JAVA：

package info.smartkit.eip.activemq;

import javax.jms.Connection;

import javax.jms.ConnectionFactory;

import javax.jms.Destination;

import javax.jms.JMSException;

import javax.jms.Message;

import javax.jms.MessageConsumer;

import javax.jms.Session;

import javax.jms.TextMessage;

import info.smartkit.eip.settings.JMSSetting;

import org.apache.activemq.ActiveMQConnectionFactory;

import org.apache.logging.log4j.LogManager;

import org.apache.logging.log4j.Logger;

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

//@see: http://www.coderpanda.com/jms-example-using-apache-activemq/

public class ActivemqReceiver

{

private ConnectionFactory factory = null;

private Connection connection = null;

private Session session = null;

private Destination destination = null;

private MessageConsumer consumer = null;

private static Logger LOG = LogManager.getLogger(ActivemqReceiver.class);

@Autowired

private JMSSetting jmsSetting;

public ActivemqReceiver()

{

}

// private static ActivemqReceiver instance = null;

//

// protected ActivemqReceiver(String queueName) {

// // Exists only to defeat instantiation.

// this.queueName = queueName;

// }

//

// public static ActivemqReceiver getInstance(String queueName) {

// if (instance == null) {

// instance = new ActivemqReceiver(queueName);

// }

// return instance;

// }

public void receiveMessage()

{

try {

factory = new ActiveMQConnectionFactory(jmsSetting.getBrokerUrl());

connection = factory.createConnection();

connection.start();

session = connection.createSession(false, Session.AUTO\_ACKNOWLEDGE);

destination = session.createQueue(ActivemqVariables.queueName);

consumer = session.createConsumer(destination);

Message message = consumer.receive();

if (message instanceof TextMessage) {

TextMessage text = (TextMessage) message;

// System.out.println("Message is : " + text.getText());

LOG.info("Received activemq message is : " + text.getText() + ",queueName:"

+ ActivemqVariables.queueName);

}

} catch (JMSException e) {

e.printStackTrace();

}

}

}

package info.smartkit.eip.activemq;

import javax.jms.Connection;

import javax.jms.ConnectionFactory;

import javax.jms.Destination;

import javax.jms.JMSException;

import javax.jms.MessageProducer;

import javax.jms.Session;

import javax.jms.TextMessage;

import org.apache.activemq.ActiveMQConnectionFactory;

import org.apache.logging.log4j.LogManager;

import org.apache.logging.log4j.Logger;

//@see: http://www.coderpanda.com/jms-example-using-apache-activemq/

public class ActivemqSender

{

private ConnectionFactory factory = null;

private Connection connection = null;

private Session session = null;

private Destination destination = null;

private MessageProducer producer = null;

private static Logger LOG = LogManager.getLogger(ActivemqSender.class);

//

public ActivemqSender()

{

//

try {

factory = new ActiveMQConnectionFactory(ActivemqVariables.brokerUrl);

connection = factory.createConnection();

connection.start();

LOG.info("Start activemq success with queueName:" + ActivemqVariables.queueName);

} catch (JMSException e) {

// e.printStackTrace();

LOG.error(e.getMessage());

}

}

// private static ActivemqSender instance = null;

//

// protected ActivemqSender(String queueName) {

// // Exists only to defeat instantiation.

// this.queueName = queueName;

// }

//

// public static ActivemqSender getInstance(String queueName) {

// if (instance == null) {

// instance = new ActivemqSender(queueName);

// }

// return instance;

// }

public void sendMessage(String value)

{

try {

session = connection.createSession(false, Session.AUTO\_ACKNOWLEDGE);

destination = session.createQueue(ActivemqVariables.queueName);

producer = session.createProducer(destination);

TextMessage message = session.createTextMessage();

// message.setText("Hello ...This is a sample message..sending from FirstClient");

message.setText(value);

producer.send(message);

// System.out.println("Sent: " + message.getText());

LOG.info("Sent activemq message: " + message.getText() + ",queueName:" + ActivemqVariables.queueName);

} catch (JMSException e) {

// e.printStackTrace();

LOG.error(e.getMessage());

}

}

}

/\*

\* Copyright 2015 the original author or authors.

\*

\* Licensed under the Apache License, Version 2.0 (the "License");

\* you may not use this file except in compliance with the License.

\* You may obtain a copy of the License at

\*

\* http://www.apache.org/licenses/LICENSE-2.0

\*

\* Unless required by applicable law or agreed to in writing, software

\* distributed under the License is distributed on an "AS IS" BASIS,

\* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.

\* See the License for the specific language governing permissions and

\* limitations under the License.

\*

\* All rights reserved.

\*/

package info.smartkit.eip.activemq;

/\*\*

\* Variables stub for ActiveMQ

\*

\* @author yangboz

\*/

public class ActivemqVariables

{

// Static variables storage here:

// Common channel/topic name

static public String channelName = null;

// Unique queue name

static public String queueName = null;

// @Autowired

// private JMSSetting jmsSetting;

static public String brokerUrl = null;

}

package info.smartkit.eip.activiti;

import org.activiti.engine.delegate.DelegateExecution;

import org.activiti.engine.delegate.ExecutionListener;

import org.apache.logging.log4j.LogManager;

import org.apache.logging.log4j.Logger;

import org.json.simple.JSONObject;

@SuppressWarnings("serial")

public class ActivitiExecutionListener implements ExecutionListener

{

private static Logger LOG = LogManager.getLogger(ActivitiExecutionListener.class);

@SuppressWarnings({"unchecked"})

@Override

public void notify(DelegateExecution execution) throws Exception

{

// execution.setVariable("variableSetInExecutionListener",

// "firstValue");

// execution.setVariable("eventReceived", execution.getEventName());

try {

JSONObject jsonObj = new JSONObject();

jsonObj.put("ProcessInstanceId", execution.getProcessInstanceId().toString());

jsonObj.put("event", execution.getEventName());

jsonObj.put("execution", execution.toString());

jsonObj.put("activityId", execution.getCurrentActivityId());

jsonObj.put("activityName", execution.getCurrentActivityName());

jsonObj.put("variables", execution.getVariables().toString());

LOG.debug("Activiti execution:" + jsonObj.toString());

// Connect to ActiveMQ to send message.

// ActivemqSender sender = new ActivemqSender("SAMPLEQUEUE");

// ActivemqSender sender =

// ActivemqSender.getInstance("SAMPLEQUEUE");

// sender.sendMessage(execution.getEventName());

// TODO:Watch "reimbursementApproved=false", to toggle update the expense status to Reject.

} catch (Error err) {

LOG.error(err.toString());

}

}

}

package info.smartkit.eip.activiti;

import info.smartkit.eip.activemq.ActivemqSender;

import org.activiti.engine.delegate.DelegateTask;

import org.activiti.engine.delegate.TaskListener;

import org.apache.logging.log4j.LogManager;

import org.apache.logging.log4j.Logger;

import org.json.simple.JSONObject;

import info.smartkit.eip.activemq.ActivemqVariables;

@SuppressWarnings("serial")

public class ActivitiTaskListener implements TaskListener

{

//

private static Logger LOG = LogManager.getLogger(ActivitiTaskListener.class);

@SuppressWarnings({"unchecked"})

@Override

public void notify(DelegateTask delegateTask)

{

try {

JSONObject jsonObj = new JSONObject();

jsonObj.put("taskId", delegateTask.getId());

jsonObj.put("assignee", delegateTask.getAssignee());

jsonObj.put("category", delegateTask.getCategory());

jsonObj.put("candidates", delegateTask.getCandidates().toString());

// LOG info.

LOG.debug("Activiti task:" + jsonObj.toString());

// Connect to ActiveMQ to send message.

ActivemqSender sender = new ActivemqSender();

// Unique the queue name

ActivemqVariables.queueName = ActivemqVariables.channelName + "/" + delegateTask.getAssignee();

LOG.debug("ActivemqSender:" + sender.toString());

// Then send corresponding message.

sender.sendMessage(jsonObj.toJSONString());

} catch (Error err) {

LOG.error(err.toString());

}

}

}

package info.smartkit.eip;

import info.smartkit.eip.activemq.ActivemqVariables;

import info.smartkit.eip.config.PropertiesInitializer;

import info.smartkit.eip.settings.WorkflowSetting;

import org.activiti.engine.ProcessEngine;

import org.activiti.engine.ProcessEngines;

import org.activiti.engine.RepositoryService;

import org.apache.logging.log4j.LogManager;

import org.apache.logging.log4j.Logger;

import org.springframework.boot.autoconfigure.EnableAutoConfiguration;

import org.springframework.boot.builder.SpringApplicationBuilder;

import org.springframework.boot.context.web.SpringBootServletInitializer;

import org.springframework.context.ApplicationContext;

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.ComponentScan;

import org.springframework.context.annotation.Configuration;

import org.springframework.context.annotation.Import;

import org.springframework.context.annotation.ImportResource;

import org.springframework.context.annotation.PropertySource;

import org.springframework.context.annotation.PropertySources;

import org.springframework.data.jpa.repository.config.EnableJpaRepositories;

import org.springframework.data.rest.webmvc.config.RepositoryRestMvcConfiguration;

import org.springframework.orm.jpa.support.PersistenceAnnotationBeanPostProcessor;

import org.springframework.web.multipart.commons.CommonsMultipartResolver;

@Configuration

//

@PropertySources({@PropertySource(value = "classpath:application-${spring.profiles.active}.properties")})

//

@ComponentScan("info.smartkit.eip")

// @EnableWebSecurity

@EnableAutoConfiguration

// @EnableAutoConfiguration(exclude={WebSocketAutoConfiguration.class,JpaProcessEngineAutoConfiguration.class})

//

@ImportResource("classpath:activiti-standalone-context-${spring.profiles.active}.xml")

// @see: http://spring.io/guides/gs/accessing-data-rest/

@EnableJpaRepositories

//

@Import(RepositoryRestMvcConfiguration.class)

//

public class Application extends SpringBootServletInitializer {

//

private static Logger LOG = LogManager.getLogger(Application.class);

//

private static Class<Application> applicationClass = Application.class;

//

@Override

protected SpringApplicationBuilder configure(SpringApplicationBuilder application) {

//

return application.sources(applicationClass);

}

//

public static void main(String[] args) throws InterruptedException {

//

ApplicationContext context =

new SpringApplicationBuilder(applicationClass).initializers(new PropertiesInitializer()).run(args);

// SpringApplication.run(applicationClass, args);

LOG.info("ApplicationContext:" + context.getDisplayName() + context.getStartupDate());

//

workflowInitialization();

//

// System.setProperty("javax.xml.parsers.SAXParserFactory","com.sun.org.apache.xerces.internal.jaxp.SAXParserFactoryImpl");

//

System.setProperty("javax.xml.parsers.SAXParserFactory", "org.apache.xerces.jaxp.SAXParserFactoryImpl");

}

//

public static void workflowInitialization() {

//

LOG.info("workflowInitialization..." + WorkflowSetting.getInstance().getBpmn());

// Deploying the process here,avoid duplication to see:

// http://forums.activiti.org/content/duplicate-deployment-processes

ProcessEngine processEngine = ProcessEngines.getDefaultProcessEngine();

RepositoryService repositoryService = processEngine.getRepositoryService();

// Setting the process deployment with configuration.

LOG.info("workflowSetting.getName():" + WorkflowSetting.getInstance().getName() + ",workflowSetting.getBpmn():"

+ WorkflowSetting.getInstance().getBpmn() + ",workflowSetting.getImage():"

+ WorkflowSetting.getInstance().getImage());

// @see:

repositoryService.createDeployment().addClasspathResource(WorkflowSetting.getInstance().getBpmn())

.addClasspathResource(WorkflowSetting.getInstance().getImage()).enableDuplicateFiltering()

.name(WorkflowSetting.getInstance().getName()).deploy();

// Log information

LOG.info("Process definitions: " + repositoryService.createProcessDefinitionQuery().list().toString());

LOG.info("Number of process definitions: " + repositoryService.createProcessDefinitionQuery().count());

/\*

\* //Starting a process instance Map<String,Object> variables = new HashMap<String,Object>();

\* variables.put("employeeName", "employee1"); variables.put("amountOfMoney", (long)99.8);

\* variables.put("reimbursmentMotivation", "Need reimbursement for taxi."); // RuntimeService runtimeService =

\* processEngine.getRuntimeService(); ProcessInstance processInstance = runtimeService

\* .startProcessInstanceByKey("reimbursementRequest",variables); //Verify that we started anew process instance

\* LOG.info("Process instance:"+processInstance.getId()); LOG.info("Process instances:"

\* +runtimeService.createProcessInstanceQuery().list().toString()); LOG.info

\* ("Number of process instances:"+runtimeService.createProcessInstanceQuery ().count());

\*/

// ActiveMQ message receiver

// ActivemqSender sender = ActivemqSender.getInstance("SAMPLEQUEUE");

String businessKey = repositoryService.createProcessDefinitionQuery().list().get(0).getKey();

String processDefinitionId = repositoryService.createProcessDefinitionQuery().list().get(0).getId();

String activemqChannelName = businessKey + "/" + processDefinitionId;

// Save the queueName.

ActivemqVariables.channelName = activemqChannelName;

LOG.info("ActiveMQ initializing with channel name:" + ActivemqVariables.channelName);

// ActivemqSender sender = new ActivemqSender(activemqQueueName);

// sender.sendMessage("echo");//For testing

// ActivemqReceiver receiver = new ActivemqReceiver("SAMPLEQUEUE");

// ActivemqReceiver receiver =

// ActivemqReceiver.getInstance("SAMPLEQUEUE");

// receiver.receiveMessage();

// Generate basic data base.

// // EntityManager entityManager =

// Persistence.createEntityManagerFactory("activiti\_test").createEntityManager();

// entityManager.getTransaction().begin();

// Company company = new Company();

// company.setBusinessKey(businessKey);

// company.setDate(new Date());

// company.setDomain("example.com");

// company.setEmail("sample@example.com");

// company.setName("EXAMPLE.COM");

// entityManager.persist(company);

// entityManager.getTransaction().commit();

}

// @see:

// http://stackoverflow.com/questions/26425067/resolvedspring-boot-access-to-entitymanager

@Bean

public PersistenceAnnotationBeanPostProcessor persistenceBeanPostProcessor() {

return new PersistenceAnnotationBeanPostProcessor();

}

// @see:

// http://stackoverflow.com/questions/23446928/spring-boot-uploading-files-path

@Bean

public CommonsMultipartResolver multipartResolver() {

return new CommonsMultipartResolver();

}

}

package info.smartkit.eip.config;

import java.util.Properties;

import org.apache.logging.log4j.LogManager;

import org.apache.logging.log4j.Logger;

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

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

import org.springframework.context.annotation.PropertySource;

import org.springframework.mail.javamail.JavaMailSender;

import org.springframework.mail.javamail.JavaMailSenderImpl;

@Configuration

@PropertySource("classpath:mail-${spring.profiles.active}.properties")

public class MailConfiguration {

@Value("${mail.protocol}")

private String protocol;

@Value("${mail.host}")

private String host;

@Value("${mail.port}")

private int port;

@Value("${mail.smtp.auth}")

private boolean auth;

@Value("${mail.smtp.starttls.enable}")

private boolean starttls;

@Value("${mail.from}")

private String from;

@Value("${mail.username}")

private String username;

@Value("${mail.password}")

private String password;

private static Logger LOG = LogManager.getLogger(MailConfiguration.class);

@Bean

public JavaMailSender javaMailSender() {

JavaMailSenderImpl mailSender = new JavaMailSenderImpl();

Properties mailProperties = new Properties();

mailProperties.put("mail.smtp.auth", auth);

mailProperties.put("mail.smtp.starttls.enable", starttls);

mailSender.setJavaMailProperties(mailProperties);

mailSender.setHost(host);

mailSender.setPort(port);

mailSender.setProtocol(protocol);

mailSender.setUsername(username);

mailSender.setPassword(password);

//

LOG.info("mailProperties:"+mailProperties.toString()+",mailSender:"+host+port+protocol+username+password);

//

return mailSender;

}

}/\*

\* Copyright 2014,2015 the original author or authors.

\*

\* Licensed under the Apache License, Version 2.0 (the "License");

\* you may not use this file except in compliance with the License.

\* You may obtain a copy of the License at

\*

\* http://www.apache.org/licenses/LICENSE-2.0

\*

\* Unless required by applicable law or agreed to in writing, software

\* distributed under the License is distributed on an "AS IS" BASIS,

\* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.

\* See the License for the specific language governing permissions and

\* limitations under the License.

\*/

package info.smartkit.eip.config;

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

import org.springframework.beans.factory.annotation.Qualifier;

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

import org.springframework.context.annotation.Primary;

import org.springframework.security.authentication.AuthenticationManager;

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

import org.springframework.security.oauth2.config.annotation.configurers.ClientDetailsServiceConfigurer;

import org.springframework.security.oauth2.config.annotation.web.configuration.AuthorizationServerConfigurerAdapter;

import org.springframework.security.oauth2.config.annotation.web.configuration.EnableAuthorizationServer;

import org.springframework.security.oauth2.config.annotation.web.configuration.EnableResourceServer;

import org.springframework.security.oauth2.config.annotation.web.configuration.ResourceServerConfigurerAdapter;

import org.springframework.security.oauth2.config.annotation.web.configurers.AuthorizationServerEndpointsConfigurer;

import org.springframework.security.oauth2.config.annotation.web.configurers.ResourceServerSecurityConfigurer;

import org.springframework.security.oauth2.provider.token.DefaultTokenServices;

import org.springframework.security.oauth2.provider.token.TokenStore;

import org.springframework.security.oauth2.provider.token.store.InMemoryTokenStore;

//@see: http://devcenter.kinvey.com/rest/guides/auth-link-for-ldap

@Configuration

public class OAuth2ServerConfiguration

{

private static final String RESOURCE\_ID = "restservice";

@Configuration

@EnableResourceServer

protected static class ResourceServerConfiguration extends ResourceServerConfigurerAdapter

{

@Override

public void configure(ResourceServerSecurityConfigurer resources)

{

// @formatter:off

resources

.resourceId(RESOURCE\_ID);

// @formatter:on

}

@Override

public void configure(HttpSecurity http) throws Exception

{

// @formatter:off

http

.authorizeRequests()

.antMatchers("/xyz/\*\*").authenticated();//TODO:Every controllers goes like clockwork.

// @formatter:on

}

}

@Configuration

@EnableAuthorizationServer

protected static class AuthorizationServerConfiguration extends AuthorizationServerConfigurerAdapter

{

private TokenStore tokenStore = new InMemoryTokenStore();

@Autowired

@Qualifier("authenticationManagerBean")

private AuthenticationManager authenticationManager;

@Override

public void configure(AuthorizationServerEndpointsConfigurer endpoints) throws Exception

{

// @formatter:off

endpoints

.tokenStore(this.tokenStore)

.authenticationManager(this.authenticationManager);

// @formatter:on

}

@Override

public void configure(ClientDetailsServiceConfigurer clients) throws Exception

{

// @formatter:off

clients

.inMemory()

.withClient("clientapp")

.authorizedGrantTypes("password","refresh\_token")

.authorities("USER")

// .scopes("read", "write")

.resourceIds(RESOURCE\_ID)

.secret("1NDgzZGY1OWViOWRmNjI5ZT");

// @formatter:on

}

@Bean

@Primary

public DefaultTokenServices tokenServices()

{

DefaultTokenServices tokenServices = new DefaultTokenServices();

tokenServices.setSupportRefreshToken(true);

tokenServices.setTokenStore(this.tokenStore);

return tokenServices;

}

}

}

package info.smartkit.eip.config;

import java.io.IOException;

import org.apache.commons.lang3.StringUtils;

import org.apache.logging.log4j.LogManager;

import org.apache.logging.log4j.Logger;

import org.springframework.context.ApplicationContextInitializer;

import org.springframework.context.ConfigurableApplicationContext;

import org.springframework.core.env.ConfigurableEnvironment;

import org.springframework.core.io.support.ResourcePropertySource;

import info.smartkit.eip.settings.ServerSetting;

import info.smartkit.eip.settings.WorkflowSetting;

/\*\*

\* Register this with the DispatcherServlet in a ServletInitializer class like:

\* dispatcherServlet.setContextInitializers(new PropertiesInitializer());

\*/

public class PropertiesInitializer implements ApplicationContextInitializer<ConfigurableApplicationContext>

{

private static final Logger LOG = LogManager.getLogger(PropertiesInitializer.class);

/\*\*

\* Runs as appInitializer so properties are wired before spring beans

\*/

@Override

public void initialize(ConfigurableApplicationContext applicationContext)

{

ConfigurableEnvironment env = applicationContext.getEnvironment();

String[] activeProfiles = getActiveProfiles(env);

for (String profileName : activeProfiles) {

LOG.info("Loading properties for Spring Active Profile: {}", profileName);

try {

ResourcePropertySource propertySource =

new ResourcePropertySource(profileName + "EnvProperties", "classpath:application-" + profileName

+ ".properties");

env.getPropertySources().addLast(propertySource);

LOG.debug("propertySource:" + propertySource.toString());

// Work-flow setting initialization here.

// TODO: @see https://github.com/EsotericSoftware/yamlbeans to replace this staff.

WorkflowSetting.getInstance().setBpmn((String) propertySource.getProperty("workflow.bpmn"));

WorkflowSetting.getInstance().setName((String) propertySource.getProperty("workflow.name"));

WorkflowSetting.getInstance().setImage((String) propertySource.getProperty("workflow.image"));

WorkflowSetting.getInstance().setRule((String) propertySource.getProperty("workflow.rule"));

//

ServerSetting.getInstance().setContextPath((String) propertySource.getProperty("server.contextPath"));

ServerSetting.getInstance().setPort((String) propertySource.getProperty("server.port"));

// ServerSetting.getInstance().setIp((String) propertySource.getProperty("mserver.ip"));

} catch (IOException e) {

LOG.error("ERROR during environment properties setup - TRYING TO LOAD: " + profileName, e);

// Okay to silently fail here, as we might have profiles that do

// not have properties files (like dev1, dev2, etc)

}

}

}

/\*\*

\* Returns either the ActiveProfiles, or if empty, then the DefaultProfiles from Spring

\*/

protected String[] getActiveProfiles(ConfigurableEnvironment env)

{

String[] activeProfiles = env.getActiveProfiles();

if (activeProfiles.length > 0) {

LOG.info("Using registered Spring Active Profiles: {}", StringUtils.join(activeProfiles, ", "));

return activeProfiles;

}

String[] defaultProfiles = env.getDefaultProfiles();

LOG.info("No Active Profiles found, using Spring Default Profiles: {}", StringUtils.join(defaultProfiles, ", "));

return defaultProfiles;

}

}

/\*

\* Copyright 2015 the original author or authors.

\*

\* Licensed under the Apache License, Version 2.0 (the "License");

\* you may not use this file except in compliance with the License.

\* You may obtain a copy of the License at

\*

\* http://www.apache.org/licenses/LICENSE-2.0

\*

\* Unless required by applicable law or agreed to in writing, software

\* distributed under the License is distributed on an "AS IS" BASIS,

\* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.

\* See the License for the specific language governing permissions and

\* limitations under the License.

\*

\* All rights reserved.

\*/

package info.smartkit.eip.config;

import org.apache.logging.log4j.LogManager;

import org.apache.logging.log4j.Logger;

import org.springframework.context.ApplicationListener;

import org.springframework.context.event.ContextRefreshedEvent;

import org.springframework.stereotype.Component;

/\*\*

\* @see: http://stackoverflow.com/questions/2401489/execute-method-on-startup-in-spring

\* @see: http://blog.comsysto.com/2014/10/17/spring-boot-my-favorite-timesaving-conventionenabling-autoconfigcreating-

\* beanmaking-classpathshaking-microcontainer/

\* @author yangboz

\*/

@Component

public class StartupHouseKeeper implements ApplicationListener<ContextRefreshedEvent>

{

private static Logger LOG = LogManager.getLogger(StartupHouseKeeper.class);

@Override

public void onApplicationEvent(final ContextRefreshedEvent event)

{

LOG.debug("onApplicationEvent!");

//

}

}

package info.smartkit.eip.config;

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

import org.springframework.context.annotation.Description;

import org.springframework.context.support.ResourceBundleMessageSource;

import org.springframework.web.servlet.ViewResolver;

import org.springframework.web.servlet.config.annotation.EnableWebMvc;

import org.springframework.web.servlet.config.annotation.ResourceHandlerRegistry;

import org.springframework.web.servlet.config.annotation.WebMvcConfigurerAdapter;

import org.thymeleaf.spring4.SpringTemplateEngine;

import org.thymeleaf.spring4.view.ThymeleafViewResolver;

import org.thymeleaf.templateresolver.ServletContextTemplateResolver;

@EnableWebMvc

@Configuration

public class StaticResourceConfiguration extends WebMvcConfigurerAdapter {

// @see http://www.netthreads.co.uk/2014/07/21/web-spring-4-thymeleaf-hot-cache/

public static final String DEFAULT\_PREFIX = "/templates/";

public static final String DEFAULT\_SUFFIX = ".html";

public static final String DEFAULT\_MODE = "HTML5";

public static final String[] DEFAULT\_VIEW\_NAMES = {"\*"};

// Hot re-load, false=will reload changes.

public static final boolean DEFAULT\_CACHE = false;

@Bean

public ServletContextTemplateResolver templateResolver() {

ServletContextTemplateResolver resolver = new ServletContextTemplateResolver();

resolver.setPrefix(DEFAULT\_PREFIX);

resolver.setSuffix(DEFAULT\_SUFFIX);

resolver.setTemplateMode(DEFAULT\_MODE);

return resolver;

}

@Bean

public SpringTemplateEngine templateEngine() {

SpringTemplateEngine engine = new SpringTemplateEngine();

engine.setTemplateResolver(templateResolver());

// Need this for hot re-load

if (!DEFAULT\_CACHE) {

engine.setCacheManager(null);

}

return engine;

}

@Bean

public ViewResolver viewResolver() {

ThymeleafViewResolver viewResolver = new ThymeleafViewResolver();

viewResolver.setTemplateEngine(templateEngine());

viewResolver.setOrder(1);

viewResolver.setCache(DEFAULT\_CACHE);

viewResolver.setViewNames(DEFAULT\_VIEW\_NAMES);

return viewResolver;

}

@Bean

@Description("Spring message resolver")

public ResourceBundleMessageSource messageSource() {

ResourceBundleMessageSource messageSource = new ResourceBundleMessageSource();

messageSource.setBasename("messages");

return messageSource;

}

private static final String[] CLASSPATH\_RESOURCE\_LOCATIONS = {"classpath:/META-INF/resources/",

"classpath:/resources/", "classpath:/static/", "classpath:/public/"};

/\*\*

\* Add our static resources folder mapping.

