* method=*"POST"* name=*"inventoryForm"*>

<table class=*"producttable"*>

<tr>

<td><input type=*"button"* class=*"navbar-brand"*

onclick="submitInvForm('/myinventory/applayDiscounts')" value=*"Apply Discounts"* /></td>

<td><input type=*"button"* class=*"navbar-brand"*

onclick="submitInvForm('/myinventory/deleteExpired')"

value=*"Delete Expired Products"* /></td>

<td><a class=*"navbar-brand"* href=*"/myinventory/sortProducts"*> Sort

Products by Expiry</a></td>

</tr>

</table>

<table class=*"producttable"*>

<tr>

<td height=*"20"*>&nbsp;</td>

</tr>

</table>

<table class=*"producttable"* border=*"1"*>

<tr>

<th width=*"10%"*>Id</th>

<th width=*"30%"*>Description</th>

<th width=*"10%"*>Weight</th>

<th width=*"10%"*>Price</th>

<th width=*"10%"*>Manufacture Date</th>

<th width=*"10%"*>Expiry Month</th>

<th width=*"10%"*>Expiry Date</th>

<th width=*"10%"*>Update Expiry</th>

<th width=*"10%"*>Delete Product</th>

</tr>

{{#productsList}}

<tr>

<td>{{id}}</td>

<td>{{description}}</td>

<td>{{weight}}</td>

<td>{{price}}</td>

<td>{{manufactureDate}}</td>

<td>{{expireMonth}}</td>

<td id=*"expiryDate{{id}}"*>{{expiryDate}}</td>

<td><a class=*"navbar-brand"* href=*"javascript:void(0);"*

onclick="updateExpiry('{{id}}');">Update</a></td>

<td><a class=*"navbar-brand"* href=*"javascript:void(0);"*

onclick="confirm('Confirm to Delete');submitInvForm('/myinventory/deleteProduct/{{id}}');">Delete</a></td>

</tr>

{{/productsList}}

</table>

</form>

</div>

</div>

<!-- /.container -->

<script type=*"text/javascript"*>

**function** submitInvForm(formActrion) {

document.inventoryForm.action = formActrion;

document.inventoryForm.submit();

}

**function** updateExpiry(id) {

$.ajax({

type : "POST",

contentType : "application/json",

url : "updateProductExpiry/" + id,

data : {

"id" : id

},

dataType : 'json',

cache : **false**,

timeout : 600000,

success : **function**(data) {

$('#expiryDate' + id).html(data.expiryDate);

console.log("SUCCESS : ", data);

},

error : **function**(e) {

alert("Failed to Update the expiry date");

}

});

}

</script>

{{>layout/footer}}

1. home.html

{{>layout/header}}

<div class=*"container"*>

<div class=*"starter-template"*>

<h1>{{title}}</h1>

<h2>

{{message}}

</h2>

</div>

</div>

<!-- /.container -->

{{>layout/footer}}

1. addProduct.html

{{>layout/header}}

<div class=*"container"*>

<div class=*"starter-template"*>

<h1>{{title}}</h1>

</div>

<div class=*"starter-template"*>

<h2>Create a new product</h2>

<h4 style="color: *red*;">{{actionMessage}}</h4>

<form action=*"addProduct"* method=*"POST"* name=*"addproductForm"*>

<table class=*"producttable"*>

<tr>

<td>Description</td>

<td><input type=*"text"* name=*"description"* value=*""*></td>

</tr>

<tr>

<td>Weight</td>

<td><input type=*"text"* name=*"weight"* value=*""*></td>

</tr>

<tr>

<td>Price</td>

<td><input type=*"text"* name=*"price"* value=*""*></td>

</tr>

<tr>

<td>Manufacture Date(dd-MM-yyyy)</td>

<td><input type=*"text"* name=*"manufactureDate"* value=*""*></td>

</tr>

<tr>

<td>Expiry Month</td>

<td><input type=*"text"* name=*"expireMonth"* value=*""*></td>

</tr>

<tr>

<td colspan=*"2"*><input type=*"submit"* value=*"Submit"*> <input

type=*"reset"*></td>

</tr>

</table>

</form>

</div>

</div>

<!-- /.container -->

{{>layout/footer}}

1. search.html

{{>layout/header}}

<div class=*"container"*>

<div class=*"starter-template"*>

<h1>{{title}}</h1>

</div>

<div class=*"starter-template"*>

<h2>{{message}}</h2>

<h4 style="color: *red*;">{{actionMessage}}</h4>

<form action=*""* method=*"POST"* name=*"inventoryForm"*>

<table class=*"producttable"*>

<tr>

<td><input type=*"text"* name=*"searchKey"* value=*""*></td>

<td><input type=*"button"* class=*"navbar-brand"*

onclick="submitInvForm('search')"