\*/

@Override

public void addResourceHandlers(ResourceHandlerRegistry registry) {

//

registry.addResourceHandler("/\*\*").addResourceLocations(CLASSPATH\_RESOURCE\_LOCATIONS);

//

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

// Activiti repository resources(diagram picture,process BPM files).

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

// Jasper report

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

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

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

//

super.addResourceHandlers(registry);

}

}

package info.smartkit.eip.config;

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

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

import com.mangofactory.swagger.configuration.SpringSwaggerConfig;

import com.mangofactory.swagger.plugin.EnableSwagger;

import com.mangofactory.swagger.plugin.SwaggerSpringMvcPlugin;

import com.wordnik.swagger.model.ApiInfo;

@Configuration

@EnableSwagger

public class SwaggerConfig {

private SpringSwaggerConfig springSwaggerConfig;

@Autowired

public void setSpringSwaggerConfig(SpringSwaggerConfig springSwaggerConfig) {

this.springSwaggerConfig = springSwaggerConfig;

}

@Bean //Don't forget the @Bean annotation

public SwaggerSpringMvcPlugin customImplementation() {

return new SwaggerSpringMvcPlugin(this.springSwaggerConfig)

.apiInfo(apiInfo())

.includePatterns("/.\*");

}

private ApiInfo apiInfo() {

ApiInfo apiInfo = new ApiInfo(

"eip.smartkit.info API",

"API for eip.smartkit.info with a macho swagger",

"http://eip.smartkit.info/terms",

"mailto:youngwelle@gmail.com?Subject=Hello,",

"Rushucloud API Licence Type",

"http://eip.smartkit.info/licenses"

);

return apiInfo;

}

}/\*

\* Copyright 2014 the original author or authors.

\*

\* Licensed under the Apache License, Version 2.0 (the "License");

\* you may not use this file except in compliance with the License.

\* You may obtain a copy of the License at

\*

\* http://www.apache.org/licenses/LICENSE-2.0

\*

\* Unless required by applicable law or agreed to in writing, software

\* distributed under the License is distributed on an "AS IS" BASIS,

\* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.

\* See the License for the specific language governing permissions and

\* limitations under the License.

\*/

package info.smartkit.eip.config;

import javax.ws.rs.HttpMethod;

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

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

import org.springframework.core.annotation.Order;

import org.springframework.security.authentication.AuthenticationManager;

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

import org.springframework.security.config.annotation.web.builders.WebSecurity;

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

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

import org.springframework.security.config.annotation.web.servlet.configuration.EnableWebMvcSecurity;

import info.smartkit.eip.services.UserDetailsServiceImpl;

@Configuration

@EnableWebSecurity

@EnableWebMvcSecurity

@Order(202)

public class WebSecurityConfiguration extends WebSecurityConfigurerAdapter {

@Autowired

private UserDetailsServiceImpl userDetailsService;

@Override

protected void configure(AuthenticationManagerBuilder auth) throws Exception {

auth.userDetailsService(userDetailsService);

}

@Override

@Bean

public AuthenticationManager authenticationManagerBean() throws Exception {

return super.authenticationManagerBean();

}

@Override

/\*\*

\* @see org.springframework.security.config.annotation.web.configuration.WebSecurityConfigurerAdapter#configure(WebSecurity)

\*/

public void configure(WebSecurity web) throws Exception {

web.ignoring().antMatchers(HttpMethod.OPTIONS, "/\*\*");

}

}

package info.smartkit.eip.consts;

import org.apache.activemq.ActiveMQConnection;

public class JMSConstants {

final static public String JMS\_DESTINATION\_ACTIVEMQ = "message-destination-activemq";

final static public String LOG\_FILE\_NAME\_ACTIVEMQ = "file-data-activemq";

static public String URL\_BROKER\_ACTIVEMQ = ActiveMQConnection.DEFAULT\_BROKER\_URL;//tcp://182.92.232.131:61616

}package info.smartkit.eip.consts;

public class ThirdPartyConstants {

final static public String DZDP\_apiUrl = "http://api.dianping.com/v1/business/find\_businesses";

final static public String DZDP\_appKey = "33331153";

final static public String DZDP\_secret = "f1ab2e9ab04a4959a1bf8cb5740bb598";

//

final static public String BAIDU\_AK = "LusR3ViMWob3pqQTy1yfQfqA";

}

package info.smartkit.eip.controllers;

import javax.persistence.EntityManager;

import javax.persistence.PersistenceContext;

import javax.validation.Valid;

import info.smartkit.eip.models.Category;

import info.smartkit.eip.models.CategoryRepository;

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

import org.springframework.data.domain.Sort;

import org.springframework.http.HttpStatus;

import org.springframework.http.ResponseEntity;

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

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

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.bind.annotation.RestController;

import info.smartkit.eip.dto.JsonObject;

import info.smartkit.eip.models.CategoryDao;

import com.wordnik.swagger.annotations.ApiOperation;

@RestController

@RequestMapping("/category")

public class CategoryController {

// ==============

// PRIVATE FIELDS

// ==============

// Autowire an object of type CategoryDao

@Autowired

private CategoryDao \_categoryDao;

//

private CategoryRepository categoryRepository;

@Autowired

public CategoryController(CategoryRepository categoryRepository) {

this.categoryRepository = categoryRepository;

}

@PersistenceContext

EntityManager em;

// ==============

// PUBLIC METHODS

// ==============

private void queryParentByChild() {

}

private void queryChildrenByParent() {

}

// @RequestMapping(method = RequestMethod.POST,params={"root","parentId"})

@RequestMapping(method = RequestMethod.POST)

@ApiOperation(httpMethod = "POST", value = "Response a string describing if the reimbursement category is successfully created or not.")

public JsonObject create(@RequestBody @Valid Category category,

@RequestParam(value = "root", defaultValue = "false") Boolean root,

@RequestParam(value = "parentId", defaultValue = "1") long parentId) {

if(root)//Root category

{

}else{//Children category

//

Category parent = em.find(Category.class, parentId);

category.setParent(parent);

}

//@JsonBackReference http://keenformatics.blogspot.sg/2013/08/how-to-solve-json-infinite-recursion.html

// this.categoryRepository.save(category);

return new JsonObject(\_categoryDao.save(category));

}

@RequestMapping(method = RequestMethod.GET)

@ApiOperation(httpMethod = "GET", value = "Response a list describing all of category that is successfully get or not.")

public JsonObject list() {

return new JsonObject(this.categoryRepository.findAll(new Sort(

new Sort.Order(Sort.Direction.ASC, "date"))));

}

@RequestMapping(value = "/{id}", method = RequestMethod.GET)

@ApiOperation(httpMethod = "GET", value = "Response a string describing if the category id is successfully get or not.")

public Category get(@PathVariable("id") long id) {

return this.categoryRepository.findOne(id);

}

@RequestMapping(value = "/{id}", method = RequestMethod.PUT)

@ApiOperation(httpMethod = "PUT", value = "Response a string describing if the reimbursement category is successfully updated or not.")

public Category update(@PathVariable("id") long id,

@RequestBody @Valid Category category) {

return categoryRepository.save(category);

}

@RequestMapping(value = "/{id}", method = RequestMethod.DELETE)

@ApiOperation(httpMethod = "DELETE", value = "Response a string describing if the category is successfully delete or not.")

public ResponseEntity<Boolean> delete(@PathVariable("id") long id) {

this.categoryRepository.delete(id);

return new ResponseEntity<Boolean>(Boolean.TRUE, HttpStatus.OK);

}

}

/\*\*

\* @author yangboz

\* @see http://blog.netgloo.com/2014/10/27/using-mysql-in-spring-boot-via-spring-data-jpa-and-hibernate/

\*/

package info.smartkit.eip.controllers;

import javax.validation.Valid;

import info.smartkit.eip.models.Company;

import info.smartkit.eip.models.CompanyDao;

import info.smartkit.eip.models.CompanyRepository;

import info.smartkit.eip.models.User;

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

import org.springframework.http.HttpStatus;

import org.springframework.http.ResponseEntity;

import org.springframework.security.web.bind.annotation.AuthenticationPrincipal;

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

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

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

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

import org.springframework.web.bind.annotation.RestController;

import info.smartkit.eip.dto.JsonObject;

import com.wordnik.swagger.annotations.ApiOperation;

@RestController

@RequestMapping("/company")

public class CompanyController {

// @RequestMapping("company/index")

// @RequestMapping(method = RequestMethod.GET, value = "company/index")

// @ApiOperation(httpMethod = "GET", value = "Response a static index html view.")

// public String home() {

// return "index";

// }

// ==============

// PRIVATE FIELDS

// ==============

// Autowire an object of type CompanyDao

@Autowired

private CompanyDao \_companyDao;

//

private CompanyRepository companyRepository;

@Autowired

public CompanyController(CompanyRepository companyRepository) {

this.companyRepository = companyRepository;

}

// ==============

// PUBLIC METHODS

// ==============

@RequestMapping(method = RequestMethod.POST)

@ApiOperation(httpMethod = "POST", value = "Response a string describing if the reimbursement company is successfully created or not.")

public Company create(@RequestBody @Valid Company company) {

return this.companyRepository.save(company);

}

@RequestMapping(method = RequestMethod.GET)

@ApiOperation(httpMethod = "GET", value = "Response a list describing all of company that is successfully get or not.")

public JsonObject list(@AuthenticationPrincipal User user) {

// return new JsonObject(this.companyRepository.findAll(new Sort(new Sort.Order(Sort.Direction.ASC,"date"))));

return new JsonObject(this.\_companyDao.findAll());

}

@RequestMapping(value = "/{id}", method = RequestMethod.GET)

@ApiOperation(httpMethod = "GET", value = "Response a string describing if the company id is successfully get or not.")

public JsonObject get(@PathVariable("id") long id) {

return new JsonObject(this.companyRepository.findOne(id));

}

@RequestMapping(value = "/{id}", method = RequestMethod.PUT)

@ApiOperation(httpMethod = "PUT", value = "Response a string describing if the reimbursement company is successfully updated or not.")

public Company update(@PathVariable("id") long id,

@RequestBody @Valid Company company) {

return companyRepository.save(company);

}

@RequestMapping(value = "/{id}", method = RequestMethod.DELETE)

@ApiOperation(httpMethod = "DELETE", value = "Response a string describing if the company is successfully delete or not.")

public ResponseEntity<Boolean> delete(@PathVariable("id") long id) {

this.companyRepository.delete(id);

return new ResponseEntity<Boolean>(Boolean.TRUE, HttpStatus.OK);

}

}

package info.smartkit.eip.controllers;

import javax.validation.Valid;

import org.apache.logging.log4j.LogManager;

import org.apache.logging.log4j.Logger;

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

import org.springframework.http.HttpStatus;

import org.springframework.http.ResponseEntity;

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

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

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.bind.annotation.ResponseStatus;

import org.springframework.web.bind.annotation.RestController;

import info.smartkit.eip.dto.JsonObject;

import info.smartkit.eip.models.Expense;

import info.smartkit.eip.models.ExpenseDao;

import info.smartkit.eip.models.ExpenseRepository;

import com.wordnik.swagger.annotations.ApiOperation;

@RestController

@RequestMapping("/expenses")

public class ExpensesController {

//

private static Logger LOG = LogManager.getLogger(ExpensesController.class);

// ==============

// PRIVATE FIELDS

// ==============

// @PersistenceContext

// protected EntityManager entityManager;

// Autowire an object of type ExpenseDao

@Autowired

private ExpenseDao \_expenseDao;

//

private ExpenseRepository expenseRepository;

@Autowired

public ExpensesController(ExpenseRepository expenseRepository) {

this.expenseRepository = expenseRepository;

}

// ==============

// PUBLIC METHODS

// ==============

@RequestMapping(method = RequestMethod.POST)

@ApiOperation(httpMethod = "POST", value = "Response a string describing if the reimbursement expense is successfully created or not.")

@ResponseStatus(HttpStatus.CREATED)

public Expense create(@RequestBody @Valid Expense expense) {

// return this.expenseRepository.save(expense);

return this.\_expenseDao.save(expense);

}

@RequestMapping(method = RequestMethod.GET,params = {"owner"})

@ApiOperation(httpMethod = "GET", value = "Response a list describing all of expense that is successfully get or not.")

public JsonObject list(@RequestParam(value = "owner") String owner) {

// return new JsonObject(this.expenseRepository.findAll());

if(owner!=null)

{

//

Iterable<Expense> result = this.\_expenseDao.findExpensesByOwner(owner);

// LOG.debug("expensesByOwner()result:"+result.toString());

return new JsonObject(result);

}else{

return new JsonObject(this.\_expenseDao.findAll());

}

}

@RequestMapping(value = "/{id}", method = RequestMethod.GET)

@ApiOperation(httpMethod = "GET", value = "Response a string describing if the expense id is successfully get or not.")

public Expense get(@PathVariable("id") long id) {

// return this.expenseRepository.findOne(id);

return this.\_expenseDao.findById(id);

}

@RequestMapping(value = "/{id}", method = RequestMethod.PUT)

@ApiOperation(httpMethod = "PUT", value = "Response a string describing if the reimbursement expense is successfully updated or not.")

public Expense update(@PathVariable("id") long id,

@RequestBody @Valid Expense expense) {

// return this.expenseRepository.save(expense);

return this.\_expenseDao.save(expense);

}

@RequestMapping(value = "/{id}", method = RequestMethod.PATCH,params = {"owner","pid","status"})

@ApiOperation(httpMethod = "PATCH", value = "Response a string describing if the reimbursement expense is successfully patched or not.")

//@see: http://vaskoz.wordpress.com/2013/07/20/spring-requestmethod-patch/

public Expense patch(@PathVariable("id") long id

,@RequestBody Expense expense) {

// return this.expenseRepository.save(expense);

Expense patchExpense = this.\_expenseDao.findById(id);

//TODO:enumrate the non-null properties and patch value.

patchExpense.setPid(expense.getPid());

patchExpense.setStatus(expense.getStatus());

return this.\_expenseDao.save(patchExpense);

}

@RequestMapping(value = "/{id}", method = RequestMethod.DELETE)

@ApiOperation(httpMethod = "DELETE", value = "Response a string describing if the expense is successfully delete or not.")

public ResponseEntity<Boolean> delete(@PathVariable("id") Long id) {

this.expenseRepository.delete(id);

// this.\_expenseDao.delete(id);

return new ResponseEntity<Boolean>(Boolean.TRUE, HttpStatus.OK);

}

}

package info.smartkit.eip.controllers;

import java.io.BufferedOutputStream;

import java.io.File;

import java.io.FileOutputStream;

import java.io.IOException;

import java.text.SimpleDateFormat;

import java.util.ArrayList;

import java.util.Date;

import java.util.HashMap;

import java.util.List;

import java.util.Map;

import javax.servlet.ServletContext;

import info.smartkit.eip.models.Invoice;

import info.smartkit.eip.models.InvoiceDao;

import info.smartkit.eip.settings.FolderSetting;

import org.apache.commons.io.FilenameUtils;

import org.apache.logging.log4j.LogManager;

import org.apache.logging.log4j.Logger;

import org.im4java.core.ConvertCmd;

import org.im4java.core.IM4JavaException;

import org.im4java.core.IMOperation;

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

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.bind.annotation.ResponseBody;

import org.springframework.web.bind.annotation.RestController;

import org.springframework.web.multipart.MultipartFile;

import com.wordnik.swagger.annotations.ApiOperation;

@RestController

// @RequestMapping("/upload")

public class FileUploadController

{

// Enum for image size.

enum ImageSize

{

ori,

sml,

ico

}

//

private static Logger LOG = LogManager.getLogger(FileUploadController.class);

// Autowire an object of type InvoiceDao

@Autowired

private InvoiceDao \_invoiceDao;

//

@Autowired

ServletContext servletContext;

//

@RequestMapping(value = "/{id}", method = RequestMethod.GET)

@ApiOperation(httpMethod = "GET", value = "Response a string describing invoices pictures is successfully get or not.")

public Invoice provideUploadInfo(@PathVariable("id") long id)

{

return \_invoiceDao.findById(id);

}

//

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

@ApiOperation(httpMethod = "POST", value = "Response a string describing invoice' picture is successfully uploaded or not.")

public @ResponseBody Invoice handleSingleFileUpload(@RequestParam("name") String name,

@RequestParam("owner") String owner, @RequestParam("file") MultipartFile file)

{

Invoice invoiceResp = null;

if (!file.isEmpty()) {

// ImageMagick convert options; @see:

// http://paxcel.net/blog/java-thumbnail-generator-imagescalar-vs-imagemagic/

Map<String, String> \_imageMagickOutput = this.fileOperation(file);

// Save to database.

try {

invoiceResp = new Invoice(file.getOriginalFilename(), owner, \_imageMagickOutput);

\_invoiceDao.save(invoiceResp);

LOG.info("\_invoiceDao save success.");

} catch (Exception ex) {

LOG.error(ex.toString());

}

return invoiceResp;

} else {

LOG.error("You failed to upload " + name + " because the file was empty.");

}

return invoiceResp;

}

// @see:

// http://www.concretepage.com/spring-4/spring-4-mvc-single-multiple-file-upload-example-with-tomcat

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

@ApiOperation(httpMethod = "POST", value = "Response a string describing invoice' pictures is successfully uploaded or not.")

public @ResponseBody List<Invoice> handleMultiFileUpload(@RequestParam("name") String name,

@RequestParam("owner") String owner, @RequestParam("files") MultipartFile[] files)

{

@SuppressWarnings("serial")

ArrayList<Invoice> invoiceResps = new ArrayList<Invoice>()

{

};

if (files != null && files.length > 0) {

for (int i = 0; i < files.length; i++) {

if (!files[i].isEmpty()) {

// ImageMagick convert options; @see:

// http://paxcel.net/blog/java-thumbnail-generator-imagescalar-vs-imagemagic/

Map<String, String> \_imageMagickOutput = this.fileOperation(files[i]);

// Save to database.

try {

Invoice invoiceResp = new Invoice(files[i].getOriginalFilename(), owner, \_imageMagickOutput);

\_invoiceDao.save(invoiceResp);

LOG.info("\_invoiceDao save success.");

invoiceResps.add(invoiceResp);

} catch (Exception ex) {

LOG.error(ex.toString());

}

} else {

LOG.error("You failed to upload " + files.toString() + " because the file was empty.");

}

}

}

return invoiceResps;

}

//

private String thumbnailImage(int width, int height, String source) throws IOException, InterruptedException,

IM4JavaException

{

//

String small4dbBase =

FilenameUtils.getBaseName(source) + "\_" + String.valueOf(width) + "x" + String.valueOf(height) + "."

+ FilenameUtils.getExtension(source);

String small4db = folderSetting.getUploads() + small4dbBase;

String small = getClassPath() + small4db;

// @see:

// http://paxcel.net/blog/java-thumbnail-generator-imagescalar-vs-imagemagic/

ConvertCmd cmd = new ConvertCmd();

// cmd.setSearchPath("");

File thumbnailFile = new File(small);

if (!thumbnailFile.exists()) {

IMOperation op = new IMOperation();

op.addImage(source);

op.thumbnail(width);

op.addImage(small);

cmd.run(op);

LOG.info("ImageMagick success result:" + small);

}

return small4dbBase;

}

// private String getWorkingDir(){

// String workingDir = System.getProperty("user.dir");

// return workingDir;

// }

public String getClassPath()

{

String classPath = this.getClass().getResource("/").getPath();

return classPath;

}

@Autowired

private FolderSetting folderSetting;

private Map<String, String> fileOperation(MultipartFile file)

{

Map<String, String> \_imageMagickOutput = new HashMap<String, String>();

String dbFileName = null;

String fullFileName = null;

try {

byte[] bytes = file.getBytes();

String fileExt = FilenameUtils.getExtension(file.getOriginalFilename());

String fileNameAppendix =

new SimpleDateFormat("yyyy-MM-dd-HH-mm-ss-SSS").format(new Date()) + "." + fileExt;

//

dbFileName = folderSetting.getUploads() + fileNameAppendix;

fullFileName = getClassPath() + dbFileName;

//

BufferedOutputStream stream = new BufferedOutputStream(new FileOutputStream(new File(fullFileName)));

stream.write(bytes);

stream.close();

LOG.info("Upload file success." + fullFileName);

// ImageMagick convert options; @see:

// http://paxcel.net/blog/java-thumbnail-generator-imagescalar-vs-imagemagic/

\_imageMagickOutput.put(ImageSize.ori.toString(), fileNameAppendix);

\_imageMagickOutput.put(ImageSize.sml.toString(), thumbnailImage(150, 150, fullFileName));

\_imageMagickOutput.put(ImageSize.ico.toString(), thumbnailImage(32, 32, fullFileName));

return \_imageMagickOutput;

} catch (Exception e) {

LOG.error("You failed to convert " + fullFileName + " => " + e.toString());

}

return \_imageMagickOutput;

}

}

package info.smartkit.eip.controllers;

import java.util.List;

import javax.persistence.EntityManager;

import javax.persistence.PersistenceContext;

import javax.validation.Valid;

import org.apache.logging.log4j.LogManager;

import org.apache.logging.log4j.Logger;

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

import org.springframework.http.HttpStatus;

import org.springframework.http.ResponseEntity;

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

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

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.bind.annotation.RestController;

import info.smartkit.eip.dto.JsonObject;

import info.smartkit.eip.models.Item;

import info.smartkit.eip.models.ItemDao;

import info.smartkit.eip.models.ItemRepository;

import com.wordnik.swagger.annotations.ApiOperation;

@RestController

@RequestMapping("/items")

// @see: http://java.dzone.com/articles/spring-rest-controller

public class ItemsController {

private Logger LOG = LogManager.getLogger(ItemsController.class);

// ==============

// PRIVATE FIELDS

// ==============

// Autowire an object of type CompanyDao

@Autowired

private ItemDao \_itemDao;

//

private ItemRepository itemRepository;

@Autowired

public ItemsController(ItemRepository itemRepository) {

this.itemRepository = itemRepository;

}

@PersistenceContext

EntityManager entityManager;

// ==============

// PUBLIC METHODS

// ==============

@RequestMapping(method = RequestMethod.POST)

@ApiOperation(httpMethod = "POST", value = "Response a string describing if the reimbursement item is successfully created or not.")

public Item create(@RequestBody @Valid Item item) {

return this.itemRepository.save(item);

}

// @RequestMapping(method = RequestMethod.GET)

// @ApiOperation(httpMethod = "GET", value =

// "Response a list describing all of item that is successfully get or not.")

// public JsonObject list() {

// return new JsonObject(this.itemRepository.findAll(new Sort(new

// Sort.Order(Sort.Direction.ASC,"date"))));

// }

@RequestMapping(method = RequestMethod.GET, params = { "owner", "used" })

@ApiOperation(httpMethod = "GET", value = "Response a list describing all of item that is successfully get or not.")

public JsonObject list(@RequestParam(value = "owner") String owner,

@RequestParam(value = "used", defaultValue = "false") String used) {

// return new JsonObject(this.expenseRepository.findAll());

if (owner != null) {

//

// Iterable<Item> result = this.\_itemDao.findItemsByOwner(owner);

//@see: http://docs.oracle.com/cd/E16439\_01/doc.1013/e13981/usclient005.htm

String queryStr = "SELECT OBJECT(item) FROM "

+ Item.class.getName() + " item WHERE item.owner='" + owner

+ "' AND item.used='"+used+"'";

@SuppressWarnings("unchecked")

List<Object[]> results = this.entityManager.createQuery(queryStr)

.getResultList();

// LOG.debug("itemsByOwner()result:"+result.toString());

return new JsonObject(results);

} else {

return new JsonObject(this.\_itemDao.findAll());

}

}

@RequestMapping(value = "/{id}", method = RequestMethod.GET)

@ApiOperation(httpMethod = "GET", value = "Response a string describing if the item id is successfully get or not.")

public Item get(@PathVariable("id") long id) {

return this.itemRepository.findOne(id);

}

@RequestMapping(value = "/{id}", method = RequestMethod.PUT)

@ApiOperation(httpMethod = "PUT", value = "Response a string describing if the reimbursement item is successfully updated or not.")

public Item update(@PathVariable("id") long id,

@RequestBody @Valid Item item) {

return itemRepository.save(item);

}

@RequestMapping(value = "/{id}", method = RequestMethod.DELETE)

@ApiOperation(httpMethod = "DELETE", value = "Response a string describing if the item is successfully delete or not.")

public ResponseEntity<Boolean> delete(@PathVariable("id") long id) {

this.itemRepository.delete(id);

return new ResponseEntity<Boolean>(Boolean.TRUE, HttpStatus.OK);

}

}

package info.smartkit.eip.controllers;

//@see: http://www.javaworld.com/article/2076073/java-web-development/ldap-and-jndi--together-forever.html

//@see: https://docs.oracle.com/javase/tutorial/jndi/ldap/operations.html

//@see: http://docs.spring.io/spring-ldap/docs/current/reference/

//@see: LDAP Manager Java code example: http://www.javafaq.nu/java-example-code-409.html

import static org.springframework.ldap.query.LdapQueryBuilder.query;

//@see: http://docs.spring.io/spring-ldap/docs/current/reference/#introduction

import java.util.List;

import java.util.Properties;

import javax.naming.Context;

import javax.naming.NamingException;

import javax.naming.directory.BasicAttribute;

import javax.naming.directory.BasicAttributes;

import javax.naming.directory.DirContext;

import javax.naming.directory.InitialDirContext;

import javax.naming.directory.ModificationItem;

import javax.validation.Valid;

import info.smartkit.eip.settings.LDAPSetting;

import org.apache.logging.log4j.LogManager;

import org.apache.logging.log4j.Logger;

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

import org.springframework.ldap.core.AuthenticatedLdapEntryContextMapper;

import org.springframework.ldap.core.DirContextOperations;

import org.springframework.ldap.core.LdapEntryIdentification;

import org.springframework.ldap.core.LdapTemplate;

import org.springframework.ldap.core.support.LdapContextSource;

import org.springframework.security.authentication.encoding.LdapShaPasswordEncoder;

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

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.bind.annotation.RestController;

import info.smartkit.eip.dto.JsonObject;

import info.smartkit.eip.ldap.plain.domain.Person;

import info.smartkit.eip.models.PersonAttributesMapper;

import info.smartkit.eip.thirdParty.WeixinUser;

import com.wordnik.swagger.annotations.ApiOperation;

@RestController

public class LDAPController

{

//

private static Logger LOG = LogManager.getLogger(LDAPController.class);

//

// private OdmPersonRepo \_personRepo;

//

// @Autowired

// public LDAPController(OdmPersonRepo personRepo) {

// this.\_personRepo = personRepo;

// }

public String hashAndEncodePassword(String password)

{

LdapShaPasswordEncoder encoder = new LdapShaPasswordEncoder();

// String encodedResult = PasswordUtil.createStoragePassword(

// password.getBytes(), LdapSecurityConstants.HASH\_METHOD\_SHA);

return encoder.encodePassword(password, null);// {SSHA}

}

//

@RequestMapping(method = RequestMethod.GET, value = "ldap/auth")

@ApiOperation(httpMethod = "GET", value = "LDAP client for authenticate user.")

public JsonObject authenticate(

@RequestParam(value = "uid", required = true, defaultValue = "employee0") String uid,

@RequestParam(value = "password", required = true, defaultValue = "passwordpassword") String password)

{

//

LdapTemplate ldapTemplate = getLdapTemplate();

//

AuthenticatedLdapEntryContextMapper<DirContextOperations> mapper =

new AuthenticatedLdapEntryContextMapper<DirContextOperations>()

{

@Override

public DirContextOperations mapWithContext(DirContext ctx,

LdapEntryIdentification ldapEntryIdentification)

{

try {

return (DirContextOperations) ctx.lookup(ldapEntryIdentification.getRelativeName());

} catch (NamingException e) {

throw new RuntimeException("Failed to lookup " + ldapEntryIdentification.getRelativeName(), e);

}

}

};

ldapTemplate.authenticate(query().where("uid").is(uid), password, mapper);

//

// CollectingAuthenticationErrorCallback errorCallback = new CollectingAuthenticationErrorCallback();

// ldapTemplate.authenticate(query().where("uid").is(uid), password,errorCallback);

// boolean result = ldapTemplate.authenticate("ou=employees,ou=www1.rushucloud.com,dc=www", "uid="+uid,

// hashAndEncodePassword(password), errorCallback);

// Person result = \_personRepo.findByUid(uid);

// LOG.debug("ldapTemplate authenticate info:"+uid+hashAndEncodePassword(password));

return new JsonObject(true);

}

//

// @RequestMapping(method = RequestMethod.GET, value = "ldap/person/{uid}")

// @ApiOperation(httpMethod = "GET", value =

// "Response a person object describing if the person uid is successfully get or not.")

// public JsonObject get(@PathVariable("uid") String uid) {

// Person result = \_personRepo.findByUid(uid);

// return new JsonObject(result);

// }

//

@RequestMapping(method = RequestMethod.GET, value = "ldap/search")

@ApiOperation(httpMethod = "GET", value = "LDAP search client for testing purpose.")

public JsonObject search(

@RequestParam(value = "baseOn", required = true, defaultValue = "ou=employees,ou=www1.rushucloud.com,dc=www") String baseOn,

@RequestParam(value = "filter", required = true, defaultValue = "(objectClass=person)") String filter)

{

LdapTemplate ldapTemplate = getLdapTemplate();

// LOG.info("ldapConfig:"+this.ldapSetting.toString());

//

List<Person> persons = ldapTemplate.search(query().base(baseOn).filter(filter),

// .where("objectclass").is("person"),

new PersonAttributesMapper());

// LOG.info("ldap searched persons:" + persons.toString());

// for (Person person : persons) {

// LOG.debug("ldapSearch person:" + person.toString());

// }

return new JsonObject(persons);

}

// Notice:

// http://stackoverflow.com/questions/4935612/how-to-create-a-partition-in-the-root-in-apacheds-server-programmatically-using

// Notice:

// http://stackoverflow.com/questions/8512393/custom-partition-in-apacheds

// So currently using o=xxx.com,parent as dc=www at child level for DIT

// structure.

// In the future, using partition as dc=xxx.com at root level for DIT

// structure.

@RequestMapping(method = RequestMethod.POST, value = "ldap/dc")

@ApiOperation(httpMethod = "POST", value = "LDAP orgnization(as domain component) adding function.")

public Boolean addOrgnizationUnit(

@RequestParam(value = "partitionId(o/ou)", required = true, defaultValue = "xxx.com") String partitionId,

@RequestParam(value = "partitionDn(domain)", required = true, defaultValue = "www") String partitionDn)

throws Exception

{

LdapTemplate ldapTemplate = getLdapTemplate();

//

// Set the Patron attributes

BasicAttributes attributes = new BasicAttributes();

attributes.put("ou", partitionId);

// Add the multiply-valued attribute

BasicAttribute objectClassAttribute = new BasicAttribute("objectclass");

objectClassAttribute.add("top");

objectClassAttribute.add("organizationalUnit");

// objectClassAttribute.add("domain");

attributes.put(objectClassAttribute);

ldapTemplate.bind("ou=" + partitionId + ",dc=" + partitionDn, null, attributes);

//

/\*

\* BasicAttributes attrs = new BasicAttributes(true); BasicAttribute objClass = new

\* BasicAttribute("objectclass"); objClass.add("top"); //

\* objClass.add("domain");//domain,organizational,organizationalUnit objClass.add("organizationalUnit");

\* attrs.put(objClass); // Create the context // DirContext dirCtx = this.getInitialContext(); DirContext dirCtx

\* = ldapTemplate.getContextSource() .getReadWriteContext(); // dirCtx.bind(partitionDn, null, attrs); // Create

\* the context // Context result = dirCtx.createSubcontext("ou=" + partitionDn, attrs); // Check that it was

\* created by listing its parent NamingEnumeration list = dirCtx.list(""); // Go through each item in list while

\* (list.hasMore()) { NameClassPair nc = (NameClassPair) list.next(); // System.out.println(nc);

\* LOG.info("NameClassPair:" + nc.toString()); } result.close(); dirCtx.close();

\*/

/\*

\* DirectoryService directoryService = this.getDirectoryService(); // Create a new partition JdbmPartition

\* helloPartition = new JdbmPartition(); helloPartition.setId("hello"); helloPartition.setSuffix("ou=hello");

\* helloPartition.init(directoryService); directoryService.addPartition(helloPartition);

\* directoryService.startup(); // ClonedServerEntry entry = directoryService.getAdminSession().lookup(new

\* LdapDN("ou=hello")); // Assert.assertNotNull(entry); directoryService.shutdown();

\*/

//

return true;

}

//

@RequestMapping(method = RequestMethod.PUT, value = "ldap/add")

@ApiOperation(httpMethod = "PUT", value = "LDAP client for adding attribute/entry purpose.")

public void add(

@RequestParam(value = "partition", required = true, defaultValue = "ou=www1.rushucloud.com,dc=www") String partition,

@RequestParam(value = "ou", required = true, defaultValue = "patrons") String ou,

@RequestParam(value = "uid", required = true, defaultValue = "patron9999") String uid,

@RequestParam(value = "sn", required = true, defaultValue = "Patron9999") String sn,

@RequestParam(value = "cn", required = true, defaultValue = "New Patron9999") String cn

// @RequestParam(value = "basicAttributes", required = false, defaultValue =

// "") BasicAttributes basicAttributes

)

{

// Set the Patron attributes

BasicAttributes attributes = new BasicAttributes();

attributes.put("sn", sn);

attributes.put("cn", cn);

// Add the multiply-valued attribute

BasicAttribute objectClassAttribute = new BasicAttribute("objectclass");

objectClassAttribute.add("top");

objectClassAttribute.add("person");

objectClassAttribute.add("organizationalperson");

objectClassAttribute.add("inetorgperson");

attributes.put(objectClassAttribute);

LdapTemplate ldapTemplate = getLdapTemplate();

ldapTemplate.bind("uid=" + uid + ",ou=" + ou + "," + partition, null, attributes);

}

//

@RequestMapping(method = RequestMethod.POST, value = "ldap/import")

@ApiOperation(httpMethod = "POST", value = "LDAP gather user and groups for importing purpose.")

public void gather(@RequestBody @Valid List<WeixinUser> users,

@RequestParam(value = "partitionId(o/ou)", required = true, defaultValue = "xxx.com") String partitionId,

@RequestParam(value = "partitionDn(domain)", required = true, defaultValue = "www") String partitionDn)

{

LdapTemplate ldapTemplate = getLdapTemplate();

for (WeixinUser user : users) {

// BasicAttribute attribute = new BasicAttribute("mobile",

// user.getMobile());

// //

// ModificationItem item = new ModificationItem(

// DirContext.ADD\_ATTRIBUTE, attribute);

// ldapTemplate.modifyAttributes("uid=" + user.getWeixinId() +

// ",ou="

// + user.getDepartment() + "," + partition,

// new ModificationItem[] { item });

// Set the Person attributes

BasicAttributes attributes = new BasicAttributes();

attributes.put("sn", user.getName());

attributes.put("cn", user.getWeixinId());

attributes.put("mobile", user.getMobile());

// attributes.put("mail", user.getEmail());

// Add the multiply-valued attribute

BasicAttribute objectClassAttribute = new BasicAttribute("objectclass");

objectClassAttribute.add("top");

objectClassAttribute.add("person");

objectClassAttribute.add("organizationalperson");

objectClassAttribute.add("inetorgperson");

// objectClassAttribute.add("mail");

attributes.put(objectClassAttribute);

//

ldapTemplate.bind("uid=" + user.getWeixinId() + ",ou=" + partitionId + ",dc=" + partitionDn, null,

attributes);

}

}

//

@RequestMapping(method = RequestMethod.PUT, value = "ldap/update")

@ApiOperation(httpMethod = "PUT", value = "LDAP search client for updating purpose.")

public void update(

@RequestParam(value = "partition", required = true, defaultValue = "dc=inflinx,dc=com") String partition,

@RequestParam(value = "ou", required = true, defaultValue = "partons") String ou,

@RequestParam(value = "uid", required = true, defaultValue = "parton9999") String uid

// @RequestParam(value = "basicAttributes", required = false, defaultValue =

// "") BasicAttributes basicAttributes

)

{

LdapTemplate ldapTemplate = getLdapTemplate();

BasicAttribute attribute = new BasicAttribute("telephoneNumber", "801 100 1000");

ModificationItem item = new ModificationItem(DirContext.ADD\_ATTRIBUTE, attribute);

ldapTemplate.modifyAttributes("uid=" + uid + ",ou=" + ou + "," + partition, new ModificationItem[] {item});

}

//

@RequestMapping(method = RequestMethod.DELETE, value = "ldap/delete")

@ApiOperation(httpMethod = "DELETE", value = "LDAP search client for deleting purpose.")

public void delete(

@RequestParam(value = "partition", required = true, defaultValue = "dc=inflinx,dc=com") String partition,

@RequestParam(value = "ou", required = true, defaultValue = "partons") String ou,

@RequestParam(value = "uid", required = true, defaultValue = "parton9999") String uid)

{

LdapTemplate ldapTemplate = getLdapTemplate();

ldapTemplate.unbind("uid=" + uid + ",ou=" + ou + "," + partition);

}

// @see

// http://hoserdude.com/2014/06/19/spring-boot-configurationproperties-and-profile-management-using-yaml/

@Autowired

private LDAPSetting ldapSetting;

//

private LdapTemplate getLdapTemplate()

{

LdapContextSource contextSource = new LdapContextSource();

contextSource.setUrl(ldapSetting.getUrl());

contextSource.setUserDn(ldapSetting.getUserOn());

contextSource.setPassword(ldapSetting.getPassword());

try {

contextSource.afterPropertiesSet();

} catch (Exception e) {

e.printStackTrace();

}

LdapTemplate ldapTemplate = new LdapTemplate();

ldapTemplate.setContextSource(contextSource);

return ldapTemplate;

}

// @see: http://www.javafaq.nu/java-example-code-409.html

@SuppressWarnings("unused")

private DirContext getInitialContext() throws NamingException

{

Properties props = new Properties();

props.put(Context.INITIAL\_CONTEXT\_FACTORY, "com.sun.jndi.ldap.LdapCtxFactory");

props.put(Context.PROVIDER\_URL, ldapSetting.getUrl());

if ((ldapSetting.getUserOn() != null) && (!ldapSetting.getUserOn().equals(""))) {

props.put(Context.SECURITY\_AUTHENTICATION, "simple");

props.put(Context.SECURITY\_PRINCIPAL, ldapSetting.getUserOn());

props.put(Context.SECURITY\_CREDENTIALS,

((ldapSetting.getPassword() == null) ? "" : ldapSetting.getPassword()));

}

return new InitialDirContext(props);

}

}

package info.smartkit.eip.controllers;

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

import org.springframework.mail.SimpleMailMessage;

import org.springframework.mail.javamail.JavaMailSender;

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.bind.annotation.RestController;

import info.smartkit.eip.dto.JsonObject;

//@see: http://www.javacodegeeks.com/2014/09/testing-mail-code-in-spring-boot-application.html

@RestController

public class MailSubmissionController {

private final JavaMailSender javaMailSender;

@Autowired

MailSubmissionController(JavaMailSender javaMailSender) {

this.javaMailSender = javaMailSender;

}

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

// @ResponseStatus(HttpStatus.CREATED)

public JsonObject send(

@RequestParam(value = "to",defaultValue="someone@localhost") String to,

@RequestParam(value = "replyTo",defaultValue="someone@localhost") String replyTo,

@RequestParam(value = "from",defaultValue="someone@localhost") String from,

@RequestParam(value = "subject",defaultValue="Lorem ipsum") String subject,

@RequestParam(value = "body",defaultValue="Lorem ipsum dolor sit amet [...]") String body) {

SimpleMailMessage mailMessage = new SimpleMailMessage();

mailMessage.setTo(to);

mailMessage.setReplyTo(replyTo);

mailMessage.setFrom(from);

mailMessage.setSubject(subject);

mailMessage.setText(body);

javaMailSender.send(mailMessage);

return new JsonObject(mailMessage);

}

}

package info.smartkit.eip.controllers;

import java.io.File;

import java.io.FileInputStream;

import java.io.FileOutputStream;

import java.io.InputStream;

import java.text.SimpleDateFormat;

import java.util.Date;

import java.util.HashMap;

import java.util.List;

import java.util.Map;

import javax.persistence.EntityManager;

import javax.persistence.PersistenceContext;

import info.smartkit.eip.models.DataBean;

import info.smartkit.eip.models.Expense;

import info.smartkit.eip.models.ExpenseDao;

import info.smartkit.eip.settings.FolderSetting;

import net.sf.jasperreports.engine.JRException;

import net.sf.jasperreports.engine.JRExporterParameter;

import net.sf.jasperreports.engine.JasperCompileManager;

import net.sf.jasperreports.engine.JasperExportManager;

import net.sf.jasperreports.engine.JasperFillManager;

import net.sf.jasperreports.engine.data.JRBeanCollectionDataSource;

import net.sf.jasperreports.engine.export.JRCsvExporter;

import net.sf.jasperreports.engine.export.JRRtfExporter;

import net.sf.jasperreports.engine.export.JRTextExporter;

import net.sf.jasperreports.engine.export.JRXlsExporter;

import org.apache.logging.log4j.LogManager;

import org.apache.logging.log4j.Logger;

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

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

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.bind.annotation.RestController;

import info.smartkit.eip.dto.JsonObject;

import info.smartkit.eip.settings.ServerSetting;

import com.wordnik.swagger.annotations.ApiOperation;

/\*\*

\* Handles Dynamic/Jasper reports download requests

\*

\* @author Krams at {@link http://krams915@blogspot.com}

\* @see http://stackoverflow.com/questions/11677572/dealing-with-xerces-hell-in-java-maven

\*/

@RestController

@RequestMapping("/report")

public class ReportController

{

private static Logger LOG = LogManager.getLogger(ReportController.class);

// @Resource(name="downloadService")

// private DownloadService downloadService;

// Auto-wire an object of type ExpenseDao

@Autowired

private ExpenseDao \_expenseDao;

@PersistenceContext

EntityManager entityManager;

@Autowired

private FolderSetting folderSetting;

// @PersistenceContext

// EntityManager entityManager;

//

@RequestMapping(method = RequestMethod.GET, params = {"owner"})

@ApiOperation(httpMethod = "GET", value = "Response a list describing all of reports that is successfully get or not.")

public JsonObject list(@RequestParam(value = "owner") String owner)

{

// return new JsonObject(this.expenseRepository.findAll());

if (owner != null) {

// Query query =

// entityManager.createQuery("SELECT status,COUNT(status) FROM expenses WHERE owner='"+owner+"' GROUP BY status");

// List<Object[]> results = query.getResultList();

// return new JsonObject(results);

// return new JsonObject(reportService.getExpensesGroupByStatus(owner));

//

Iterable<Expense> result = this.\_expenseDao.findExpensesByOwner(owner);

// // LOG.debug("itemsByOwner()result:"+result.toString());

return new JsonObject(result);

} else {

return new JsonObject(this.\_expenseDao.findAll());

}

}

public static enum DocType

{

PDF("PDF", "application/pdf"),

HTML("HTML", "application/vnd.ms-excel"),

XLS("XLS", "application/vnd.ms-excel"),

XML("XML", "text/xml"),

RTF("RTF", "application/rtf"),

CSV("CSV", "text/plain"),

TXT("TXT", "text/plain");

//

private String value;

public String getValue()

{

return value;

}

public void setValue(String value)

{

this.value = value;

}

private String contentType;

public String getContentType()

{

return contentType;

}

public void setContentType(String contentType)

{

this.contentType = contentType;

}

private DocType(String value, String contentType)

{

this.setValue(value);

this.setContentType(contentType);

}

}

/\*\*

\* @throws JRException

\* @see:

\* http://stackoverflow.com/questions/25195440/swagger-ui-with-collection-list-input-parameter-for-jax-rs-resource

\* @see: http://jasperreports.sourceforge.net/sample.reference/fonts/index.html Downloads the report as an PDF

\* format.

\* <p>

\* Make sure this method doesn't return any model. Otherwise, you'll get an

\* "IllegalStateException: getOutputStream() has already been called for this response"

\*/

@RequestMapping(value = "/doc", method = RequestMethod.POST, headers = {"Content-type=application/json"})

@ApiOperation(value = "Create(s) documents", responseContainer = "List")

// @ResponseBody,@Valid

public JsonObject getDocument(@RequestParam(value = "type") String type, @RequestBody List<DataBean> dataBeanList)

throws JRException

// public ResponseEntity<List<DataBean>> getDocument(@RequestParam(value = "type") String type,

// @RequestBody List<DataBean> dataBeanList) throws JRException

{

// @see: http://www.leveluplunch.com/java/tutorials/014-post-json-to-spring-rest-webservice/

// dataBeanList.stream().forEach(c -> c.setItems\_amount(c.getItems\_amount() + 100));

// return new ResponseEntity<List<DataBean>>(dataBeanList, HttpStatus.OK);

String docUrl = "";

String docType = "";

LOG.info("Required document type is: " + type);

// FastReport fastReport = new FastReport();

// Create a JRDataSource,the Collection used here contains dummy hard-coded objects...

// List<Expense> expensesAll = Lists.newArrayList(this.\_expenseDao.findAll());

// Log.info("expensesAll for JRBeanCollectionDataSource"+expensesAll);

// JRBeanCollectionDataSource ds = new JRBeanCollectionDataSource(expensesAll);

// LOG.debug("fastReport data(before PDF generate):" + expensesAll.toString());

//

long start = System.currentTimeMillis();

//

String jasperDestFile = JasperCompileManager.compileReportToFile(getJRXML\_SOURCE\_FILE());

//

String printFileName = null;

// DataBeanList DataBeanList = new DataBeanList();

// ArrayList dataList = DataBeanList.getDataBeanList();

// ArrayList<DataBean> dataBeanList = new ArrayList<DataBean>();

// dataBeanList.add(dataBean);

//

JRBeanCollectionDataSource beanColDataSource = new JRBeanCollectionDataSource(dataBeanList, true);

//

Map parameters = new HashMap();

LOG.info("JASPER\_DEST\_FILE:" + JASPER\_DEST\_FILE);

try {

printFileName = JasperFillManager.fillReportToFile(JASPER\_DEST\_FILE, parameters, beanColDataSource);

} catch (Exception e) {

LOG.error("JasperFillManager.fillReportToFile: " + e.toString());

e.printStackTrace();

}

if (printFileName != null) {

//

// SAXParser parser = new SAXParser();

// LOG.info("DocType.PDF.getValue() == type.toUpperCase(): "

// + DocType.PDF.getValue().toString().equals(type.toUpperCase()));

/\*\*

\* 1- export to PDF

\*/

if (DocType.PDF.getValue().toString().equals(type.toUpperCase())) {

docType = DocType.PDF.getValue().toLowerCase();

docUrl = getJASPER\_REPORT\_BASE\_FILE() + "." + docType;

try {

JasperExportManager.exportReportToPdfFile(printFileName, docUrl);

// } catch (NullPointerException ex) {

// LOG.error("NPE encountered in body: " + ex.getStackTrace().toString());

// } catch (Throwable ex) {

// LOG.error("Regular Throwable: " + ex.getMessage());

} catch (Exception e) {

LOG.error("JasperExportManager.exportReportToPdfFile: " + e.getClass().toString()

+ e.getStackTrace().toString());

e.printStackTrace();

}

}

/\*\*

\* 2- export to HTML

\*/

if (DocType.HTML.getValue().toString().equals(type.toUpperCase())) {

docType = DocType.HTML.getValue().toLowerCase();

docUrl = getJASPER\_REPORT\_BASE\_FILE() + "." + docType;

try {

JasperExportManager.exportReportToHtmlFile(printFileName, docUrl);

// } catch (NullPointerException ex) {

// LOG.error("NPE encountered in body: " + ex.getStackTrace().toString());

// } catch (Throwable ex) {

// LOG.error("Regular Throwable: " + ex.getMessage());

} catch (Exception e) {

LOG.error("JasperExportManager.exportReportToHTMLFile: " + e.getClass().toString()

+ e.getStackTrace().toString());

e.printStackTrace();

}

}

/\*\*

\* 3- export to Excel sheet

\*/

if (DocType.XLS.getValue().toString().equals(type.toUpperCase())) {

docType = DocType.XLS.getValue().toLowerCase();

docUrl = getJASPER\_REPORT\_BASE\_FILE() + "." + docType;

try {

JRXlsExporter exporter = new JRXlsExporter();

exporter.setParameter(JRExporterParameter.INPUT\_FILE\_NAME, printFileName);

exporter.setParameter(JRExporterParameter.OUTPUT\_FILE\_NAME, docUrl);

exporter.exportReport();

} catch (Exception e) {

LOG.error("JasperExportManager.exportReportToXLSFile: " + e.getClass().toString()

+ e.getStackTrace().toString());

e.printStackTrace();

}

}

/\*\*

\* 4- export to XML

\*/

if (DocType.XML.getValue().toString().equals(type.toUpperCase())) {

docType = DocType.XML.getValue().toLowerCase();

docUrl = getJASPER\_REPORT\_BASE\_FILE() + "." + docType;

try {

JasperExportManager.exportReportToXmlFile(printFileName, docUrl, true);

// } catch (NullPointerException ex) {

// LOG.error("NPE encountered in body: " + ex.getStackTrace().toString());

// } catch (Throwable ex) {

// LOG.error("Regular Throwable: " + ex.getMessage());

} catch (Exception e) {

LOG.error("JasperExportManager.exportReportToXMLFile: " + e.getClass().toString()

+ e.getStackTrace().toString());

e.printStackTrace();

}

}

/\*\*

\* 5- export to TXT

\*/

if (DocType.TXT.getValue().toString().equals(type.toUpperCase())) {

docType = DocType.CSV.getValue().toLowerCase();

docUrl = getJASPER\_REPORT\_BASE\_FILE() + "." + docType;

try {

JRTextExporter exporter = new JRTextExporter();

exporter.setParameter(JRExporterParameter.INPUT\_FILE\_NAME, printFileName);

exporter.setParameter(JRExporterParameter.OUTPUT\_FILE\_NAME, docUrl);

exporter.exportReport();

} catch (Exception e) {

LOG.error("JasperExportManager.exportReportToXLSFile: " + e.getClass().toString()

+ e.getStackTrace().toString());

e.printStackTrace();

}

}

/\*\*

\* 6- export to RTF

\*/

if (DocType.RTF.getValue().toString().equals(type.toUpperCase())) {

docType = DocType.RTF.getValue().toLowerCase();

docUrl = getJASPER\_REPORT\_BASE\_FILE() + "." + docType;

try {

JRRtfExporter exporter = new JRRtfExporter();

exporter.setParameter(JRExporterParameter.INPUT\_FILE\_NAME, printFileName);

exporter.setParameter(JRExporterParameter.OUTPUT\_FILE\_NAME, docUrl);

exporter.exportReport();

} catch (Exception e) {

LOG.error("JasperExportManager.exportReportToXLSFile: " + e.getClass().toString()

+ e.getStackTrace().toString());

e.printStackTrace();

}

}

/\*\*

\* 7- export to CSV

\*/

if (DocType.CSV.getValue().toString().equals(type.toUpperCase())) {

docType = DocType.CSV.getValue().toLowerCase();

docUrl = getJASPER\_REPORT\_BASE\_FILE() + "." + docType;

try {

JRCsvExporter exporter = new JRCsvExporter();

exporter.setParameter(JRExporterParameter.INPUT\_FILE\_NAME, printFileName);

exporter.setParameter(JRExporterParameter.OUTPUT\_FILE\_NAME, docUrl);

exporter.exportReport();

} catch (Exception e) {

LOG.error("JasperExportManager.exportReportToXLSFile: " + e.getClass().toString()

+ e.getStackTrace().toString());

e.printStackTrace();

}

}

}

LOG.info("Report document created time : " + (System.currentTimeMillis() - start) + ",docType:" + docType);

// e.g. http://localhost:8082/api/reports/com.rushucloud.eip.reports.FastReport.pdf

LOG.info("Report document file:" + docUrl);// /reports/jasper\_report\_template\_1234567\_2015-03-26-15-34-27-948.csv

// File save as.