value=*"Search Product"* /></td>

</tr>

</table>

<table class=*"producttable"* border=*"1"*>

<tr>

<th width=*"10%"*>Id</th>

<th width=*"40%"*>Description</th>

<th width=*"10%"*>Weight</th>

<th width=*"10%"*>Price</th>

<th width=*"10%"*>Manufacture Date</th>

<th width=*"10%"*>Expiry Month</th>

<th width=*"10%"*>Expiry Date</th>

<th width=*"10%"*>Update Expiry</th>

</tr>

{{#productsList}}

<tr>

<td>{{id}}</td>

<td>{{description}}</td>

<td>{{weight}}</td>

<td>{{price}}</td>

<td>{{manufactureDate}}</td>

<td>{{expireMonth}}</td>

<td id=*"expiryDate{{id}}"*>{{expiryDate}}</td>

<td><a class=*"navbar-brand"* href=*"javascript:void(0);"*

onclick="updateExpiry('{{id}}');">Update</a></td>

</tr>

{{/productsList}}

</table>

</form>

</div>

</div>

<!-- /.container -->

<script type=*"text/javascript"*>

**function** submitInvForm(formActrion) {

document.inventoryForm.action = formActrion;

document.inventoryForm.submit();

}

</script>

{{>layout/footer}}

1. login.html

<html>

<head>

<title>Login Page</title>

</head>

<body>

<h1>User Login Form</h1>

<div style="color: *red*;">{{error}}</div>

<div style="color: *red*;">{{message}}</div>

<div id=*"login-box"*>

<form name=*'loginForm'* action=*"/login"* method=*'POST'*>

<table>

<tr>

<td>User:</td>

<td><input type=*'text'* name=*'username'*></td>

</tr>

<tr>

<td>Password:</td>

<td><input type=*'password'* name=*'password'* /></td>

</tr>

<tr>

<td colspan=*'2'*><input name=*"submit"* type=*"submit"*

value=*"submit"* /></td>

</tr>

</table>

</form>

</div>

</body>

</html>

1. application.properties

#spring.datasource.driver-class-name=com.mysql.jdbc.Driver

spring.datasource.url=jdbc:mysql://localhost:3306/inventory

spring.datasource.username=root

spring.datasource.password=root

spring.datasource.tomcat.max-wait=20000

spring.datasource.tomcat.max-active=50

spring.datasource.tomcat.max-idle=20

spring.datasource.tomcat.min-idle=15

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

spring.jpa.properties.hibernate.id.new\_generator\_mappings = false

spring.jpa.properties.hibernate.format\_sql = true

logging.level.org.hibernate.SQL=DEBUG

logging.level.org.hibernate.type.descriptor.sql.BasicBinder=TRACE

inventory.rest.url: http://localhost:8080/inventory/

inventory.web.testing: true

inventory.home.title: Inventory Management System

inventory.home.welcome: Welcome to Manu Inventory Management System

inventory.home.message: All Products

inventory.product.updateexpiry.success= Successfully updated the expiry date

inventory.product.deleteexpired.success= Successfully deleted the expired product

inventory.product.applydiscount.success= Successfully apply discounts to products

inventory.product.updateexpiry.failed= Failed to updated the expiry date

inventory.product.deleteexpired.failed= Failed to deleted the expired product

inventory.product.applydiscount.failed= Failed to apply discounts to products

inventory.product.add.success: Product Added Successfully

inventory.product.add.failed: Failed to add the product

1. validation.properties

user.not.empty=User name not empty

password.not.empty=Password not empty

duplicate.username=Duplicate User Name.

incorrect.credentials=Incorrect User Name or Password.

1. Pom.xml

<?xml version=*"1.0"* encoding=*"UTF-8"*?>

<project xmlns=*"http://maven.apache.org/POM/4.0.0"* xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*

xsi:schemaLocation=*"http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd"*>

<modelVersion>4.0.0</modelVersion>

<groupId>com.cts.casestudy.inventroy</groupId>

<artifactId>InventoryManagementSystem</artifactId>

<version>0.0.1-SNAPSHOT</version>

<packaging>jar</packaging>

<name>Inventory Management System</name>

<description>Inventory Management System</description>

<parent>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-parent</artifactId>

<version>1.5.2.RELEASE</version>

</parent>

<properties>

<java.version>1.8</java.version>

</properties>

<dependencies>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-web</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-security</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-data-jpa</artifactId>

</dependency>

<dependency>

<groupId>mysql</groupId>

<artifactId>mysql-connector-java</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-mustache</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-test</artifactId>

<scope>test</scope>

</dependency>

<!-- hot swapping, live reload -->

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-devtools</artifactId>

<optional>false</optional>

</dependency>

<!-- Optional, for bootstrap -->

<dependency>

<groupId>org.webjars</groupId>

<artifactId>bootstrap</artifactId>

<version>3.3.7</version>

</dependency>

<dependency>

<groupId>org.webjars</groupId>

<artifactId>jquery</artifactId>

<version>2.2.4</version>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-maven-plugin</artifactId>

</plugin>

</plugins>

</build>

</project>