String newPath = this.getDocumentUrl(dataBeanList.get(0).getExpenses\_id(), docType);

this.copyFile(docUrl, newPath);

docUrl = this.getUrlPath() + newPath;

LOG.info("Report document url:" + docUrl);// e.g: http://localhost:8082/api/reports/jasper\_report\_template.pdf

//

return new JsonObject(docUrl);

}

public String getClassPath()

{

String classPath = this.getClass().getResource("/").getPath();

return classPath;

}

private String getUrlPath()

{

return ServerSetting.getInstance().getUrl() + folderSetting.getReports();

}

public void copyFile(String oldPath, String newPath)

{

try {

int bytesum = 0;

int byteread = 0;

File oldfile = new File(oldPath);

if (oldfile.exists()) { //

InputStream inStream = new FileInputStream(oldPath); //

FileOutputStream fs = new FileOutputStream(newPath);

byte[] buffer = new byte[1444];

int length;

while ((byteread = inStream.read(buffer)) != -1) {

bytesum += byteread; //

System.out.println(bytesum);

fs.write(buffer, 0, byteread);

}

inStream.close();

}

} catch (Exception e) {

e.printStackTrace();

LOG.error(e.toString());

}

}

private final String JASPER\_REPORT\_BASE\_NAME = "jasper\_report\_template";

// private final String JASPER\_REPORT\_BASE\_URL = getUrlPath() + "/reports/" + JASPER\_REPORT\_BASE\_NAME;

// private final String JASPER\_REPORT\_BASE\_FILE = getClassPath() + "/reports/" + JASPER\_REPORT\_BASE\_NAME;

// private final String JRXML\_SOURCE\_FILE = JASPER\_REPORT\_BASE\_FILE + ".jrxml";

private String getJRXML\_SOURCE\_FILE()

{

return getClassPath() + "/reports/" + JASPER\_REPORT\_BASE\_NAME + ".jrxml";

}

private String getJASPER\_REPORT\_BASE\_URL()

{

return getUrlPath() + "/reports/" + JASPER\_REPORT\_BASE\_NAME;

}

private String getJASPER\_REPORT\_BASE\_FILE()

{

return getClassPath() + "/reports/" + JASPER\_REPORT\_BASE\_NAME;

}

private final String JASPER\_DEST\_FILE = getJASPER\_REPORT\_BASE\_FILE() + ".jasper";

private String getDocumentUrl(long reportId, String fileExt)

{

String docIdentifier =

JASPER\_REPORT\_BASE\_NAME + "\_" + Long.toString(reportId) + "\_"

+ new SimpleDateFormat("yyyy-MM-dd-HH-mm-ss-SSS").format(new Date()) + "." + fileExt;

return docIdentifier;

}

// @ExceptionHandler(IllegalArgumentException.class)

// public void handleBadRequests(HttpServletResponse response) throws IOException

// {

// response.sendError(HttpStatus.BAD\_REQUEST.value(), response.toString());

// }

}

package info.smartkit.eip.controllers;

import javax.validation.Valid;

import info.smartkit.eip.dto.JsonObject;

import info.smartkit.eip.models.Tag;

import info.smartkit.eip.models.TagDao;

import info.smartkit.eip.models.TagRepository;

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

import org.springframework.data.domain.Sort;

import org.springframework.http.HttpStatus;

import org.springframework.http.ResponseEntity;

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

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

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

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

import org.springframework.web.bind.annotation.RestController;

import com.wordnik.swagger.annotations.ApiOperation;

@RestController

@RequestMapping("/tags")

public class TagsController {

// ==============

// PRIVATE FIELDS

// ==============

// Autowire an object of type TagDao

@Autowired

private TagDao \_tagDao;

//

private TagRepository tagRepository;

@Autowired

public TagsController(TagRepository tagRepository) {

this.tagRepository = tagRepository;

}

// ==============

// PUBLIC METHODS

// ==============

@RequestMapping(method = RequestMethod.POST)

@ApiOperation(httpMethod = "POST", value = "Response a string describing if the reimbursement tag is successfully created or not.")

public Tag create(@RequestBody @Valid Tag tag) {

return this.tagRepository.save(tag);

}

@RequestMapping(method = RequestMethod.GET)

@ApiOperation(httpMethod = "GET", value = "Response a list describing all of tag that is successfully get or not.")

public JsonObject list() {

return new JsonObject(this.tagRepository.findAll(new Sort(

new Sort.Order(Sort.Direction.ASC, "date"))));

}

@RequestMapping(value = "/{id}", method = RequestMethod.GET)

@ApiOperation(httpMethod = "GET", value = "Response a string describing if the tag id is successfully get or not.")

public Tag get(@PathVariable("id") long id) {

return this.tagRepository.findOne(id);

}

@RequestMapping(value = "/{id}", method = RequestMethod.PUT)

@ApiOperation(httpMethod = "PUT", value = "Response a string describing if the reimbursement tag is successfully updated or not.")

public Tag update(@PathVariable("id") long id, @RequestBody @Valid Tag tag) {

return tagRepository.save(tag);

}

@RequestMapping(value = "/{id}", method = RequestMethod.DELETE)

@ApiOperation(httpMethod = "DELETE", value = "Response a string describing if the tag is successfully delete or not.")

public ResponseEntity<Boolean> delete(@PathVariable("id") long id) {

this.tagRepository.delete(id);

return new ResponseEntity<Boolean>(Boolean.TRUE, HttpStatus.OK);

}

}

package info.smartkit.eip.controllers;

import java.util.List;

import javax.naming.directory.BasicAttribute;

import javax.naming.directory.BasicAttributes;

import javax.validation.Valid;

import info.smartkit.eip.models.SimplePerson;

import info.smartkit.eip.models.User;

import info.smartkit.eip.settings.LDAPSetting;

import org.apache.logging.log4j.LogManager;

import org.apache.logging.log4j.Logger;

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

import org.springframework.http.HttpStatus;

import org.springframework.http.ResponseEntity;

import org.springframework.security.web.bind.annotation.AuthenticationPrincipal;

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

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

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.bind.annotation.RestController;

import info.smartkit.eip.dto.JsonObject;

import info.smartkit.eip.ldap.plain.dao.PersonDao;

import info.smartkit.eip.ldap.plain.domain.Person;

import com.wordnik.swagger.annotations.ApiOperation;

@RestController

@RequestMapping("/users")

public class UserController

{

//

private static Logger LOG = LogManager.getLogger(UserController.class);

// ==============

// PRIVATE FIELDS

// ==============

@Autowired

private PersonDao personDao;

// @see

// http://hoserdude.com/2014/06/19/spring-boot-configurationproperties-and-profile-management-using-yaml/

@Autowired

private LDAPSetting ldapSetting;

// ==============

// PUBLIC METHODS

// ==============

@RequestMapping(method = RequestMethod.POST)

@ApiOperation(httpMethod = "POST", value = "Response a string describing if the reimbursement company is successfully created or not.")

public JsonObject create(@RequestBody @Valid SimplePerson simplePerson,

@RequestParam(value = "group", required = true, defaultValue = "employees") String group,

@RequestParam(value = "company", required = true, defaultValue = "www1.rushucloud.com") String company)

{

//

BasicAttributes personAttributes = new BasicAttributes();

BasicAttribute personBasicAttribute = new BasicAttribute("objectclass");

// personBasicAttribute.add("person");

personBasicAttribute.add("top");

personBasicAttribute.add("extensibleObject");

// this is the object class extension.

// TODO:@see:

// http://forum.spring.io/forum/spring-projects/data/ldap/90794-how-to-insert-multi-value-attribute-in-ad

// personBasicAttribute.add("wxToken");

// personBasicAttribute.add("code");

personBasicAttribute.add("person");

personBasicAttribute.add("organizationalperson");

personBasicAttribute.add("inetorgperson");

//

personAttributes.put(personBasicAttribute);

personAttributes.put("cn", "cn\_" + simplePerson.getUsername());

personAttributes.put("sn", "sn\_" + simplePerson.getUsername());

personAttributes.put("uid", simplePerson.getUsername());

personAttributes.put("postalAddress", simplePerson.getWxToken());// wx\_token

personAttributes.put("postalCode", simplePerson.getCode());// code

personAttributes.put("mobile", simplePerson.getMobile());

personAttributes.put("mail", simplePerson.getEmail());

personAttributes.put("userPassword", simplePerson.getPassword());

// personAttributes.put("wxToken", simplePerson.getWxToken());

// personAttributes.put("code", simplePerson.getCode());

// BasicAttribute wxTokenAttribute = new BasicAttribute("wxToken", simplePerson.getWxToken());

// wxTokenAttribute.add(simplePerson.getWxToken());

// personAttributes.put("wxToken", wxTokenAttribute);

// BasicAttribute codeAttribute = new BasicAttribute("code", simplePerson.getCode());

// codeAttribute.add(simplePerson.getCode());

// personAttributes.put("code", codeAttribute);

personDao.getLdapTemplate().bind("uid=" + simplePerson.getUsername() + ",ou=" + group + ",ou=" + company, null,

personAttributes);

//

return new JsonObject(simplePerson);

}

@RequestMapping(method = RequestMethod.GET)

@ApiOperation(httpMethod = "GET", value = "Response a list describing all of peron entry that is successfully get or not.")

public JsonObject list(@AuthenticationPrincipal User user,

@RequestParam(value = "groups", required = true, defaultValue = "employees,www1.rushucloud.com") String groups)

{

// List<Person> persons = personDao.findAll();

String[] strGroups = groups.split(",");

String plainGroups = "";

for (int i = 0; i < strGroups.length; i++) {

plainGroups += ",ou=" + strGroups[i];

}

List<Person> persons = personDao.findAll(plainGroups);

return new JsonObject(persons);

}

@RequestMapping(value = "/{uid}", method = RequestMethod.PUT)

@ApiOperation(httpMethod = "PUT", value = "Response a entry describing if the person entry is successfully updated or not.")

public JsonObject update(@PathVariable("uid") String uid, @RequestBody @Valid Person person)

{

//

personDao.update(person);

return new JsonObject(personDao.findByUid(uid));

}

@RequestMapping(method = RequestMethod.DELETE)

@ApiOperation(httpMethod = "DELETE", value = "Response a result describing if the peron entry is successfully delete or not.")

public ResponseEntity<Boolean> delete(@RequestBody @Valid Person person)

{

personDao.delete(person);

return new ResponseEntity<Boolean>(Boolean.TRUE, HttpStatus.OK);

}

@RequestMapping(value = "/{uid}", method = RequestMethod.GET)

@ApiOperation(httpMethod = "GET", value = "Response a entry describing if the person entry is successfully retrieved or not.")

public JsonObject get(@PathVariable("uid") String uid)

{

//

return new JsonObject(personDao.findByUid(uid));

}

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

@ApiOperation(httpMethod = "GET", value = "LDAP client for authenticate user.")

public boolean ldapAuthenticate(

@RequestParam(value = "uid", required = true, defaultValue = "employee0") String uid,

@RequestParam(value = "password", required = true, defaultValue = "passwordpassword") String password,

@RequestParam(value = "groups", required = true, defaultValue = "employees,www1.rushucloud.com") String groups)

{

String[] strGroups = groups.split(",");

String plainGroups = "";

for (int i = 0; i < strGroups.length; i++) {

plainGroups += ",ou=" + strGroups[i];

}

return personDao.getLdapTemplate().authenticate("uid=" + uid + plainGroups, "(objectClass=Person)", password);

}

}

package info.smartkit.eip.controllers;

import java.io.BufferedReader;

import java.io.IOException;

import java.io.InputStreamReader;

import java.util.Arrays;

import java.util.HashMap;

import java.util.Map;

import java.util.Map.Entry;

import info.smartkit.eip.models.VendorDao;

import org.activiti.engine.impl.util.json.JSONObject;

import org.apache.commons.httpclient.HttpClient;

import org.apache.commons.httpclient.HttpMethod;

import org.apache.commons.httpclient.URIException;

import org.apache.commons.httpclient.methods.GetMethod;

import org.apache.commons.httpclient.params.HttpClientParams;

import org.apache.commons.httpclient.util.URIUtil;

import org.apache.commons.lang.StringUtils;

import org.apache.http.client.ClientProtocolException;

import org.apache.logging.log4j.LogManager;

import org.apache.logging.log4j.Logger;

import org.json.simple.parser.ParseException;

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

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.bind.annotation.RestController;

import info.smartkit.eip.consts.ThirdPartyConstants;

import info.smartkit.eip.dto.JsonObject;

import info.smartkit.eip.models.VendorRepository;

import info.smartkit.eip.utils.ParserTools;

import com.wordnik.swagger.annotations.ApiOperation;

@RestController

@RequestMapping("/vendors")

// @see: http://java.dzone.com/articles/spring-rest-controller

public class VendorsController {

private Logger LOG = LogManager.getLogger(VendorsController.class);

// ==============

// PRIVATE FIELDS

// ==============

// Autowire an object of type CompanyDao

@Autowired

private VendorDao \_vendorDao;

//

private VendorRepository vendorRepository;

@Autowired

public VendorsController(VendorRepository vendorRepository) {

this.vendorRepository = vendorRepository;

}

// ==============

// PUBLIC METHODS

// ==============

@RequestMapping(method = RequestMethod.GET, params = { "latitude",

"longitude", "category" })

@ApiOperation(httpMethod = "GET", value = "Response a list describing all of vendors that is successfully get or not.")

public JsonObject list(

@RequestParam(value = "latitude", defaultValue = "31.21524") String latitude,

@RequestParam(value = "longitude", defaultValue = "121.420033") String longitude,

@RequestParam(value = "category", defaultValue = "美食") String category)

throws ClientProtocolException, IOException, ParseException {

// return new JsonObject(this.vendorRepository.findAll());

// @see: http://www.tuicool.com/articles/3aIFrm

String url\_geocoder = "http://api.map.baidu.com/geocoder/v2/?ak="

+ ThirdPartyConstants.BAIDU\_AK + "&location=" + latitude + ","

+ longitude + "&output=json&pois=0";

String url2ParsedStr = ParserTools.doGet(url\_geocoder);

// String encodedJsonStr = URLEncoder.encode(url2ParsedStr, "utf-8");

// {"status":0,"result":{"location":{"lng":121.4200330434,"lat":31.215239865509},"formatted\_address":"上海市长宁区天山路1890弄-1-～41号","business":"天山路,遵义路,虹桥","addressComponent":{"city":"上海市","direction":"附近","distance":"24","district":"长宁区","province":"上海市","street":"天山路","street\_number":"1890弄-1-～41号"},"poiRegions":[],"cityCode":289}}

//

JSONObject jObject = new JSONObject(url2ParsedStr);

JSONObject resultObject = jObject.getJSONObject("result");

JSONObject addressComponentObject = resultObject

.getJSONObject("addressComponent");

String cityStr = addressComponentObject.getString("city");

cityStr = cityStr.substring(0, cityStr.length()-1);

String districtStr = addressComponentObject.getString("district");

//

LOG.info("Geocoder by BAIDU,city :" + url\_geocoder

+ ",then decoding JSON:" + resultObject.toString()+",city:"+cityStr);

//Call DZDP API.

Map<String, String> paramMap = new HashMap<String, String>();

paramMap.put("city", cityStr);//Notice:"上海市" to "上海"

paramMap.put("latitude", latitude);

paramMap.put("longitude", longitude);

paramMap.put("category", category);

paramMap.put("region", districtStr);

paramMap.put("limit", "20");

paramMap.put("radius", "2000");

paramMap.put("offset\_type", "0");

// paramMap.put("has\_coupon", "1");

// paramMap.put("has\_deal", "1");

// paramMap.put("keyword", "泰国菜");

paramMap.put("sort", "7");

paramMap.put("format", "json");

String requestResult = requestApi(ThirdPartyConstants.DZDP\_apiUrl,

ThirdPartyConstants.DZDP\_appKey,

ThirdPartyConstants.DZDP\_secret, paramMap);

//

return new JsonObject(requestResult);

}

@SuppressWarnings("unused")

private String requestApi(String apiUrl, String appKey, String secret,

Map<String, String> paramMap) {

String queryString = getQueryString(appKey, secret, paramMap);

StringBuffer response = new StringBuffer();

HttpClientParams httpConnectionParams = new HttpClientParams();

httpConnectionParams.setConnectionManagerTimeout(1000);

HttpClient client = new HttpClient(httpConnectionParams);

HttpMethod method = new GetMethod(apiUrl);

try {

if (StringUtils.isNotBlank(queryString)) {

// Encode query string with UTF-8

String encodeQuery = URIUtil.encodeQuery(queryString, "UTF-8");

LOG.debug("Encoded Query:" + encodeQuery);

method.setQueryString(encodeQuery);

}

client.executeMethod(method);

BufferedReader reader = new BufferedReader(new InputStreamReader(

method.getResponseBodyAsStream(), "UTF-8"));

String line = null;

while ((line = reader.readLine()) != null) {

response.append(line).append(

System.getProperty("line.separator"));

}

reader.close();

} catch (URIException e) {

LOG.error("Can not encode query: " + queryString

+ " with charset UTF-8. ", e);

} catch (IOException e) {

LOG.error("Request URL: " + apiUrl + " failed. ", e);

} finally {

method.releaseConnection();

}

//

// LOG.info("DZDP API response:"+response);

return response.toString();

// return new JSONObject(response.toString());

}

private String getQueryString(String appKey, String secret,

Map<String, String> paramMap) {

String sign = sign(appKey, secret, paramMap);

// 添加签名

StringBuilder stringBuilder = new StringBuilder();

stringBuilder.append("appkey=").append(appKey).append("&sign=")

.append(sign);

for (Entry<String, String> entry : paramMap.entrySet()) {

stringBuilder.append('&').append(entry.getKey()).append('=')

.append(entry.getValue());

}

String queryString = stringBuilder.toString();

return queryString;

}

private String sign(String appKey, String secret,

Map<String, String> paramMap) {

// 对参数名进行字典排序

String[] keyArray = paramMap.keySet().toArray(new String[0]);

Arrays.sort(keyArray);

// 拼接有序的参数名-值串

StringBuilder stringBuilder = new StringBuilder();

stringBuilder.append(appKey);

for (String key : keyArray) {

stringBuilder.append(key).append(paramMap.get(key));

}

stringBuilder.append(secret);

String codes = stringBuilder.toString();

// SHA-1编码， 这里使用的是Apache

// codec，即可获得签名(shaHex()会首先将中文转换为UTF8编码然后进行sha1计算，使用其他的工具包请注意UTF8编码转换)

/\*

\* 以下sha1签名代码效果等同 byte[] sha =

\* org.apache.commons.codec.digest.DigestUtils

\* .sha(org.apache.commons.codec

\* .binary.StringUtils.getBytesUtf8(codes)); String sign =

\* org.apache.commons

\* .codec.binary.Hex.encodeHexString(sha).toUpperCase();

\*/

String sign = org.apache.commons.codec.digest.DigestUtils.shaHex(codes)

.toUpperCase();

return sign;

}

/\*

\* @RequestMapping(method = RequestMethod.POST)

\*

\* @ApiOperation(httpMethod = "POST", value =

\* "Response a string describing if the vendor is successfully created or not."

\* ) public Vendor create(@RequestBody @Valid Vendor vendor) { return

\* this.vendorRepository.save(vendor); }

\*

\* @RequestMapping(method = RequestMethod.GET)

\*

\* @ApiOperation(httpMethod = "GET", value =

\* "Response a list describing all of vendors that is successfully get or not."

\* ) public JsonObject list() { return new

\* JsonObject(this.vendorRepository.findAll()); }

\*

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

\*

\* @ApiOperation(httpMethod = "GET", value =

\* "Response a string describing if the vendor id is successfully get or not."

\* ) public Vendor get(@PathVariable("id") long id) { return

\* this.vendorRepository.findOne(id); }

\*

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

\*

\* @ApiOperation(httpMethod = "PUT", value =

\* "Response a string describing if the reimbursement item is successfully updated or not."

\* ) public Vendor update(@PathVariable("id") long id,

\*

\* @RequestBody @Valid Vendor vendor) { return

\* vendorRepository.save(vendor); }

\*

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

\*

\* @ApiOperation(httpMethod = "DELETE", value =

\* "Response a string describing if the item is successfully delete or not."

\* ) public ResponseEntity<Boolean> delete(@PathVariable("id") long id) {

\* this.vendorRepository.delete(id); return new

\* ResponseEntity<Boolean>(Boolean.TRUE, HttpStatus.OK); }

\*/

}package info.smartkit.eip.dto;

public class JsonObject {

public JsonObject(Object data) {

this.data = data;

}

// Setter,getters

private Object data;

public Object getData() {

return data;

}

public void setData(Object data) {

this.data = data;

}

}

package info.smartkit.eip.dto;

public class JsonString {

public JsonString(String body) {

this.body = body;

}

// Setter,getters

private String body;

public String getBody() {

return body;

}

public void setBody(String body) {

this.body = body;

}

}

/\*

\* Copyright 2005-2013 the original author or authors.

\*

\* Licensed under the Apache License, Version 2.0 (the "License");

\* you may not use this file except in compliance with the License.

\* You may obtain a copy of the License at

\*

\* http://www.apache.org/licenses/LICENSE-2.0

\*

\* Unless required by applicable law or agreed to in writing, software

\* distributed under the License is distributed on an "AS IS" BASIS,

\* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.

\* See the License for the specific language governing permissions and

\* limitations under the License.

\*/

package info.smartkit.eip.ldap.odm.dao;

import static org.springframework.ldap.query.LdapQueryBuilder.query;

import java.util.List;

import javax.naming.NamingException;

import javax.naming.directory.Attributes;

import javax.naming.ldap.LdapName;

import info.smartkit.eip.ldap.plain.domain.Person;

import org.apache.logging.log4j.LogManager;

import org.apache.logging.log4j.Logger;

import org.springframework.ldap.core.AttributesMapper;

import org.springframework.ldap.core.LdapTemplate;

import org.springframework.ldap.support.LdapNameBuilder;

import info.smartkit.eip.ldap.plain.dao.PersonDao;

/\*\*

\* Default implementation of PersonDao. This implementation uses the Object-Directory Mapping feature, which requires

\* the entity classes to be annotated, but relieves the programmer from the tedious task of mapping to and from entity

\* objects, using attribute or dn component values.

\*

\* @author Mattias Hellborg Arthursson

\*/

public class PersonDaoImpl implements PersonDao

{

private static Logger LOG = LogManager.getLogger(PersonDaoImpl.class);

private LdapTemplate ldapTemplate;

@Override

public LdapTemplate getLdapTemplate()

{

return ldapTemplate;

}

@Override

public void create(Person person)

{

ldapTemplate.create(person);

}

@Override

public void update(Person person)

{

ldapTemplate.update(person);

}

@Override

public void delete(Person person)

{

ldapTemplate.delete(ldapTemplate.findByDn(buildDn(person), Person.class));

}

@Override

public List<String> getAllPersonNames()

{

return ldapTemplate.search(query().attributes("cn").where("objectclass").is("person"),

new AttributesMapper<String>()

{

@Override

public String mapFromAttributes(Attributes attrs) throws NamingException

{

return attrs.get("cn").get().toString();

}

});

}

@Override

public List<String> getAllPersonUids()

{

return ldapTemplate.search(query().attributes("cn").where("objectclass").is("person"),

new AttributesMapper<String>()

{

@Override

public String mapFromAttributes(Attributes attrs) throws NamingException

{

return attrs.get("uid").get().toString();

}

});

}

@Override

public List<Person> findAll()

{

return ldapTemplate.findAll(Person.class);

}

@Override

public List<Person> findAll(String baseOn)

{

List<Person> persons = ldapTemplate.find(query().base(baseOn).filter("(objectClass=person)"), Person.class);

// List<Person> persons =

// ldapTemplate.search(query().base(baseOn).filter("(objectClass=person)"), new PersonAttributesMapper());

// LOG.info("ldap search query:" + query().toString());

// for (Person person : persons) {

// LOG.info("ldap searched person:" + person.toString());

// }

return persons;

// return ldapTemplate.findAll(Person.class);

}

@Override

public Person findByPrimaryKey(String country, String company, String fullname)

{

LdapName dn = buildDn(country, company, fullname);

Person person = ldapTemplate.findByDn(dn, Person.class);

return person;

}

@Override

public Person findByUid(String uid)

{

Person person = ldapTemplate.findOne(query().where("uid").is(uid), Person.class);

return person;

}

private LdapName buildDn(Person person)

{

return buildDn(person.getCn(), person.getOu(), person.getUid());

}

private LdapName buildDn(String country, String company, String completeName)

{

return LdapNameBuilder.newInstance().add("c", country).add("ou", company).add("cn", completeName).build();

}

public void setLdapTemplate(LdapTemplate ldapTemplate)

{

this.ldapTemplate = ldapTemplate;

}

}

/\*

\* Copyright 2005-2010 the original author or authors.

\*

\* Licensed under the Apache License, Version 2.0 (the "License");

\* you may not use this file except in compliance with the License.

\* You may obtain a copy of the License at

\*

\* http://www.apache.org/licenses/LICENSE-2.0

\*

\* Unless required by applicable law or agreed to in writing, software

\* distributed under the License is distributed on an "AS IS" BASIS,

\* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.

\* See the License for the specific language governing permissions and

\* limitations under the License.

\*/

package info.smartkit.eip.ldap.plain.dao;

import java.util.List;

import info.smartkit.eip.ldap.plain.domain.Person;

import org.springframework.ldap.core.LdapTemplate;

/\*\*

\* Data Access Object interface for the Person entity.

\*

\* @author Mattias Hellborg Arthursson

\* @author Ulrik Sandberg

\*/

public interface PersonDao

{

void create(Person person);

void update(Person person);

void delete(Person person);

List<String> getAllPersonNames();

List<String> getAllPersonUids();

List<Person> findAll();

List<Person> findAll(String baseOn);

Person findByPrimaryKey(String country, String company, String fullname);

Person findByUid(String uid);

LdapTemplate getLdapTemplate();

}

/\*

\* Copyright 2005-2013 the original author or authors.

\*

\* Licensed under the Apache License, Version 2.0 (the "License");

\* you may not use this file except in compliance with the License.

\* You may obtain a copy of the License at

\*

\* http://www.apache.org/licenses/LICENSE-2.0

\*

\* Unless required by applicable law or agreed to in writing, software

\* distributed under the License is distributed on an "AS IS" BASIS,

\* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.

\* See the License for the specific language governing permissions and

\* limitations under the License.

\*/

package info.smartkit.eip.ldap.plain.domain;

import javax.naming.Name;

import org.apache.commons.lang.builder.EqualsBuilder;

import org.apache.commons.lang.builder.HashCodeBuilder;

import org.apache.commons.lang.builder.ToStringBuilder;

import org.apache.commons.lang.builder.ToStringStyle;

import org.springframework.ldap.odm.annotations.Attribute;

import org.springframework.ldap.odm.annotations.DnAttribute;

import org.springframework.ldap.odm.annotations.Entry;

import org.springframework.ldap.odm.annotations.Id;

import org.springframework.ldap.odm.annotations.Transient;

import com.fasterxml.jackson.annotation.JsonIgnore;

import com.fasterxml.jackson.annotation.JsonIgnoreProperties;

/\*\*

\* Simple class representing a single person.

\*

\* @author Mattias Hellborg Arthursson

\* @author Ulrik Sandberg

\* @see: http://oav.net/mirrors/LDAP-ObjectClasses.html

\*/

@JsonIgnoreProperties(ignoreUnknown = true)

@Entry(objectClasses = {"inetOrgPerson", "organizationalPerson", "person", "top"})

public class Person

{

// @see:http://www.jayxu.com/2011/12/14/13131/

// @JsonIgnoreProperties("dn")

@Id

private Name dn;

@Attribute(name = "uid")

@DnAttribute(value = "uid", index = 2)

private String uid;

public String getUid()

{

return uid;

}

public void setUid(String uid)

{

this.uid = uid;

}

@Attribute(name = "cn")

@DnAttribute(value = "cn", index = 1)

private String cn;

public String getCn()

{

return cn;

}

public void setCn(String cn)

{

this.cn = cn;

}

@Attribute(name = "sn")

private String sn;

public String getSn()

{

return sn;

}

public void setSn(String sn)

{

this.sn = sn;

}

@Attribute(name = "description")

private String description;

@Transient

@DnAttribute(value = "ou", index = 0)

private String ou;

public String getOu()

{

return ou;

}

public void setOu(String ou)

{

this.ou = ou;

}

@Attribute(name = "mobile")

private String mobile;

public String getMobile()

{

return mobile;

}

public void setMobile(String mobile)

{

this.mobile = mobile;

}

@JsonIgnore

public Name getDn()

{

return dn;

}

public void setDn(Name dn)

{

this.dn = dn;

}

public String getDescription()

{

return description;

}

public void setDescription(String description)

{

this.description = description;

}

@Attribute(name = "postalAddress")

private String wxToken;

public String getWxToken()

{

return wxToken;

}

public void setWxToken(String wxToken)

{

this.wxToken = wxToken;

}

@Attribute(name = "postalCode")

private String code;

public String getCode()

{

return code;

}

public void setCode(String code)

{

this.code = code;

}

@Attribute(name = "jpegPhoto")

private String jpegPhoto;

public String getJpegPhoto()

{

return jpegPhoto;

}

public void setJpegPhoto(String jpegPhoto)

{

this.jpegPhoto = jpegPhoto;

}

@Override

public boolean equals(Object obj)

{

return EqualsBuilder.reflectionEquals(this, obj);

}

@Override

public int hashCode()

{

return HashCodeBuilder.reflectionHashCode(this);

}

@Override

public String toString()

{

return ToStringBuilder.reflectionToString(this, ToStringStyle.MULTI\_LINE\_STYLE);

}

}

package info.smartkit.eip.models;

import java.util.Set;

import javax.persistence.CascadeType;

import javax.persistence.Entity;

import javax.persistence.FetchType;

import javax.persistence.GeneratedValue;

import javax.persistence.GenerationType;

import javax.persistence.Id;

import javax.persistence.JoinColumn;

import javax.persistence.ManyToOne;

import javax.persistence.OneToMany;

import javax.persistence.Table;

import javax.validation.constraints.NotNull;

import com.fasterxml.jackson.annotation.JsonBackReference;

@Entity

@Table(name = "categories")

//@Embeddable

public class Category extends ModelBase {

// ==============

// PRIVATE FIELDS

// ==============

// An auto-generated id (unique for each user in the db)

@Id

@GeneratedValue(strategy = GenerationType.AUTO)

private long id;

public long getId() {

return id;

}

public void setId(long value) {

this.id = value;

}

// The category icon

@NotNull

private String icon;

public String getIcon() {

return icon;

}

public void setIcon(String icon) {

this.icon = icon;

}

// The category name

@NotNull

private String name;

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

// The category label for displaying name.

@NotNull

private String label;

public String getLabel() {

return label;

}

public void setLabel(String label) {

this.label = label;

}

// @see: http://stackoverflow.com/questions/14388037/create-a-tree-using-jpa

// This field is a table column

// It identifies the parent of the current row

// It it will be written as the type of categoryId

// By default this relationship will be eagerly fetched

// , which you may or may not want

@ManyToOne(fetch = FetchType.LAZY, cascade = { CascadeType.PERSIST,CascadeType.MERGE })

// @ManyToOne

@JoinColumn(name="parent\_id")

@JsonBackReference

private Category parent;

public Category getParent() {

return parent;

}

public void setParent(Category parent) {

this.parent = parent;

}

// This field is not a table column

// It is a collection of those Category rows that have this row as a parent.

// This is the other side of the relationship defined by the parent field.

// @OneToMany(mappedBy = "parent")

@OneToMany(cascade=CascadeType.PERSIST,fetch=FetchType.LAZY)

@JoinColumn(name="parent\_id")

private Set<Category> children;

public Set<Category> getChildren() {

return children;

}

public void setChildren(Set<Category> children) {

this.children = children;

}

// ==============

// PUBLIC METHODS

// ==============

public Category() {

}

public Category(long id) {

this.id = id;

}

//Root category

public Category(String icon, String name) {

this.icon = icon;

this.name = name;

}

//Child category

public Category(String icon, String name, Category parent,

Set<Category> children) {

this.icon = icon;

this.name = name;

this.parent = parent;

this.children = children;

}

}

package info.smartkit.eip.models;

import javax.transaction.Transactional;

import org.springframework.data.repository.CrudRepository;

@Transactional

public interface CategoryDao extends CrudRepository<Category, Long> {

/\*\*

\* This method is not implemented and its working code will be

\* auto-magically generated from its signature by Spring Data JPA.

\*

\* @param id

\* the category id.

\* @return the user having the passed id or null if no category is found.

\*/

public Category findById(long id);

}

package info.smartkit.eip.models;

import java.util.List;

import org.springframework.data.repository.PagingAndSortingRepository;

import org.springframework.data.repository.query.Param;

import org.springframework.data.rest.core.annotation.RepositoryRestResource;

@RepositoryRestResource(collectionResourceRel="category",path="category")

public interface CategoryRepository extends PagingAndSortingRepository<Category, Long> {

List<Category> findById(@Param("id") long id);

}

/\*\*

\* @author yangboz

\* @see http://blog.netgloo.com/2014/10/27/using-mysql-in-spring-boot-via-spring-data-jpa-and-hibernate/

\*/

package info.smartkit.eip.models;

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 javax.persistence.Version;

import javax.validation.constraints.NotNull;

@Entity

@Table(name = "companies")

public class Company extends ModelBase{

//public class Company{

// ==============

// PRIVATE FIELDS

// ==============

// An auto-generated id (unique for each user in the db)

@Id

@GeneratedValue(strategy = GenerationType.AUTO)

private long id = 0;

public long getId() {

return id;

}

public void setId(long value) {

this.id = value;

}

// The company email

@NotNull

private String email = "demo@rushucloud.com";

public String getEmail() {

return email;

}

public void setEmail(String email) {

this.email = email;

}

// The company domain

@NotNull

private String domain = "rushucloud.com";

public String getDomain() {

return domain;

}

public void setDomain(String domain) {

this.domain = domain;

}

// The company name

@NotNull

private String name;

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

// The company business keys

@NotNull

private String businessKey = "reimbursementRequest";//default value for Reimbursement.

//

public String getBusinessKey() {

return businessKey;

}

public void setBusinessKey(String businessKey) {

this.businessKey = businessKey;

}

// @see: http://stackoverflow.com/questions/2572566/java-jpa-version-annotation

@Version

@Column(name = "version", columnDefinition = "integer DEFAULT 0", nullable = false)

private long version = 0L;

// ==============

// PUBLIC METHODS

// ==============

public Company() {

}

public Company(long id) {

this.id = id;

}

public Company(String email, String name, String domain,String businessKey) {

this.email = email;

this.name = name;

this.domain = domain;

this.businessKey = businessKey;

}

}

/\*\*

\* @author yangboz

\* @see http://blog.netgloo.com/2014/10/27/using-mysql-in-spring-boot-via-spring-data-jpa-and-hibernate/

\*/

package info.smartkit.eip.models;

import javax.transaction.Transactional;

import org.springframework.data.repository.CrudRepository;

@Transactional

public interface CompanyDao extends CrudRepository<Company, Long> {

/\*\*

\* This method is not implemented and its working code will be

\* auto-magically generated from its signature by Spring Data JPA.

\*

\* @param domain

\* the company domain.

\* @return the user having the passed domain or null if no company is found.

\*/

public Company findByDomain(String domain);

}

package info.smartkit.eip.models;

import java.util.List;

import org.springframework.data.repository.PagingAndSortingRepository;

import org.springframework.data.repository.query.Param;

import org.springframework.data.rest.core.annotation.RepositoryRestResource;

@RepositoryRestResource(collectionResourceRel="company",path="company")

public interface CompanyRepository extends PagingAndSortingRepository<Company, Long> {

List<Company> findById(@Param("id") long id);

}

/\*

\* Copyright 2015 the original author or authors.

\*

\* Licensed under the Apache License, Version 2.0 (the "License");

\* you may not use this file except in compliance with the License.

\* You may obtain a copy of the License at

\*

\* http://www.apache.org/licenses/LICENSE-2.0

\*

\* Unless required by applicable law or agreed to in writing, software

\* distributed under the License is distributed on an "AS IS" BASIS,

\* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.

\* See the License for the specific language governing permissions and

\* limitations under the License.

\*

\* All rights reserved.

\*/

package info.smartkit.eip.models;

import com.fasterxml.jackson.annotation.JsonIgnoreProperties;

import com.fasterxml.jackson.annotation.JsonProperty;

/\*\*

\* Contents of POJO file.

\*

\* @author yangboz

\*/

@JsonIgnoreProperties(ignoreUnknown = true)

public class DataBean

{

public DataBean(long expenses\_id, String expenses\_owner, String expenses\_name, String expenses\_date,

long expenses\_invoices\_num, String items\_date, String items\_vendors, String items\_invoices, String items\_notes,

String items\_participantIds, long items\_amount)

{

this.expenses\_id = expenses\_id;

this.expenses\_date = expenses\_date;

this.expenses\_name = expenses\_name;

this.expenses\_owner = expenses\_owner;

this.expenses\_invoices\_num = expenses\_invoices\_num;

this.items\_date = items\_date;

this.items\_invoices = items\_invoices;

this.items\_notes = items\_notes;

this.items\_participantIds = items\_participantIds;

this.items\_vendors = items\_vendors;

this.items\_amount = items\_amount;

}

public DataBean()

{

}

@JsonProperty("expenses\_id")

private long expenses\_id;

public long getExpenses\_id()

{

return expenses\_id;

}

public void setExpenses\_id(long expenses\_id)

{

this.expenses\_id = expenses\_id;

}

@JsonProperty("expenses\_owner")

private String expenses\_owner;

public String getExpenses\_owner()

{

return expenses\_owner;

}

public void setExpenses\_owner(String expenses\_owner)

{

this.expenses\_owner = expenses\_owner;

}

@JsonProperty("expenses\_name")

private String expenses\_name;

public String getExpenses\_name()

{

return expenses\_name;

}

public void setExpenses\_name(String expenses\_name)

{

this.expenses\_name = expenses\_name;

}

@JsonProperty("expenses\_date")

private String expenses\_date;

public String getExpenses\_date()

{

return expenses\_date;

}

public void setExpenses\_date(String expenses\_date)

{

this.expenses\_date = expenses\_date;

}

@JsonProperty("expenses\_invoices\_num")

private long expenses\_invoices\_num;

public long getExpenses\_invoices\_num()

{

return expenses\_invoices\_num;

}

public void setExpenses\_invoices\_num(long expenses\_invoices\_num)

{

this.expenses\_invoices\_num = expenses\_invoices\_num;

}

@JsonProperty("items\_date")

private String items\_date;

public String getItems\_date()

{

return items\_date;

}

public void setItems\_date(String items\_date)

{

this.items\_date = items\_date;

}

@JsonProperty("items\_vendors")

private String items\_vendors;

public String getItems\_vendors()

{

return items\_vendors;

}

public void setItems\_vendors(String items\_vendors)

{

this.items\_vendors = items\_vendors;

}

@JsonProperty("items\_notes")

private String items\_notes;

public String getItems\_notes()

{

return items\_notes;

}

public void setItems\_notes(String items\_notes)

{

this.items\_notes = items\_notes;

}

@JsonProperty("items\_invoices")

private String items\_invoices;

public String getItems\_invoices()

{

return items\_invoices;

}

public void setItems\_invoices(String items\_invoices)

{

this.items\_invoices = items\_invoices;

}

@JsonProperty("items\_participantIds")

private String items\_participantIds;

public String getItems\_participantIds()

{

return items\_participantIds;

}

public void setItems\_participantIds(String items\_participantIds)

{

this.items\_participantIds = items\_participantIds;

}

@JsonProperty("items\_amount")

private long items\_amount;

public long getItems\_amount()

{

return items\_amount;

}

public void setItems\_amount(long items\_amount)

{

this.items\_amount = items\_amount;

}

}

/\*

\* Copyright 2015 the original author or authors.

\*

\* Licensed under the Apache License, Version 2.0 (the "License");

\* you may not use this file except in compliance with the License.

\* You may obtain a copy of the License at

\*

\* http://www.apache.org/licenses/LICENSE-2.0

\*

\* Unless required by applicable law or agreed to in writing, software

\* distributed under the License is distributed on an "AS IS" BASIS,

\* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.

\* See the License for the specific language governing permissions and

\* limitations under the License.

\*

\* All rights reserved.

\*/

package info.smartkit.eip.models;

import java.util.ArrayList;

/\*\*

\* Contents of POJO list.

\*

\* @author yangboz

\*/

public class DataBeanList

{

public ArrayList<DataBean> getDataBeanList()

{

ArrayList<DataBean> dataBeanList = new ArrayList<DataBean>();

//

dataBeanList.add(new DataBean(112233, "2012-05-07", "员工01", "吃吃喝喝", 5, "2012-04-27", "商家0", "i1", "说明000",

"BB", 296));

dataBeanList

.add(new DataBean(22222, "2012-06-01", "USA", "AA", 5, "2012-05-27", "商家1", "i2", "说明001", "BB", 77));

dataBeanList.add(new DataBean(3333, "2012-07-08", "한국", "AA", 5, "2012-06-27", "商家2", "i3", "说明002", "BB", 88));

return dataBeanList;

}

}

package info.smartkit.eip.models;

import java.util.Date;

import javax.persistence.Entity;

import javax.persistence.EnumType;

import javax.persistence.Enumerated;

import javax.persistence.GeneratedValue;

import javax.persistence.GenerationType;

import javax.persistence.Id;

import javax.persistence.Table;

import javax.validation.constraints.NotNull;

import org.hibernate.annotations.SQLDelete;

import org.hibernate.annotations.Where;

@Entity

@Table(name = "expenses")

//@Embeddable

//Override the default Hibernation delete and set the deleted flag rather than deleting the record from the db.

@SQLDelete(sql="UPDATE expenses SET deleted = '1' WHERE id = ?")

//Filter added to retrieve only records that have not been soft deleted.

@Where(clause="deleted <> '1'")

public class Expense extends OwnerModelBase {

//

public enum ExpenseStatus {

Approved, Saved, Submitted, Rejected, Completed

}

// ==============

// PRIVATE FIELDS

// ==============

// An auto-generated id (unique for each user in the db)

@Id

@GeneratedValue(strategy = GenerationType.AUTO)

private long id;

public long getId() {

return id;

}

public void setId(long value) {

this.id = value;

}

// The expenses money amount(of expense item)

@NotNull

private double amount;

public double getAmount() {

return amount;

}

public void setAmount(double amount) {

this.amount = amount;

}

// The expense name

@NotNull

private String name;

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

// The item type

@Enumerated(EnumType.STRING)

private ExpenseStatus status;

public ExpenseStatus getStatus() {

return status;

}

public void setStatus(ExpenseStatus status) {

this.status = status;

}

// report manager id.

@NotNull

private String managerId;

public String getManagerId() {

return managerId;

}

public void setManagerId(String managerId) {

this.managerId = managerId;

}

// The expense involves,id strings

// @CollectionOfElements,@see:

// https://jazzy.id.au/2008/03/24/jpa\_2\_0\_new\_features\_part\_1.html

private String participantIds;// sort of participant IDs:"1,2,3,4"

public String getParticipantIds() {

return participantIds;

}

public void setParticipantIds(String participantIds) {

this.participantIds = participantIds;

}

//

@NotNull

private String itemIds;// sort of expense item IDs:"1,2,3,4"

public String getItemIds() {

return itemIds;

}

public void setItemIds(String itemIds) {

this.itemIds = itemIds;

}

//Activiti process instance id

private long pid;

public long getPid() {

return pid;

}

public void setPid(long pid) {

this.pid = pid;

}

//Soft-delete

//@see: http://featurenotbug.com/2009/07/soft-deletes-using-hibernate-annotations/

private char deleted='0';

public char getDeleted() {

return deleted;

}

public void setDeleted(char deleted) {

this.deleted = deleted;

}

// ==============

// PUBLIC METHODS

// ==============

public Expense() {

}

public Expense(long id) {

this.id = id;

}

public Expense(double amount, String name, ExpenseStatus status, Date date,

String owner, String managerId, String participantIds,

String itemIds,long pid) {

this.amount = amount;

this.name = name;

this.status = status;

this.date = date;

this.owner = owner;

this.managerId = managerId;

this.participantIds = participantIds;

this.itemIds = itemIds;

this.pid = pid;

}

}

package info.smartkit.eip.models;

import javax.transaction.Transactional;

import org.springframework.data.repository.CrudRepository;

import org.springframework.data.repository.query.Param;

@Transactional

//public interface ExpenseDao extends CrudRepository<Expense, Long> {

public interface ExpenseDao extends CrudRepository<Expense, String> {

/\*\*

\* This method is not implemented and its working code will be

\* auto-magically generated from its signature by Spring Data JPA.

\*

\* @param owner

\* the expense owner.

\* @return the expense having the passed owner or null if no expense is

\* found.

\*/

public Expense findById(long id);

//@see: http://ufasoli.blogspot.ca/2014/02/spring-boot-and-spring-data-jpa\_6.html

public Iterable<Expense> findExpensesByOwner(@Param("owner") String owner);

}

package info.smartkit.eip.models;

import java.util.List;

import org.springframework.data.repository.PagingAndSortingRepository;

import org.springframework.data.repository.query.Param;

import org.springframework.data.rest.core.annotation.RepositoryRestResource;

@RepositoryRestResource(collectionResourceRel="expenses",path="expenses")

public interface ExpenseRepository extends PagingAndSortingRepository<Expense, Long> {

List<Expense> findById(@Param("id") long id);

}package info.smartkit.eip.models;

import java.util.Map;

import javax.persistence.ElementCollection;

import javax.persistence.Entity;

import javax.persistence.GeneratedValue;

import javax.persistence.GenerationType;

import javax.persistence.Id;

import javax.persistence.Table;

import javax.validation.constraints.NotNull;

@Entity

@Table(name = "invoices")

//@Embeddable

public class Invoice extends OwnerModelBase {

// ==============

// PRIVATE FIELDS

// ==============

// An auto-generated id (unique for each user in the db)

@Id

@GeneratedValue(strategy = GenerationType.AUTO)

private long id;

public long getId() {

return id;

}

public void setId(long value) {

this.id = value;

}

// The invoice name(default picture file name)

@NotNull

private String name;

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

// The invoice picture

// TODO:big/middle/small size generation.

@NotNull

@ElementCollection

Map<String,String> picture;

// private String picture;

//

// public String getPicture() {

// return picture;

// }

//

// public void setPicture(String picture) {

// this.picture = picture;

// }

public Map<String, String> getPicture() {

return picture;

}

public void setPicture(Map<String, String> picture) {

this.picture = picture;

}

// ==============

// PUBLIC METHODS

// ==============

public Invoice() {

}

public Invoice(long id) {

this.id = id;

}

public Invoice(String name, String owner,Map<String,String> picture) {

this.name = name;

this.owner = owner;

this.picture = picture;

}

}

package info.smartkit.eip.models;

import javax.transaction.Transactional;

import org.springframework.data.repository.CrudRepository;

@Transactional

public interface InvoiceDao extends CrudRepository<Invoice, Long> {

/\*\*

\* This method is not implemented and its working code will be

\* auto-magically generated from its signature by Spring Data JPA.

\*

\* @param id

\* the item id.

\* @return the invoice having the passed id or null if no invoice is found.

\*/

public Invoice findById(long id);

}

package info.smartkit.eip.models;

import java.util.Date;

import javax.persistence.Entity;

import javax.persistence.EnumType;

import javax.persistence.Enumerated;

import javax.persistence.GeneratedValue;

import javax.persistence.GenerationType;

import javax.persistence.Id;

import javax.persistence.Table;

import javax.validation.constraints.NotNull;

@Entity

@Table(name = "items")

public class Item extends OwnerModelBase {

// ApproveAhead,CostComsumed

public enum ItemType {

ApproveAhead, CostConsumed

}

// ==============

// PRIVATE FIELDS

// ==============

// An auto-generated id (unique for each user in the db)

@Id

@GeneratedValue(strategy = GenerationType.AUTO)

private long id;

public long getId() {

return id;

}

public void setId(long value) {

this.id = value;

}

// The item money amount

@NotNull

private double amount;

public double getAmount() {

return amount;

}

public void setAmount(double amount) {

this.amount = amount;

}

// The item name

@NotNull

private String name;

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

// The item type

@Enumerated(EnumType.STRING)

private ItemType type;

public ItemType getType() {

return type;

}

public void setType(ItemType type) {

this.type = type;

}

// The item invoices,id strings

// @CollectionOfElements,@see:

// https://jazzy.id.au/2008/03/24/jpa\_2\_0\_new\_features\_part\_1.html

private String invoices = "1";// sort of invoice ids:"1,2,3,4"

// @ElementCollection

// @CollectionTable(name = "item\_invoices", joinColumns = @JoinColumn(name = "itemId"))

// private Set<Invoice> invoices = new HashSet<Invoice>();

//

// public Set<Invoice> getInvoices() {

// return invoices;

// }

//

// public void setInvoices(Set<Invoice> invoices) {

// this.invoices = invoices;

// }

public String getInvoices() {

return invoices;

}

public void setInvoices(String invoices) {

this.invoices = invoices;

}

// The item vendors,id strings

private String vendors = "1";// sort of vendor ids:"1,2,3,4"

// @see:

// http://www.thejavageek.com/2014/01/31/jpa-elementcollection-annotation/

// @ElementCollection

// @CollectionTable(name = "item\_vendors", joinColumns = @JoinColumn(name = "itemId"))

// private Set<Vendor> vendors = new HashSet<Vendor>();

//

// public Set<Vendor> getVendors() {

// return vendors;

// }

//

// public void setVendors(Set<Vendor> vendors) {

// this.vendors = vendors;

// }

public String getVendors() {

return vendors;

}

public void setVendors(String vendors) {

this.vendors = vendors;

}

// The item category,id strings.

private String category = "1";//one category id:"1"

// @ElementCollection

// @CollectionTable(name = "item\_categories", joinColumns = @JoinColumn(name = "itemId"))

// private Set<Category> categories = new HashSet<Category>();

//

//

// public Set<Category> getCategories() {

// return categories;

// }

//

// public void setCategories(Set<Category> categories) {

// this.categories = categories;

// }

public String getCategory() {

return category;

}

public void setCategory(String category) {

this.category = category;

}

//Vendor related place.

private String place = "";

public String getPlace() {

return place;

}

public void setPlace(String place) {

this.place = place;

}

// The expense involves,id strings

// @CollectionOfElements,@see:

// https://jazzy.id.au/2008/03/24/jpa\_2\_0\_new\_features\_part\_1.html

private String participantIds;// sort of participant IDs:"1,2,3,4"

public String getParticipantIds() {

return participantIds;

}

public void setParticipantIds(String participantIds) {

this.participantIds = participantIds;

}

private String notes;

public String getNotes() {

return notes;

}

public void setNotes(String notes) {

this.notes = notes;

}

/\*

\* Indicate whether or not used.

\*/

private Boolean used=false;

public Boolean getUsed() {

return used;

}

public void setUsed(Boolean used) {

this.used = used;

}

// ==============

// PUBLIC METHODS

// ==============

public Item() {

}

public Item(long id) {

this.id = id;

}

// public Item(double amount, String name, ItemType type, Date date,

// Set<Invoice> invoices, Set<Vendor> vendors, String owner,

// Set<Category> category) {

public Item(double amount, String name, ItemType type, Date date,

String invoices, String vendors, String owner, String place,

String category, String participantIds,String notes) {

this.amount = amount;

this.name = name;

this.type = type;

this.date = date;

this.invoices = invoices;

this.vendors = vendors;

this.owner = owner;

this.category = category;

this.participantIds = participantIds;

this.notes = notes;

this.place = place;

}

}

package info.smartkit.eip.models;

import javax.transaction.Transactional;

import org.springframework.data.repository.CrudRepository;

import org.springframework.data.repository.query.Param;

@Transactional

public interface ItemDao extends CrudRepository<Item, Long> {

/\*\*

\* This method is not implemented and its working code will be

\* auto-magically generated from its signature by Spring Data JPA.

\*

\* @param id

\* the item id.

\* @return the item having the passed id or null if no item is found.

\*/

public Item findById(long id);

/\*\*

\* This method is not implemented and its working code will be

\* auto-magically generated from its signature by Spring Data JPA.

\*

\* @param name

\* the item owner'name.

\* @return the item having the passed owner'name or null if no item is found.

\*/

//@see: http://ufasoli.blogspot.ca/2014/02/spring-boot-and-spring-data-jpa\_6.html

public Iterable<Item> findItemsByOwner(@Param("owner") String owner);

}

package info.smartkit.eip.models;

import java.util.List;

import org.springframework.data.repository.PagingAndSortingRepository;

import org.springframework.data.repository.query.Param;

import org.springframework.data.rest.core.annotation.RepositoryRestResource;

@RepositoryRestResource(collectionResourceRel="items",path="items")

public interface ItemRepository extends PagingAndSortingRepository<Item, Long> {

List<Item> findById(@Param("id") long id);

// List<Item> findByOwner(@Param("owner") String owner);

}package info.smartkit.eip.models;

import java.util.Calendar;

import java.util.Date;

import javax.persistence.Column;

import javax.persistence.MappedSuperclass;

import javax.persistence.PrePersist;

import javax.persistence.PreUpdate;

import javax.persistence.Temporal;

import javax.persistence.TemporalType;

@MappedSuperclass

public class ModelBase {

// @Version

// @Temporal(TemporalType.DATE)

// @Column(name = "created", nullable = false)

// @Type(type="org.joda.time.contrib.hibernate.PersistentDateTime")

@Column(name = "CREATED", nullable = false, insertable = true, updatable = true, length = 19, precision = 0)

@Temporal(TemporalType.TIMESTAMP)

private Date created;

@Temporal(TemporalType.TIMESTAMP)

@Column(name = "updated", nullable = false)

private Date updated;

@PrePersist

protected void onCreate() {

updated = created = new Date();

}

@PreUpdate

protected void onUpdate() {

updated = new Date();

}

public Date getCreated() {

return created;

}

public Date getUpdated() {

return updated;

}

// The item date

// @NotNull

// private Date date = new Date();

protected Date date;

public Date getDate() {

return date;

}

public void setDate(Date date) {

this.date = date;

}

}

package info.smartkit.eip.models;

import javax.persistence.MappedSuperclass;

import javax.validation.constraints.NotNull;

@MappedSuperclass

public class OwnerModelBase extends ModelBase{

// The base owner name or id.

@NotNull

protected String owner;

public String getOwner() {

return owner;

}

public void setOwner(String owner) {

this.owner = owner;

}

}

package info.smartkit.eip.models;

import javax.naming.NamingException;

import javax.naming.directory.Attributes;

import info.smartkit.eip.ldap.plain.domain.Person;

import org.springframework.ldap.core.AttributesMapper;

public class PersonAttributesMapper implements AttributesMapper<Person>

{

@Override

public Person mapFromAttributes(Attributes attrs) throws NamingException

{

Person person = new Person();

person.setCn((String) attrs.get("cn").get());

person.setSn((String) attrs.get("sn").get());

person.setDescription((String) attrs.get("description").get());

person.setUid((String) attrs.get("uid").get());

person.setWxToken((String) attrs.get("postalAddress").get());

person.setCode((String) attrs.get("postalCode").get());

return person;

}

}

/\*

\* Copyright 2015 the original author or authors.

\*

\* Licensed under the Apache License, Version 2.0 (the "License");

\* you may not use this file except in compliance with the License.

\* You may obtain a copy of the License at

\*

\* http://www.apache.org/licenses/LICENSE-2.0

\*

\* Unless required by applicable law or agreed to in writing, software

\* distributed under the License is distributed on an "AS IS" BASIS,

\* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.

\* See the License for the specific language governing permissions and

\* limitations under the License.

\*

\* All rights reserved.

\*/

package info.smartkit.eip.models;

import org.springframework.ldap.odm.annotations.Entry;

/\*\*

\* Simple person object for OdmPerson posting to LDAP server.

\*

\* @author yangboz

\*/

@Entry(objectClasses = {"person", "top"}, base = "")

public class SimplePerson

{

private String mobile;

public String getMobile()

{

return mobile;

}

public void setMobile(String mobile)

{

this.mobile = mobile;

}

private String email;

public String getEmail()

{

return email;

}

public void setEmail(String email)

{

this.email = email;

}

private String password;

public String getPassword()

{

return password;

}

public void setPassword(String password)

{

this.password = password;

}

private String username;

public String getUsername()

{

return username;

}

public void setUsername(String username)

{

this.username = username;

}

private String wxToken;

public String getWxToken()

{

return wxToken;

}

public void setWxToken(String wxToken)

{

this.wxToken = wxToken;

}

private String code;

public String getCode()

{

return code;

}

public void setCode(String code)

{

this.code = code;

}

}

package info.smartkit.eip.models;

import javax.persistence.Entity;

import javax.persistence.GeneratedValue;

import javax.persistence.GenerationType;

import javax.persistence.Id;

import javax.persistence.Table;

import javax.validation.constraints.NotNull;

@Entity

@Table(name = "tags")

public class Tag extends ModelBase {

//public class Tag {

// ==============

// PRIVATE FIELDS

// ==============

// An auto-generated id (unique for each user in the db)

@Id

@GeneratedValue(strategy = GenerationType.AUTO)

private long id;

public long getId() {

return id;

}

public void setId(long value) {

this.id = value;

}

// The tag icon

// @NotNull

private String icon;

public String getIcon() {

return icon;

}

public void setIcon(String icon) {

this.icon = icon;

}

// The tag name

@NotNull

private String name;

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

// The tag name

// @NotNull

private String label;

public String getLabel() {

return label;

}

public void setLabel(String label) {

this.label = label;

}

// Foreign key(item\_id)

// @CollectionTable(name = "tag\_items", joinColumns = @JoinColumn(name =

// "idtag"))

@NotNull

private long itemId;

public long getItemId() {

return itemId;

}

public void setItemId(long itemId) {

this.itemId = itemId;

}

// ==============

// PUBLIC METHODS

// ==============

public Tag() {

}

public Tag(long id) {

this.id = id;

}

public Tag(String icon, String name, long itemId) {

this.icon = icon;

this.name = name;

this.itemId = itemId;

}

}

package info.smartkit.eip.models;

import javax.transaction.Transactional;

import org.springframework.data.repository.CrudRepository;

@Transactional

public interface TagDao extends CrudRepository<Tag, Long> {

/\*\*

\* This method is not implemented and its working code will be

\* auto-magically generated from its signature by Spring Data JPA.

\*

\* @param id

\* the tag id.

\* @return the tag having the passed id or null if no tag is found.

\*/

public Tag findById(long id);

}

package info.smartkit.eip.models;

import java.util.List;

import org.springframework.data.repository.PagingAndSortingRepository;

import org.springframework.data.repository.query.Param;

import org.springframework.data.rest.core.annotation.RepositoryRestResource;

@RepositoryRestResource(collectionResourceRel = "tags", path = "tags")

public interface TagRepository extends PagingAndSortingRepository<Tag, Long> {

List<Tag> findById(@Param("id") long id);

}

/\*

\* Copyright 2014 the original author or authors.

\*

\* Licensed under the Apache License, Version 2.0 (the "License");

\* you may not use this file except in compliance with the License.

\* You may obtain a copy of the License at

\*

\* http://www.apache.org/licenses/LICENSE-2.0

\*

\* Unless required by applicable law or agreed to in writing, software

\* distributed under the License is distributed on an "AS IS" BASIS,

\* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.

\* See the License for the specific language governing permissions and

\* limitations under the License.

\*/

package info.smartkit.eip.models;

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.hibernate.validator.constraints.NotEmpty;

@Entity

@Table(name = "users")

public class User

{

@Id

@GeneratedValue(strategy = GenerationType.AUTO)

private Integer id;

@NotEmpty

private String name;

@NotEmpty

@Column(unique = true, nullable = false)

private String login;

@NotEmpty

private String password;

public User()

{

}

public User(User user)

{

super();

this.id = user.getId();

this.name = user.getName();

this.login = user.getLogin();

this.password = user.getPassword();

}

public Integer getId()

{

return id;

}

public void setId(Integer id)

{

this.id = id;

}

public String getName()

{

return name;

}

public void setName(String name)

{

this.name = name;

}

public String getLogin()

{

return login;

}

public void setLogin(String login)

{

this.login = login;

}

public String getPassword()

{

return password;

}

public void setPassword(String password)

{

this.password = password;

}

}

/\*

\* Copyright 2014 the original author or authors.

\*

\* Licensed under the Apache License, Version 2.0 (the "License");

\* you may not use this file except in compliance with the License.

\* You may obtain a copy of the License at

\*

\* http://www.apache.org/licenses/LICENSE-2.0

\*

\* Unless required by applicable law or agreed to in writing, software

\* distributed under the License is distributed on an "AS IS" BASIS,

\* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.

\* See the License for the specific language governing permissions and

\* limitations under the License.

\*/

package info.smartkit.eip.models;

import org.springframework.data.repository.CrudRepository;

public interface UserRepository extends CrudRepository<User, Long> {

User findByLogin(String login);

}

package info.smartkit.eip.models;

import javax.persistence.Entity;

import javax.persistence.GeneratedValue;

import javax.persistence.GenerationType;

import javax.persistence.Id;

import javax.persistence.Table;

import javax.validation.constraints.NotNull;

@Entity

@Table(name = "vendors")

//@Embeddable

public class Vendor extends ModelBase {

// ==============

// PRIVATE FIELDS

// ==============

// An auto-generated id (unique for each user in the db)

@Id

@GeneratedValue(strategy = GenerationType.AUTO)

private long id;

public long getId() {

return id;

}

public void setId(long value) {

this.id = value;

}

// The item name

@NotNull

private String name;

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

// ==============

// PUBLIC METHODS

// ==============

public Vendor() {

}

public Vendor(long id) {

this.id = id;

}

public Vendor(String name) {

this.name = name;

}

}

package info.smartkit.eip.models;

import javax.transaction.Transactional;

import org.springframework.data.repository.CrudRepository;

@Transactional

public interface VendorDao extends CrudRepository<Vendor, Long> {

/\*\*

\* This method is not implemented and its working code will be

\* auto-magically generated from its signature by Spring Data JPA.

\*

\* @param id

\* the vendor id.

\* @return the vendor having the passed id or null if no vendor is found.

\*/

public Vendor findById(long id);

}

package info.smartkit.eip.models;

import java.util.List;

import org.springframework.data.repository.PagingAndSortingRepository;

import org.springframework.data.repository.query.Param;

import org.springframework.data.rest.core.annotation.RepositoryRestResource;

@RepositoryRestResource(collectionResourceRel="vendors",path="vendors")

public interface VendorRepository extends PagingAndSortingRepository<Vendor, Long> {

List<Item> findById(@Param("id") long id);

}/\*

\* DynamicJasper: A library for creating reports dynamically by specifying

\* columns, groups, styles, etc. at runtime. It also saves a lot of development

\* time in many cases! (http://sourceforge.net/projects/dynamicjasper)

\*

\* Copyright (C) 2008 FDV Solutions (http://www.fdvsolutions.com)

\*

\* This library is free software; you can redistribute it and/or

\* modify it under the terms of the GNU Lesser General Public

\*

\* License as published by the Free Software Foundation; either

\*

\* version 2.1 of the License, or (at your option) any later version.

\*

\* This library is distributed in the hope that it will be useful,

\* but WITHOUT ANY WARRANTY; without even the implied warranty of

\*

\* MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU

\*

\* Lesser General Public License for more details.

\*

\* You should have received a copy of the GNU Lesser General Public

\* License along with this library; if not, write to the Free Software

\*

\* Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA

\*

\*

\*/

package info.smartkit.eip.reports.domain;

public class DummyLevel3 {

private String name;

private Long number;

public DummyLevel3(String name, Long number) {

super();

this.name = name;

this.number = number;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public Long getNumber() {

return number;

}

public void setNumber(Long number) {

this.number = number;

}

}

/\*

\* DynamicJasper: A library for creating reports dynamically by specifying

\* columns, groups, styles, etc. at runtime. It also saves a lot of development

\* time in many cases! (http://sourceforge.net/projects/dynamicjasper)

\*

\* Copyright (C) 2008 FDV Solutions (http://www.fdvsolutions.com)

\*

\* This library is free software; you can redistribute it and/or

\* modify it under the terms of the GNU Lesser General Public

\*

\* License as published by the Free Software Foundation; either

\*

\* version 2.1 of the License, or (at your option) any later version.

\*

\* This library is distributed in the hope that it will be useful,

\* but WITHOUT ANY WARRANTY; without even the implied warranty of

\*

\* MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU

\*

\* Lesser General Public License for more details.

\*

\* You should have received a copy of the GNU Lesser General Public

\* License along with this library; if not, write to the Free Software

\*

\* Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA

\*

\*

\*/

package info.smartkit.eip.reports.domain;

import java.io.InputStream;

import java.text.ParseException;

import java.text.SimpleDateFormat;

import java.util.ArrayList;

import java.util.Collections;

import java.util.Date;

import java.util.List;

import java.util.Random;

public class Product {

private Long id;

private String productLine;

private String item;

private String state;

private String branch;

private Long quantity;

private Float amount;

private Code code = new Code();

private Boolean showAsPrices;

public static List statistics\_ = new ArrayList();

private static SimpleDateFormat formatter = new SimpleDateFormat("dd/MM/yyyy");

static String[] images = {"confused.gif","cool.gif","happy.gif","puaj.gif","ungry.gif","what.gif"};

static int counter = 0;

static {

try {

statistics\_.add(new Statistic(formatter.parse("01/01/2003"),"West",14.3f,50.4f,43.1f));

statistics\_.add(new Statistic(formatter.parse("01/01/2004"),"West",40.0f,49.4f,44.5f));

statistics\_.add(new Statistic(formatter.parse("01/01/2005"),"North",33.3f,63.4f,45f));

statistics\_.add(new Statistic(formatter.parse("01/01/2006"),"East",91.1f,34.4f,46f));

statistics\_.add(new Statistic(formatter.parse("01/01/2007"),"South",99.3f,52.4f,47f));

} catch (ParseException e) {

e.printStackTrace();

}

}

public List getStatistics(){

return statistics\_;

}

public List getEmptyStatistics(){

return Collections.EMPTY\_LIST;

}

public Product(){}

public Product(Long id, String productLine, String item, String state, String branch, Long quantity, Float amount) {

this.id = id;

this.productLine = productLine;

this.item = item;

this.state = state;

this.branch = branch;

this.quantity = quantity;

this.amount = amount;

}

public Product(Long id, String productLine, String item, String state, String branch, Long quantity, Float amount, boolean showAsPrices) {

this(id, productLine, item, state, branch, quantity, amount);

this.showAsPrices = showAsPrices;

}

public String getBranch() {

return branch;

}

public void setBranch(String branch) {

this.branch = branch;

}

public Float getAmount() {

return amount;

// return new Float(quantity.floatValue());

}

public void setAmount(Float amount) {

this.amount = amount;

}

public Long getId() {

return id;

}

public void setId(Long id) {

this.id = id;

}

public String getItem() {

return item;

}

public void setItem(String item) {

this.item = item;

}

public String getProductLine() {

return productLine;

}

public void setProductLine(String productLine) {

this.productLine = productLine;

}

public Long getQuantity() {

return quantity;

}

public void setQuantity(Long quantity) {

this.quantity = quantity;

}

public String getState() {

return state;

}

public void setState(String state) {

this.state = state;

}

public Date getDate() {

return new Date();

}

public Boolean getIsAvailable(){

return Boolean.valueOf(random.nextBoolean());

}

public Boolean getShowAsPrices() {

return showAsPrices;

}

public void setShowAsPrices(Boolean showAsPrices) {

this.showAsPrices = showAsPrices;

}

static Random random = new Random();

public InputStream getImage() {

InputStream ret = this.getClass().getClassLoader().getResourceAsStream("images/emoticons/" + images[counter++]);

if (counter >= images.length)

counter = 0;

return ret;

}

public Code getCode() {

return code;

}

public class Code {

public String getCode() {

return "001-123ABC-HRC";

}

}

}

/\*

\* DynamicJasper: A library for creating reports dynamically by specifying

\* columns, groups, styles, etc. at runtime. It also saves a lot of development

\* time in many cases! (http://sourceforge.net/projects/dynamicjasper)

\*

\* Copyright (C) 2008 FDV Solutions (http://www.fdvsolutions.com)

\*

\* This library is free software; you can redistribute it and/or

\* modify it under the terms of the GNU Lesser General Public

\*

\* License as published by the Free Software Foundation; either

\*

\* version 2.1 of the License, or (at your option) any later version.

\*

\* This library is distributed in the hope that it will be useful,

\* but WITHOUT ANY WARRANTY; without even the implied warranty of

\*

\* MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU

\*

\* Lesser General Public License for more details.

\*

\* You should have received a copy of the GNU Lesser General Public

\* License along with this library; if not, write to the Free Software

\*

\* Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA

\*

\*

\*/

package info.smartkit.eip.reports.domain;

import java.util.ArrayList;

import java.util.Date;

import java.util.List;

public class Statistic {

private Date date;

private String name;

private float percentage;

private float average;

private float amount;

public static List dummy3 = new ArrayList();

static {

dummy3.add(new DummyLevel3("name1",new Long(1)));

dummy3.add(new DummyLevel3("name2",new Long(2)));

dummy3.add(new DummyLevel3("name3",new Long(3)));

dummy3.add(new DummyLevel3("name3",new Long(4)));

}

public Statistic() {

super();

}

public Statistic(Date date, String name, float percentage, float average, float amount) {

super();

this.date = date;

this.name = name;

this.percentage = percentage;

this.average = average;

this.amount = amount;

}

public Date getDate() {

return date;

}

public void setDate(Date date) {

this.date = date;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public float getPercentage() {

return percentage;

}

public void setPercentage(float percentage) {

this.percentage = percentage;

}

public float getAverage() {

return average;

}

public void setAverage(float average) {

this.average = average;

}

public float getAmount() {

return amount;

}

public void setAmount(float amount) {

this.amount = amount;

}

public List getDummy3() {

return dummy3;

}

}

/\*

\* DynamicJasper: A library for creating reports dynamically by specifying

\* columns, groups, styles, etc. at runtime. It also saves a lot of development

\* time in many cases! (http://sourceforge.net/projects/dynamicjasper)

\*

\* Copyright (C) 2008 FDV Solutions (http://www.fdvsolutions.com)

\*

\* This library is free software; you can redistribute it and/or

\* modify it under the terms of the GNU Lesser General Public

\*

\* License as published by the Free Software Foundation; either

\*

\* version 2.1 of the License, or (at your option) any later version.

\*

\* This library is distributed in the hope that it will be useful,

\* but WITHOUT ANY WARRANTY; without even the implied warranty of

\*

\* MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU

\*

\* Lesser General Public License for more details.

\*

\* You should have received a copy of the GNU Lesser General Public

\* License along with this library; if not, write to the Free Software

\*

\* Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA

\*

\*

\*/

package info.smartkit.eip.reports.utils;

import java.lang.reflect.InvocationTargetException;

import java.util.Comparator;

import java.util.List;

import org.apache.commons.beanutils.PropertyUtils;

import org.apache.commons.logging.Log;

import org.apache.commons.logging.LogFactory;

/\*\*

\* @author Alejandro Gomez

\* Date: Feb 26, 2007

\* Time: 10:27:32 AM

\*/

public class MultiPropertyComparator implements Comparator {

private static final Log LOGGER = LogFactory.getLog(MultiPropertyComparator.class);

private List info;

/\*\*

\*

\* @param \_info List<SortInfo>

\*/

public MultiPropertyComparator(final List \_info) {

info = \_info;

}

public int compare(final Object \_o1, final Object \_o2) {

int result = 0;

for (int i = 0; result == 0 && i < info.size(); i++) {

final SortInfo sortInfo = (SortInfo)info.get(i);

try {

final String propertyName = sortInfo.getPropertyName();

final Comparable value1 = getValue(\_o1, propertyName);

final Comparable value2 = getValue(\_o2, propertyName);

result = compare(value1, value2) \* (sortInfo.isAscending() ? 1 : -1);

} catch (IllegalAccessException ex) {

LOGGER.warn(ex);

} catch (InvocationTargetException ex) {

LOGGER.warn(ex);

} catch (NoSuchMethodException ex) {

LOGGER.warn(ex);

}

}

return result;

}

private static Comparable getValue(final Object \_object, final String \_field) throws NoSuchMethodException, IllegalAccessException, InvocationTargetException {

final Object value = PropertyUtils.getProperty(\_object, \_field);

if (value == null)

return null;

if (!(value instanceof Comparable))

throw new RuntimeException("Objects are not Comparable, class " + value.getClass().getName());

return (Comparable) value;

}

private static int compare(final Comparable \_value1, final Comparable \_value2) {

if (\_value1 == null) {

if (\_value2 == null) {

return 0;

} else {

return -1;

}

} else if (\_value2 == null) {

return 1;

} else {

return \_value1.compareTo(\_value2);

}

}

}

/\*

\* DynamicJasper: A library for creating reports dynamically by specifying

\* columns, groups, styles, etc. at runtime. It also saves a lot of development

\* time in many cases! (http://sourceforge.net/projects/dynamicjasper)

\*

\* Copyright (C) 2008 FDV Solutions (http://www.fdvsolutions.com)

\*

\* This library is free software; you can redistribute it and/or

\* modify it under the terms of the GNU Lesser General Public

\*

\* License as published by the Free Software Foundation; either

\*

\* version 2.1 of the License, or (at your option) any later version.

\*

\* This library is distributed in the hope that it will be useful,

\* but WITHOUT ANY WARRANTY; without even the implied warranty of

\*

\* MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU

\*

\* Lesser General Public License for more details.

\*

\* You should have received a copy of the GNU Lesser General Public

\* License along with this library; if not, write to the Free Software

\*

\* Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA

\*

\*

\*/

package info.smartkit.eip.reports.utils;

/\*\*

\* @author Alejandro Gomez

\* Date: Feb 26, 2007

\* Time: 10:27:04 AM

\*/

public class SortInfo {

private String field;

private boolean ascending;

public SortInfo(final String \_field, final boolean \_ascending) {

field = \_field;

ascending = \_ascending;

}

public String getPropertyName() {

return field;

}

public boolean isAscending() {

return ascending;

}

}

/\*

\* Copyright 2014 the original author or authors.

\*

\* Licensed under the Apache License, Version 2.0 (the "License");

\* you may not use this file except in compliance with the License.

\* You may obtain a copy of the License at

\*

\* http://www.apache.org/licenses/LICENSE-2.0

\*

\* Unless required by applicable law or agreed to in writing, software

\* distributed under the License is distributed on an "AS IS" BASIS,

\* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.

\* See the License for the specific language governing permissions and

\* limitations under the License.

\*/

package info.smartkit.eip.services;

import java.util.Collection;

import info.smartkit.eip.models.User;

import info.smartkit.eip.models.UserRepository;

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

import org.springframework.security.core.GrantedAuthority;

import org.springframework.security.core.authority.AuthorityUtils;

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

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

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

import org.springframework.stereotype.Service;

@Service

public class UserDetailsServiceImpl implements UserDetailsService {

private final UserRepository userRepository;

@Autowired

public UserDetailsServiceImpl(UserRepository userRepository) {

this.userRepository = userRepository;

}

@Override

public UserDetails loadUserByUsername(String username)

throws UsernameNotFoundException {

User user = userRepository.findByLogin(username);

if (user == null) {

throw new UsernameNotFoundException(String.format(

"User %s does not exist!", username));

}

return new UserRepositoryUserDetails(user);

}

private final static class UserRepositoryUserDetails extends User implements

UserDetails {

private static final long serialVersionUID = 1L;

private UserRepositoryUserDetails(User user) {

super(user);

}

@Override

public Collection<? extends GrantedAuthority> getAuthorities() {

return AuthorityUtils.createAuthorityList("ROLE\_USER");

}

@Override

public String getUsername() {

return getLogin();

}

@Override

public boolean isAccountNonExpired() {

return true;

}

@Override

public boolean isAccountNonLocked() {

return true;

}

@Override

public boolean isCredentialsNonExpired() {

return true;

}

@Override

public boolean isEnabled() {

return true;

}

}

}

package info.smartkit.eip.settings;

import org.springframework.boot.context.properties.ConfigurationProperties;

import org.springframework.boot.context.properties.EnableConfigurationProperties;

import org.springframework.context.annotation.Configuration;

//@see https://docs.oracle.com/javase/tutorial/jndi/ldap/jndi.html

@Configuration

@EnableConfigurationProperties

@ConfigurationProperties(prefix="db")

public class DBSetting {

private String driver;

public String getDriver() {

return driver;

}

public void setDriver(String driver) {

this.driver = driver;

}

private String url;

public String getUrl() {

return url;

}

public void setUrl(String url) {

this.url = url;

}

private String username;

public String getUsername() {

return username;

}

public void setUsername(String username) {

this.username = username;

}

private String password;

public String getPassword() {

return password;

}

public void setPassword(String password) {

this.password = password;

}

}

package info.smartkit.eip.settings;

import org.springframework.boot.context.properties.ConfigurationProperties;

import org.springframework.boot.context.properties.EnableConfigurationProperties;

import org.springframework.context.annotation.Configuration;

@Configuration

@EnableConfigurationProperties

@ConfigurationProperties(prefix = "folder")

public class FolderSetting

{

private String uploads;

public String getUploads()

{

return uploads;

}

public void setUploads(String uploads)

{

this.uploads = uploads;

}

private String reports;

public String getReports()

{

return reports;

}

public void setReports(String reports)

{

this.reports = reports;

}

}

package info.smartkit.eip.settings;

import org.apache.logging.log4j.LogManager;

import org.apache.logging.log4j.Logger;

import org.springframework.boot.context.properties.ConfigurationProperties;

import org.springframework.boot.context.properties.EnableConfigurationProperties;

import org.springframework.context.annotation.Configuration;

import info.smartkit.eip.activemq.ActivemqVariables;

@Configuration

@EnableConfigurationProperties

@ConfigurationProperties(prefix = "jms")

public class JMSSetting

{

private static Logger LOG = LogManager.getLogger(JMSSetting.class);

private String brokerUrl;

public String getBrokerUrl()

{

return brokerUrl;

}

public void setBrokerUrl(String brokerUrl)

{

this.brokerUrl = brokerUrl;

//

ActivemqVariables.brokerUrl = brokerUrl;

LOG.info("brokerUrl:" + ActivemqVariables.brokerUrl);

}

}

package info.smartkit.eip.settings;

import org.apache.logging.log4j.LogManager;

import org.apache.logging.log4j.Logger;

import org.springframework.boot.context.properties.ConfigurationProperties;

import org.springframework.boot.context.properties.EnableConfigurationProperties;

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

import org.springframework.ldap.core.ContextSource;

import org.springframework.ldap.core.LdapTemplate;

import org.springframework.ldap.core.support.LdapContextSource;

import info.smartkit.eip.ldap.odm.dao.PersonDaoImpl;

//@see https://docs.oracle.com/javase/tutorial/jndi/ldap/jndi.html

@Configuration

@EnableConfigurationProperties

@ConfigurationProperties(prefix = "ldap")

public class LDAPSetting

{

private static Logger LOG = LogManager.getLogger(LDAPSetting.class);

private String url;

public String getUrl()

{

return url;

}

public void setUrl(String url)

{

this.url = url;

}

private String userOn;

public String getUserOn()

{

return userOn;

}

public void setUserOn(String userOn)

{

this.userOn = userOn;

}

private String password;

public String getPassword()

{

return password;

}

public void setPassword(String password)

{

this.password = password;

}

private String base;

public String getBase()

{

return base;

}

public void setBase(String base)

{

this.base = base;

}

//

@Override

public String toString()

{

return "url:" + this.url + ",userOn:" + this.userOn + ",base:" + this.base + ",password:" + this.password;

}

// @see: http://stackoverflow.com/questions/26004062/ldap-query-configuration-using-spring-boot

@Bean

@ConfigurationProperties(prefix = "ldap")

public LdapContextSource contextSource()

{

LdapContextSource contextSource = new LdapContextSource();

contextSource.setUrl(this.getUrl());

contextSource.setUserDn(this.getUserOn());

contextSource.setPassword(this.getPassword());

try {

contextSource.afterPropertiesSet();

} catch (Exception e) {

e.printStackTrace();

}

LOG.debug("LdapContextSource:" + this.toString());

// LdapContextSource contextSource = new LdapContextSource();

return contextSource;

}

@Bean

public LdapTemplate ldapTemplate(ContextSource contextSource)

{

return new LdapTemplate(contextSource);

}

@Bean

public PersonDaoImpl personDaoImpl(LdapTemplate ldapTemplate)

{

PersonDaoImpl personRepo = new PersonDaoImpl();

personRepo.setLdapTemplate(ldapTemplate);

return personRepo;

}

}

package info.smartkit.eip.settings;

import org.springframework.boot.context.properties.EnableConfigurationProperties;

import org.springframework.context.annotation.Configuration;

@Configuration

@EnableConfigurationProperties

public class ServerSetting

{

//

private static ServerSetting instance = null;

protected ServerSetting()

{

// Exists only to defeat instantiation.

}

public static ServerSetting getInstance()

{

if (instance == null) {

instance = new ServerSetting();

}

return instance;

}

//

private String contextPath;

public String getContextPath()

{

return contextPath;

}

public void setContextPath(String contextPath)

{

this.contextPath = contextPath;

}

private String port;

public String getPort()

{

return port;

}

public void setPort(String port)

{

this.port = port;

}

// private String ip = "127.0.0.1";

//

// public String getIp()

// {

// return ip;

// }

//

// public void setIp(String ip)

// {

// this.ip = ip;

// }

public String getUrl()

{

return "http://127.0.0.1:" + ServerSetting.getInstance().getPort()

+ ServerSetting.getInstance().getContextPath() + "/";

}

}

package info.smartkit.eip.settings;

import org.springframework.boot.context.properties.EnableConfigurationProperties;

import org.springframework.context.annotation.Configuration;

@Configuration

@EnableConfigurationProperties

// Notice:Yaml properties later than properties injection.

// @ConfigurationProperties(prefix = "workflow")

public class WorkflowSetting

{

//

private static WorkflowSetting instance = null;

protected WorkflowSetting()

{

// Exists only to defeat instantiation.

}

public static WorkflowSetting getInstance()

{

if (instance == null) {

instance = new WorkflowSetting();

}

return instance;

}

//

private String bpmn;

public String getBpmn()

{

return bpmn;

}

public void setBpmn(String bpmn)

{

this.bpmn = bpmn;

}

private String image;

public String getImage()

{

return image;

}

public void setImage(String image)

{

this.image = image;

}

private String rule;

public String getRule()

{

return rule;

}

public void setRule(String rule)

{

this.rule = rule;

}

private String name;// enableDuplicateFiltering().name

public String getName()

{

return name;

}

public void setName(String name)

{

this.name = name;

}

}

package info.smartkit.eip.thirdParty;

import java.lang.reflect.Array;

public class BaiduLoc {

private Object location;

public Object getLocation() {

return location;

}

public void setLocation(Object location) {

this.location = location;

}

private String formatted\_address;

public String getFormatted\_address() {

return formatted\_address;

}

public void setFormatted\_address(String formatted\_address) {

this.formatted\_address = formatted\_address;

}

private String business;

public String getBusiness() {

return business;

}

public void setBusiness(String business) {

this.business = business;

}

private Object addressComponent;

public Object getAddressComponent() {

return addressComponent;

}

public void setAddressComponent(Object addressComponent) {

this.addressComponent = addressComponent;

}

private Object poiRegions;

public Object getPoiRegions() {

return poiRegions;

}

public void setPoiRegions(Object poiRegions) {

this.poiRegions = poiRegions;

}

private long cityCode;

public long getCityCode() {

return cityCode;

}

public void setCityCode(long cityCode) {

this.cityCode = cityCode;

}

}

package info.smartkit.eip.thirdParty;

//@see: http://www.cnblogs.com/wuhuacong/p/3995484.html

//@see: http://www.vchangyi.com/faq/202.html

public class WeixinUser {

private String name;

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

private String userId;

public String getUserId() {

return userId;

}

public void setUserId(String userId) {

this.userId = userId;

}

private String weixinId;

public String getWeixinId() {

return weixinId;

}

public void setWeixinId(String weixinId) {

this.weixinId = weixinId;

}

private String mobile;

public String getMobile() {

return mobile;

}

public void setMobile(String mobile) {

this.mobile = mobile;

}

private String email;

public String getEmail() {

return email;

}

public void setEmail(String email) {

this.email = email;

}

private String department;

public String getDepartment() {

return department;

}

public void setDepartment(String department) {

this.department = department;

}

private String title;

public String getTitle() {

return title;

}

public void setTitle(String title) {

this.title = title;

}

}

/\*--

Copyright (C) 2001 Brett McLaughlin.

All rights reserved.

Redistribution and use in source and binary forms, with or without

modification, are permitted provided that the following conditions

are met:

1. Redistributions of source code must retain the above copyright

notice, this list of conditions, and the following disclaimer.

2. Redistributions in binary form must reproduce the above copyright

notice, this list of conditions, and the disclaimer that follows

these conditions in the do\*\*\*\*\*entation and/or other materials

provided with the distribution.

3. The name "Building Java Enterprise Applications" must not be used

to endorse or promote products derived from this software without

prior written permission. For written permission, please contact

brett@newInstance.com.

In addition, we request (but do not require) that you include in the

end-user do\*\*\*\*\*entation provided with the redistribution and/or in the

software itself an acknowledgement equivalent to the following:

"This product includes software developed for the

'Building Java Enterprise Applications' book, by Brett McLaughlin

(O'Reilly & Associates)."

THIS SOFTWARE IS PROVIDED ``AS IS'' AND ANY EXPRESSED OR IMPLIED

WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES

OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE

DISCLAIMED. IN NO EVENT SHALL THE JDOM AUTHORS OR THE PROJECT

CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,

SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT

LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF

USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND

ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY,

OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT

OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF

SUCH DAMAGE.

\*/

package info.smartkit.eip.utils;

import java.util.HashMap;

import java.util.Properties;

import java.util.LinkedList;

import java.util.List;

import java.util.Map;

import javax.naming.Context;

import javax.naming.NameNotFoundException;

import javax.naming.NamingEnumeration;

import javax.naming.NamingException;

import javax.naming.directory.Attribute;

import javax.naming.directory.AttributeInUseException;

import javax.naming.directory.Attributes;

import javax.naming.directory.BasicAttribute;

import javax.naming.directory.BasicAttributes;

import javax.naming.directory.DirContext;

import javax.naming.directory.InitialDirContext;

import javax.naming.directory.ModificationItem;

import javax.naming.directory.NoSuchAttributeException;

import javax.naming.directory.SearchControls;

import javax.naming.directory.SearchResult;

public class LDAPManager {

/\*\* The OU (organizational unit) to add users to \*/

private static final String USERS\_OU =

"ou=People,o=forethought.com";

/\*\* The OU (organizational unit) to add groups to \*/

private static final String GROUPS\_OU =

"ou=Groups,o=forethought.com";

/\*\* The OU (organizational unit) to add permissions to \*/

private static final String PERMISSIONS\_OU =

"ou=Permissions,o=forethought.com";

/\*\* The default LDAP port \*/

private static final int DEFAULT\_PORT = 10389;

/\*\* The LDAPManager instance object \*/

private static Map instances = new HashMap();

/\*\* The connection, through a <code>DirContext</code>, to LDAP \*/

private DirContext context;

/\*\* The hostname connected to \*/

private String hostname;

/\*\* The port connected to \*/

private int port;

protected LDAPManager(String hostname, int port,

String username, String password)

throws NamingException {

context = getInitialContext(hostname, port, username, password);

// Only save data if we got connected

this.hostname = hostname;

this.port = port;

}

public static LDAPManager getInstance(String hostname,

int port,

String username,

String password)

throws NamingException {

// Construct the key for the supplied information

String key = new StringBuffer()

.append(hostname)

.append(":")

.append(port)

.append("|")

.append((username == null ? "" : username))

.append("|")

.append((password == null ? "" : password))

.toString();

if (!instances.containsKey(key)) {

synchronized (LDAPManager.class) {

if (!instances.containsKey(key)) {

LDAPManager instance =

new LDAPManager(hostname, port,

username, password);

instances.put(key, instance);

return instance;

}

}

}

return (LDAPManager)instances.get(key);

}

public static LDAPManager getInstance(String hostname, int port)

throws NamingException {

return getInstance(hostname, port, null, null);

}

public static LDAPManager getInstance(String hostname)

throws NamingException {

return getInstance(hostname, DEFAULT\_PORT, null, null);

}

public void addUser(String username, String firstName,

String lastName, String password)

throws NamingException {

// Create a container set of attributes

Attributes container = new BasicAttributes();

// Create the objectclass to add

Attribute objClasses = new BasicAttribute("objectClass");

objClasses.add("top");

objClasses.add("person");

objClasses.add("organizationalPerson");

objClasses.add("inetOrgPerson");

// Assign the username, first name, and last name

String cnValue = new StringBuffer(firstName)

.append(" ")

.append(lastName)

.toString();

Attribute cn = new BasicAttribute("cn", cnValue);

Attribute givenName = new BasicAttribute("givenName", firstName);

Attribute sn = new BasicAttribute("sn", lastName);

Attribute uid = new BasicAttribute("uid", username);

// Add password

Attribute userPassword =

new BasicAttribute("userpassword", password);

// Add these to the container

container.put(objClasses);

container.put(cn);

container.put(sn);

container.put(givenName);

container.put(uid);

container.put(userPassword);

// Create the entry

context.createSubcontext(getUserDN(username), container);

}

public void deleteUser(String username) throws NamingException {

try {

context.destroySubcontext(getUserDN(username));

} catch (NameNotFoundException e) {

// If the user is not found, ignore the error

}

}

public boolean isValidUser(String username, String password)

throws UserNotFoundException {

try {

DirContext context =

getInitialContext(hostname, port, getUserDN(username),

password);

return true;

} catch (javax.naming.NameNotFoundException e) {

throw new UserNotFoundException(username);

} catch (NamingException e) {

// Any other error indicates couldn't log user in

return false;

}

}

public void addGroup(String name, String description)

throws NamingException {

// Create a container set of attributes

Attributes container = new BasicAttributes();

// Create the objectclass to add

Attribute objClasses = new BasicAttribute("objectClass");

objClasses.add("top");

objClasses.add("groupOfUniqueNames");

objClasses.add("groupOfForethoughtNames");

// Assign the name and description to the group

Attribute cn = new BasicAttribute("cn", name);

Attribute desc = new BasicAttribute("description", description);

// Add these to the container

container.put(objClasses);

container.put(cn);

container.put(desc);

// Create the entry

context.createSubcontext(getGroupDN(name), container);

}

public void deleteGroup(String name) throws NamingException {

try {

context.destroySubcontext(getGroupDN(name));

} catch (NameNotFoundException e) {

// If the group is not found, ignore the error

}

}

public void addPermission(String name, String description)

throws NamingException {

// Create a container set of attributes

Attributes container = new BasicAttributes();

// Create the objectclass to add

Attribute objClasses = new BasicAttribute("objectClass");

objClasses.add("top");

objClasses.add("forethoughtPermission");

// Assign the name and description to the group

Attribute cn = new BasicAttribute("cn", name);

Attribute desc = new BasicAttribute("description", description);

// Add these to the container

container.put(objClasses);

container.put(cn);

container.put(desc);

// Create the entry

context.createSubcontext(getPermissionDN(name), container);

}

public void deletePermission(String name) throws NamingException {

try {

context.destroySubcontext(getPermissionDN(name));

} catch (NameNotFoundException e) {

// If the permission is not found, ignore the error

}

}

public void assignUser(String username, String groupName)

throws NamingException {

try {

ModificationItem[] mods = new ModificationItem[1];

Attribute mod =

new BasicAttribute("uniqueMember",

getUserDN(username));

mods[0] =

new ModificationItem(DirContext.ADD\_ATTRIBUTE, mod);

context.modifyAttributes(getGroupDN(groupName), mods);

} catch (AttributeInUseException e) {

// If user is already added, ignore exception

}

}

public void removeUser(String username, String groupName)

throws NamingException {

try {

ModificationItem[] mods = new ModificationItem[1];

Attribute mod =

new BasicAttribute("uniqueMember",

getUserDN(username));

mods[0] =

new ModificationItem(DirContext.REMOVE\_ATTRIBUTE, mod);

context.modifyAttributes(getGroupDN(groupName), mods);

} catch (NoSuchAttributeException e) {

// If user is not assigned, ignore the error

}

}

public boolean userInGroup(String username, String groupName)

throws NamingException {

// Set up attributes to search for

String[] searchAttributes = new String[1];

searchAttributes[0] = "uniqueMember";

Attributes attributes =

context.getAttributes(getGroupDN(groupName),

searchAttributes);

if (attributes != null) {

Attribute memberAtts = attributes.get("uniqueMember");

if (memberAtts != null) {

for (NamingEnumeration vals = memberAtts.getAll();

vals.hasMoreElements();

) {

if (username.equalsIgnoreCase(

getUserUID((String)vals.nextElement()))) {

return true;

}

}

}

}

return false;

}

public List getMembers(String groupName) throws NamingException {

List members = new LinkedList();

// Set up attributes to search for

String[] searchAttributes = new String[1];

searchAttributes[0] = "uniqueMember";

Attributes attributes =

context.getAttributes(getGroupDN(groupName),

searchAttributes);

if (attributes != null) {

Attribute memberAtts = attributes.get("uniqueMember");

if (memberAtts != null) {

for (NamingEnumeration vals = memberAtts.getAll();

vals.hasMoreElements();

members.add(

getUserUID((String)vals.nextElement()))) ;

}

}

return members;

}

public List getGroups(String username) throws NamingException {

List groups = new LinkedList();

// Set up criteria to search on

String filter = new StringBuffer()

.append("(&")

.append("(objectClass=groupOfForethoughtNames)")

.append("(uniqueMember=")

.append(getUserDN(username))

.append(")")

.append(")")

.toString();

// Set up search constraints

SearchControls cons = new SearchControls();

cons.setSearchScope(SearchControls.ONELEVEL\_SCOPE);

NamingEnumeration results =

context.search(GROUPS\_OU, filter, cons);

while (results.hasMore()) {

SearchResult result = (SearchResult)results.next();

groups.add(getGroupCN(result.getName()));

}

return groups;

}

public void assignPermission(String groupName, String permissionName)

throws NamingException {

try {

ModificationItem[] mods = new ModificationItem[1];

Attribute mod =

new BasicAttribute("uniquePermission",

getPermissionDN(permissionName));

mods[0] =

new ModificationItem(DirContext.ADD\_ATTRIBUTE, mod);

context.modifyAttributes(getGroupDN(groupName), mods);

} catch (AttributeInUseException e) {

// Ignore the attribute if it is already assigned

}

}

public void revokePermission(String groupName, String permissionName)

throws NamingException {

try {

ModificationItem[] mods = new ModificationItem[1];

Attribute mod =

new BasicAttribute("uniquePermission",

getPermissionDN(permissionName));

mods[0] =

new ModificationItem(DirContext.REMOVE\_ATTRIBUTE, mod);

context.modifyAttributes(getGroupDN(groupName), mods);

} catch (NoSuchAttributeException e) {

// Ignore errors if the attribute doesn't exist

}

}

public boolean hasPermission(String groupName, String permissionName)

throws NamingException {

// Set up attributes to search for

String[] searchAttributes = new String[1];

searchAttributes[0] = "uniquePermission";

Attributes attributes =

context.getAttributes(getGroupDN(groupName),

searchAttributes);

if (attributes != null) {

Attribute permAtts = attributes.get("uniquePermission");

if (permAtts != null) {

for (NamingEnumeration vals = permAtts.getAll();

vals.hasMoreElements();

) {

if (permissionName.equalsIgnoreCase(

getPermissionCN((String)vals.nextElement()))) {

return true;

}

}

}

}

return false;

}

public List getPermissions(String groupName) throws NamingException {

List permissions = new LinkedList();

// Set up attributes to search for

String[] searchAttributes = new String[1];

searchAttributes[0] = "uniquePermission";

Attributes attributes =

context.getAttributes(getGroupDN(groupName),

searchAttributes);

if (attributes != null) {

Attribute permAtts = attributes.get("uniquePermission");

if (permAtts != null) {

for (NamingEnumeration vals = permAtts.getAll();

vals.hasMoreElements();

permissions.add(

getPermissionCN((String)vals.nextElement()))) ;

}

}

return permissions;

}

private String getUserDN(String username) {

return new StringBuffer()

.append("uid=")

.append(username)

.append(",")

.append(USERS\_OU)

.toString();

}

private String getUserUID(String userDN) {

int start = userDN.indexOf("=");

int end = userDN.indexOf(",");

if (end == -1) {

end = userDN.length();

}

return userDN.substring(start+1, end);

}

private String getGroupDN(String name) {

return new StringBuffer()

.append("cn=")

.append(name)

.append(",")

.append(GROUPS\_OU)

.toString();

}

private String getGroupCN(String groupDN) {

int start = groupDN.indexOf("=");

int end = groupDN.indexOf(",");

if (end == -1) {

end = groupDN.length();

}

return groupDN.substring(start+1, end);

}

private String getPermissionDN(String name) {

return new StringBuffer()

.append("cn=")

.append(name)

.append(",")

.append(PERMISSIONS\_OU)

.toString();

}

private String getPermissionCN(String permissionDN) {

int start = permissionDN.indexOf("=");

int end = permissionDN.indexOf(",");

if (end == -1) {

end = permissionDN.length();

}

return permissionDN.substring(start+1, end);

}

private DirContext getInitialContext(String hostname, int port,

String username, String password)

throws NamingException {

String providerURL =

new StringBuffer("ldap://")

.append(hostname)

.append(":")

.append(port)

.toString();

Properties props = new Properties();

props.put(Context.INITIAL\_CONTEXT\_FACTORY,

"com.sun.jndi.ldap.LdapCtxFactory");

props.put(Context.PROVIDER\_URL, providerURL);

if ((username != null) && (!username.equals(""))) {

props.put(Context.SECURITY\_AUTHENTICATION, "simple");

props.put(Context.SECURITY\_PRINCIPAL, username);

props.put(Context.SECURITY\_CREDENTIALS,

((password == null) ? "" : password));

}

return new InitialDirContext(props);

}

}

/\*

\* Building Java Enterprise Applications

\* By Brett McLaughlin

\* First Edition March 2002

\* ISBN: 0-596-00123-1

\* http://www.oreilly.com/catalog/javentappsv1/

\*/

@SuppressWarnings("serial")

class UserNotFoundException extends RuntimeException {

public UserNotFoundException(String userId) {

super("could not find user '" + userId + "'.");

}

}package info.smartkit.eip.utils;

import java.io.IOException;

import java.nio.charset.StandardCharsets;

import java.util.List;

import javax.servlet.http.Cookie;

import org.activiti.engine.impl.util.json.JSONObject;

import org.apache.commons.httpclient.cookie.CookieOrigin;

import org.apache.commons.httpclient.cookie.MalformedCookieException;

import org.apache.http.HttpEntity;

import org.apache.http.HttpResponse;

import org.apache.http.NameValuePair;

import org.apache.http.client.ClientProtocolException;

import org.apache.http.client.entity.UrlEncodedFormEntity;

import org.apache.http.client.methods.HttpGet;

import org.apache.http.client.methods.HttpPost;

import org.apache.http.client.params.ClientPNames;

import org.apache.http.cookie.CookieSpec;

import org.apache.http.cookie.CookieSpecFactory;

import org.apache.http.impl.client.DefaultHttpClient;

import org.apache.http.impl.cookie.BrowserCompatSpec;

import org.apache.http.params.HttpParams;

import org.apache.http.util.EntityUtils;

public class ParserTools

{

// /////////////////////////////http 请求/////////////////////////////////////

/\*\*

\* post 获取 rest 资源

\*

\* @param url

\* @param name\_value\_pair

\* @return

\* @throws IOException

\*/

public static String doPost(String url, List<NameValuePair> name\_value\_pair) throws IOException

{

String body = "{}";

DefaultHttpClient httpclient = new DefaultHttpClient();

try {

HttpPost httpost = new HttpPost(url);

httpost.setEntity(new UrlEncodedFormEntity(name\_value\_pair, StandardCharsets.UTF\_8));

HttpResponse response = httpclient.execute(httpost);

HttpEntity entity = response.getEntity();

body = EntityUtils.toString(entity);

} finally {

httpclient.getConnectionManager().shutdown();

}

return body;

}

/\*\*

\* get 获取 rest 资源

\*

\* @param url

\* @return

\* @throws ClientProtocolException

\* @throws IOException

\*/

public static String doGet(String url) throws ClientProtocolException, IOException

{

String body = "{}";

DefaultHttpClient httpclient = new DefaultHttpClient();

httpclient.getCookieSpecs().register("easy", csf);

httpclient.getParams().setParameter(ClientPNames.COOKIE\_POLICY, "easy");

try {

HttpGet httpget = new HttpGet(url);

HttpResponse response = httpclient.execute(httpget);

HttpEntity entity = response.getEntity();

body = EntityUtils.toString(entity);

// parsing JSON

JSONObject result = new JSONObject(body); // Convert String to JSON Object

} finally {

httpclient.getConnectionManager().shutdown();

}

return body;

}

// customer cookie policy, ignore cookie check

private static CookieSpecFactory csf = new CookieSpecFactory()

{

@Override

public CookieSpec newInstance(HttpParams params)

{

return new BrowserCompatSpec()

{

public void validate(Cookie cookie, CookieOrigin origin) throws MalformedCookieException

{

// Oh, I am easy

}

};

}

};

}

package info.smartkit.eip.utils;

import java.io.IOException;

import javax.servlet.Filter;

import javax.servlet.FilterChain;

import javax.servlet.FilterConfig;

import javax.servlet.ServletException;

import javax.servlet.ServletRequest;

import javax.servlet.ServletResponse;

import javax.servlet.http.HttpServletResponse;

import org.springframework.stereotype.Component;

/\*\*

\* @author yangboz

\* @see https://spring.io/guides/gs/rest-service-cors/

\*/

@Component

public class SimpleCORSFilter implements Filter

{

@Override

public void doFilter(ServletRequest req, ServletResponse res, FilterChain chain) throws IOException,

ServletException

{

HttpServletResponse response = (HttpServletResponse) res;

response.setHeader("Access-Control-Allow-Origin", "\*");

response.setHeader("Access-Control-Allow-Methods", "POST, GET, PUT,OPTIONS, DELETE,PATCH");

response.setHeader("Access-Control-Max-Age", "3600");

// @see:

// http://stackoverflow.com/questions/25727306/request-header-field-access-control-allow-headers-is-not-allowed-by-access-contr

response.setHeader("Access-Control-Allow-Headers", "Origin, X-Requested-With, Content-Type, Accept");

// @see:

// http://stackoverflow.com/questions/10548883/request-header-field-authorization-is-not-allowed-error-tastypie

response.addHeader("Access-Control-Allow-Headers", "Authorization");

chain.doFilter(req, res);

}

@Override

public void init(FilterConfig filterConfig)

{

}

@Override

public void destroy()

{

}

}

package info.smartkit.eip;

import javax.jms.Connection;

import javax.jms.ConnectionFactory;

import javax.jms.Destination;

import javax.jms.JMSException;

import javax.jms.Message;

import javax.jms.MessageConsumer;

import javax.jms.Session;

import javax.jms.TextMessage;

import org.apache.activemq.ActiveMQConnection;

import org.apache.activemq.ActiveMQConnectionFactory;

//@see: http://www.coderpanda.com/jms-example-using-apache-activemq/

public class ActivemqReceiverTest {

private ConnectionFactory factory = null;

private Connection connection = null;

private Session session = null;

private Destination destination = null;

private MessageConsumer consumer = null;

public ActivemqReceiverTest() {

}

public void receiveMessage() {

try {

factory = new ActiveMQConnectionFactory(

ActiveMQConnection.DEFAULT\_BROKER\_URL);

connection = factory.createConnection();

connection.start();

session = connection.createSession(false, Session.AUTO\_ACKNOWLEDGE);

destination = session.createQueue("SAMPLEQUEUE");

consumer = session.createConsumer(destination);

Message message = consumer.receive();

if (message instanceof TextMessage) {

TextMessage text = (TextMessage) message;

System.out.println("Message is : " + text.getText());

}

} catch (JMSException e) {

e.printStackTrace();

}

}

public static void main(String[] args) {

ActivemqReceiverTest receiver = new ActivemqReceiverTest();

receiver.receiveMessage();

}

}

package info.smartkit.eip;

import javax.jms.Connection;

import javax.jms.ConnectionFactory;

import javax.jms.Destination;

import javax.jms.JMSException;

import javax.jms.MessageProducer;

import javax.jms.Session;

import javax.jms.TextMessage;

import org.apache.activemq.ActiveMQConnection;

import org.apache.activemq.ActiveMQConnectionFactory;

//@see: http://www.coderpanda.com/jms-example-using-apache-activemq/

public class ActivemqSenderTest {

private ConnectionFactory factory = null;

private Connection connection = null;

private Session session = null;

private Destination destination = null;

private MessageProducer producer = null;

public ActivemqSenderTest() {

}

public void sendMessage() {

try {

factory = new ActiveMQConnectionFactory(

ActiveMQConnection.DEFAULT\_BROKER\_URL);

connection = factory.createConnection();

connection.start();

session = connection.createSession(false, Session.AUTO\_ACKNOWLEDGE);

destination = session.createQueue("SAMPLEQUEUE");

producer = session.createProducer(destination);

TextMessage message = session.createTextMessage();

message.setText("Hello ...This is a sample message..sending from FirstClient");

producer.send(message);

System.out.println("Sent: " + message.getText());

} catch (JMSException e) {

e.printStackTrace();

}

}

public static void main(String[] args) {

ActivemqSenderTest sender = new ActivemqSenderTest();

sender.sendMessage();

}

}

package info.smartkit.eip;

import java.io.IOException;

import java.net.UnknownHostException;

import org.apache.activemq.transport.stomp.Stomp;

import org.apache.activemq.transport.stomp.Stomp.Headers.Subscribe;

import org.apache.activemq.transport.stomp.StompConnection;

import org.apache.activemq.transport.stomp.StompFrame;

import org.junit.Test;

public class ActivemqUnitTest {

@Test

public void test() throws Exception{

StompConnection connection = new StompConnection();

connection.open("localhost", 61613);

connection.connect("system", "manager");

StompFrame connect = connection.receive();

if (!connect.getAction().equals(Stomp.Responses.CONNECTED)) {

throw new Exception ("Not connected");

}

connection.begin("tx1");

connection.send("/queue/test", "message1", "tx1", null);

connection.send("/queue/test", "message2", "tx1", null);

connection.commit("tx1");

connection.subscribe("/queue/test", Subscribe.AckModeValues.CLIENT);

connection.begin("tx2");

StompFrame message = connection.receive();

System.out.println(message.getBody());

connection.ack(message, "tx2");

message = connection.receive();

System.out.println(message.getBody());

connection.ack(message, "tx2");

connection.commit("tx2");

connection.disconnect();

}

}

package info.smartkit.eip;

import java.util.HashMap;

import java.util.Map;

import org.activiti.engine.impl.test.PluggableActivitiTestCase;

import org.activiti.engine.runtime.ProcessInstance;

import org.activiti.engine.task.Task;

import org.activiti.engine.test.Deployment;

public class ActivitiDynamicExternalFormTest extends PluggableActivitiTestCase {

@Deployment(resources = { "test/FormKey.bpmn20.xml",

"test/start.form", "test/first-step.form",

"test/second-step.form" })

public void testTaskFormsWithVacationRequestProcess() {

// Get start form

String procDefId = repositoryService.createProcessDefinitionQuery()

.singleResult().getId();

Object startForm = formService.getRenderedStartForm(procDefId);

assertNotNull(startForm);

assertEquals("<input id=\"start-name\" />", startForm);

Map<String, String> formProperties = new HashMap<String, String>();

formProperties.put("startName", "yangboz");

ProcessInstance processInstance = formService.submitStartFormData(

procDefId, formProperties);

assertNotNull(processInstance);

Task task = taskService.createTaskQuery().taskAssignee("user1")

.singleResult();

Object renderedTaskForm = formService.getRenderedTaskForm(task.getId());

assertEquals(

"<input id=\"start-name\" value=\"yangboz\" />\n<input id=\"first-name\" />",

renderedTaskForm);

formProperties = new HashMap<String, String>();

formProperties.put("firstName", "yangboz");

formService.submitTaskFormData(task.getId(), formProperties);

task = taskService.createTaskQuery().taskAssignee("user2")

.singleResult();

assertNotNull(task);

renderedTaskForm = formService.getRenderedTaskForm(task.getId());

assertEquals("<input id=\"first-name\" value=\"yangboz\" />",

renderedTaskForm);

}

}

package info.smartkit.eip;

import static org.junit.Assert.assertEquals;

import static org.junit.Assert.assertNotNull;

import static org.junit.Assert.assertNull;

import java.io.FileInputStream;

import java.util.HashMap;

import java.util.List;

import java.util.Map;

import java.util.Map.Entry;

import java.util.Set;

import org.activiti.engine.FormService;

import org.activiti.engine.HistoryService;

import org.activiti.engine.RepositoryService;

import org.activiti.engine.RuntimeService;

import org.activiti.engine.TaskService;

import org.activiti.engine.form.FormProperty;

import org.activiti.engine.form.StartFormData;

import org.activiti.engine.form.TaskFormData;

import org.activiti.engine.history.HistoricDetail;

import org.activiti.engine.repository.ProcessDefinition;

import org.activiti.engine.runtime.ProcessInstance;

import org.activiti.engine.task.Task;

import org.activiti.engine.test.ActivitiRule;

import org.apache.commons.lang.builder.ToStringBuilder;

import org.junit.Rule;

import org.junit.Test;

//@see http://www.kafeitu.me/activiti/2012/08/05/diff-activiti-workflow-forms.html

public class ActivitiDynamicFormTest {

// private String filename = "/Users/yangboz/Documents/Git/north-american-adventure/RushuMicroService/eip-rushucloud/src/main/resources/test/DynamicForm.bpmn";

@Rule

public ActivitiRule activitiRule = new ActivitiRule();

@Test

public void startProcess() throws Exception {

RepositoryService repositoryService = activitiRule

.getRepositoryService();

repositoryService

.createDeployment()

// .addInputStream("DynamicForm.bpmn20.xml",

// new FileInputStream(filename))

// .deploy();

.addClasspathResource("test/DynamicForm.bpmn")

.addClasspathResource("test/DynamicForm.png")

.enableDuplicateFiltering()

.name("dynamicForm")

.deploy();

ProcessDefinition processDefinition = repositoryService

.createProcessDefinitionQuery()

.processDefinitionKey("DynamicForm").latestVersion()

.singleResult();

FormService formService = activitiRule.getFormService();

StartFormData startFormData = formService

.getStartFormData(processDefinition.getId());

assertNull(startFormData.getFormKey());

Map<String, String> formProperties = new HashMap<String, String>();

formProperties.put("name", "YangboZ");

ProcessInstance processInstance = formService.submitStartFormData(

processDefinition.getId(), formProperties);

assertNotNull(processInstance);

// 运行时变量

RuntimeService runtimeService = activitiRule.getRuntimeService();

Map<String, Object> variables = runtimeService

.getVariables(processInstance.getId());

assertEquals(variables.size(), 1);

Set<Entry<String, Object>> entrySet = variables.entrySet();

for (Entry<String, Object> entry : entrySet) {

System.out.println(entry.getKey() + "=" + entry.getValue());

}

// 历史记录

HistoryService historyService = activitiRule.getHistoryService();

List<HistoricDetail> list = historyService.createHistoricDetailQuery()

.formProperties().list();

assertEquals(1, list.size());

// 获取第一个节点

TaskService taskService = activitiRule.getTaskService();

Task task = taskService.createTaskQuery().singleResult();

assertEquals("First Step", task.getName());

TaskFormData taskFormData = formService.getTaskFormData(task.getId());

assertNotNull(taskFormData);

assertNull(taskFormData.getFormKey());

List<FormProperty> taskFormProperties = taskFormData

.getFormProperties();

assertNotNull(taskFormProperties);

for (FormProperty formProperty : taskFormProperties) {

System.out

.println(ToStringBuilder.reflectionToString(formProperty));

}

formProperties = new HashMap<String, String>();

formProperties.put("setInFirstStep", "11/27/2014");

formService.submitTaskFormData(task.getId(), formProperties);

// 获取第二个节点

task = taskService.createTaskQuery().taskName("Second Step")

.singleResult();

assertNotNull(task);

taskFormData = formService.getTaskFormData(task.getId());

assertNotNull(taskFormData);

List<FormProperty> formProperties2 = taskFormData.getFormProperties();

assertNotNull(formProperties2);

assertEquals(1, formProperties2.size());

assertNotNull(formProperties2.get(0).getValue());

assertEquals(formProperties2.get(0).getValue(), "11/28/2014");

}

}

package info.smartkit.eip;

import org.activiti.engine.runtime.ProcessInstance;

import org.activiti.engine.task.Task;

import org.activiti.engine.test.ActivitiRule;

import org.activiti.engine.test.Deployment;

import org.junit.Rule;

import org.junit.Test;

import static org.junit.Assert.\*;

//Dynamic,@see: http://stacktrace.be/blog/2013/03/dynamic-process-creation-and-deployment-in-100-lines/

//@see:https://github.com/Activiti/activiti-unit-test-template/blob/master/src/test/java/org/activiti/MyUnitTest.java

public class ActivitiUnitTest {

@Rule

public ActivitiRule activitiRule = new ActivitiRule();

@Test

// @Deployment(resources = {"org/activiti/test/my-process.bpmn20.xml"})

@Deployment(resources = {"processes/ReimbursementRequest.bpmn20.xml"})

public void test() {

// ProcessInstance processInstance = activitiRule.getRuntimeService().startProcessInstanceByKey("my-process");

ProcessInstance processInstance = activitiRule.getRuntimeService().startProcessInstanceByKey("reimbursementRequest");

assertNotNull(processInstance);

Task task = activitiRule.getTaskService().createTaskQuery().singleResult();

assertEquals("Activiti is awesome!", task.getName());

}

}

package info.smartkit.eip;

/\*

\* Licensed to the Apache Software Foundation (ASF) under one

\* or more contributor license agreements. See the NOTICE file

\* distributed with this work for additional information

\* regarding copyright ownership. The ASF licenses this file

\* to you under the Apache License, Version 2.0 (the

\* "License"); you may not use this file except in compliance

\* with the License. You may obtain a copy of the License at

\*

\* http://www.apache.org/licenses/LICENSE-2.0

\*

\* Unless required by applicable law or agreed to in writing,

\* software distributed under the License is distributed on an

\* "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY

\* KIND, either express or implied. See the License for the

\* specific language governing permissions and limitations

\* under the License.

\*

\*/

import java.io.File;

import org.apache.directory.server.core.DefaultDirectoryService;

import org.apache.directory.server.core.DirectoryService;

import org.apache.directory.server.core.entry.ClonedServerEntry;

import org.apache.directory.server.core.filtering.EntryFilteringCursor;

import org.apache.directory.server.core.interceptor.context.AddOperationContext;

import org.apache.directory.server.core.interceptor.context.BindOperationContext;

import org.apache.directory.server.core.interceptor.context.DeleteOperationContext;

import org.apache.directory.server.core.interceptor.context.ListOperationContext;

import org.apache.directory.server.core.interceptor.context.ModifyOperationContext;

import org.apache.directory.server.core.interceptor.context.MoveAndRenameOperationContext;

import org.apache.directory.server.core.interceptor.context.MoveOperationContext;

import org.apache.directory.server.core.interceptor.context.RenameOperationContext;

import org.apache.directory.server.core.interceptor.context.SearchOperationContext;

import org.apache.directory.server.core.interceptor.context.UnbindOperationContext;

import org.apache.directory.server.core.partition.AbstractPartition;

import org.apache.directory.server.core.partition.impl.btree.jdbm.JdbmPartition;

import org.apache.directory.shared.ldap.name.LdapDN;

import org.junit.Assert;

import org.junit.Test;

public class LdapPartitionIntegrationTest {

@Test

public void startTheServerWithThePartition() throws Exception {

DirectoryService directoryService;

directoryService = new DefaultDirectoryService();

directoryService.setShutdownHookEnabled(true);

File workingDir = new File("work");

directoryService.setWorkingDirectory(workingDir);

// Create a new partition

JdbmPartition helloPartition = new JdbmPartition();

helloPartition.setId("hello");

helloPartition.setSuffix("ou=hello");

helloPartition.init(directoryService);

directoryService.addPartition(helloPartition);

directoryService.startup();

ClonedServerEntry entry = directoryService.getAdminSession().lookup(new LdapDN("ou=hello"));

Assert.assertNotNull(entry);

directoryService.shutdown();

}

}package info.smartkit.eip.test.activiti.delegate;

import org.activiti.engine.delegate.DelegateExecution;

import org.activiti.engine.delegate.JavaDelegate;

//

public class ServiceClassDelegateSample implements JavaDelegate {

/\*

\* (non-Javadoc)

\*

\* @see

\* org.activiti.engine.delegate.JavaDelegate#execute(org.activiti.engine

\* .delegate.DelegateExecution)

\*/

public void execute(DelegateExecution delegateExecution) throws Exception {

System.out.println("Executed process with key "

+ delegateExecution.getProcessBusinessKey()

+ " with process definition Id "

+ delegateExecution.getProcessDefinitionId()

+ " with process instance Id "

+ delegateExecution.getProcessInstanceId()

+ " and current task name is "

+ delegateExecution.getCurrentActivityName());

}

}package info.smartkit.eip.test.jpa;

import java.util.HashSet;

import java.util.Iterator;

import java.util.Set;

import javax.persistence.EntityManager;

import javax.persistence.PersistenceContext;

import info.smartkit.eip.models.Category;

import org.junit.After;

import org.junit.Before;

import org.junit.Test;

import org.junit.runner.RunWith;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

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

import org.springframework.boot.test.SpringApplicationConfiguration;

import org.springframework.context.annotation.Bean;

import org.springframework.orm.jpa.support.PersistenceAnnotationBeanPostProcessor;

import org.springframework.test.context.junit4.SpringJUnit4ClassRunner;

import org.springframework.test.context.web.WebAppConfiguration;

import org.springframework.web.context.WebApplicationContext;

/\*\*

\* Integration test to run the application.

\*

\* @author Oliver Gierke

\* @author Dave Syer

\*/

@RunWith(SpringJUnit4ClassRunner.class)

@SpringApplicationConfiguration(classes = JPATest.class)

@WebAppConfiguration

// Enable JMX so we can test the MBeans (you can't do this in a properties file)

//@TestPropertySource(properties = { "spring.jmx.enabled:true","spring.datasource.jmx-enabled:true" })

//@ActiveProfiles("scratch")

// Separate profile for web tests to avoid clashing databases

public class JPATest {

@Autowired

private WebApplicationContext context;

@Bean

public PersistenceAnnotationBeanPostProcessor persistenceBeanPostProcessor() {

return new PersistenceAnnotationBeanPostProcessor();

}

// EntityManagerFactory emf = null;

@PersistenceContext

protected EntityManager em;

private static Logger LOG = LoggerFactory.getLogger(JPATest.class);

@Before

public void before() {

// //根据在persistence.xml中配置的persistence-unit name 创建EntityManagerFactory

// emf = Persistence.createEntityManagerFactory("myJPA");

LOG.info("entityManager:"+em.toString());

}

/\*\*

\* 保存父组织,级联保存子组织

\*/

@Test

public void testAddParentCat() {

//父组织

Category chinaOrg = new Category();

chinaOrg.setName("中国");

chinaOrg.setIcon("CHINA");

//子组织

Category gdOrg = new Category();

gdOrg.setName("广东");

gdOrg.setIcon("GD");

//子组织

Category gxOrg = new Category();

gxOrg.setName("广西");

gxOrg.setIcon("GX");

Set<Category> children = new HashSet<Category>();

children.add(gdOrg);

children.add(gxOrg);

//添加子组织

chinaOrg.setChildren(children);

// EntityManager em = emf.createEntityManager();

em.getTransaction().begin();

em.persist(chinaOrg);

em.getTransaction().commit();

em.close();

}

/\*\*

\* 保存子组织

\*/

@Test

public void testAddChildOrg() {

Category gzOrg = new Category();

gzOrg.setName("广州");

gzOrg.setIcon("GZ");

// EntityManager em = emf.createEntityManager();

//找出广州所属的父组织广东

Category parent = em.find(Category.class, 3L);

//设置广州的父组织

gzOrg.setParent(parent);

em.getTransaction().begin();

//保存广州

em.persist(gzOrg);

em.getTransaction().commit();

em.close();

}

/\*\*

\* 根据子组织查询父组织

\*/

@Test

public void testQueryParentByChild() {

// EntityManager em = emf.createEntityManager();

//找出广州

Category gzOrg = em.find(Category.class, 4L);

//找出父组织

Category parent = gzOrg.getParent();

System.out.println("父组织名称："+parent.getName());

em.close();

}

/\*\*

\* 根据父组织查询子组织

\*/

@Test

public void testQueryChildrenByParent() {

// EntityManager em = emf.createEntityManager();

//找出广东

Category gdOrg = em.find(Category.class, 3L);

//找出子组织

Set<Category> children = gdOrg.getChildren();

Iterator<Category> it = children.iterator();

while(it.hasNext()) {

Category child = it.next();

System.out.println("子组织名称："+child.getName());

}

em.close();

}

/\*\*

\* 关闭EntityManagerFactory

\*/

@After

public void after() {

// if(null != emf) {

// emf.close();

// }

}

}

/\*

\* DynamicJasper: A library for creating reports dynamically by specifying

\* columns, groups, styles, etc. at runtime. It also saves a lot of development

\* time in many cases! (http://sourceforge.net/projects/dynamicjasper)

\*

\* Copyright (C) 2008 FDV Solutions (http://www.fdvsolutions.com)

\*

\* This library is free software; you can redistribute it and/or

\* modify it under the terms of the GNU Lesser General Public

\*

\* License as published by the Free Software Foundation; either

\*

\* version 2.1 of the License, or (at your option) any later version.

\*

\* This library is distributed in the hope that it will be useful,

\* but WITHOUT ANY WARRANTY; without even the implied warranty of

\*

\* MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU

\*

\* Lesser General Public License for more details.

\*

\* You should have received a copy of the GNU Lesser General Public

\* License along with this library; if not, write to the Free Software

\*

\* Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA

\*

\*

\*/

package info.smartkit.eip.test.reports.domain;

public class DummyLevel3 {

private String name;

private Long number;

public DummyLevel3(String name, Long number) {

super();

this.name = name;

this.number = number;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public Long getNumber() {

return number;

}

public void setNumber(Long number) {

this.number = number;

}

}

/\*

\* DynamicJasper: A library for creating reports dynamically by specifying

\* columns, groups, styles, etc. at runtime. It also saves a lot of development

\* time in many cases! (http://sourceforge.net/projects/dynamicjasper)

\*

\* Copyright (C) 2008 FDV Solutions (http://www.fdvsolutions.com)

\*

\* This library is free software; you can redistribute it and/or

\* modify it under the terms of the GNU Lesser General Public

\*

\* License as published by the Free Software Foundation; either

\*

\* version 2.1 of the License, or (at your option) any later version.

\*

\* This library is distributed in the hope that it will be useful,

\* but WITHOUT ANY WARRANTY; without even the implied warranty of

\*

\* MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU

\*

\* Lesser General Public License for more details.

\*

\* You should have received a copy of the GNU Lesser General Public

\* License along with this library; if not, write to the Free Software

\*

\* Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA

\*

\*

\*/

package info.smartkit.eip.test.reports.domain;

import java.io.InputStream;

import java.text.ParseException;

import java.text.SimpleDateFormat;

import java.util.ArrayList;

import java.util.Collections;

import java.util.Date;

import java.util.List;

import java.util.Random;

public class Product {

private Long id;

private String productLine;

private String item;

private String state;

private String branch;

private Long quantity;

private Float amount;

private Code code = new Code();

private Boolean showAsPrices;

public static List statistics\_ = new ArrayList();

private static SimpleDateFormat formatter = new SimpleDateFormat("dd/MM/yyyy");

static String[] images = {"confused.gif","cool.gif","happy.gif","puaj.gif","ungry.gif","what.gif"};

static int counter = 0;

static {

try {

statistics\_.add(new Statistic(formatter.parse("01/01/2003"),"West",14.3f,50.4f,43.1f));

statistics\_.add(new Statistic(formatter.parse("01/01/2004"),"West",40.0f,49.4f,44.5f));

statistics\_.add(new Statistic(formatter.parse("01/01/2005"),"North",33.3f,63.4f,45f));

statistics\_.add(new Statistic(formatter.parse("01/01/2006"),"East",91.1f,34.4f,46f));

statistics\_.add(new Statistic(formatter.parse("01/01/2007"),"South",99.3f,52.4f,47f));

} catch (ParseException e) {

e.printStackTrace();

}

}

public List getStatistics(){

return statistics\_;

}

public List getEmptyStatistics(){

return Collections.EMPTY\_LIST;

}

public Product(){}

public Product(Long id, String productLine, String item, String state, String branch, Long quantity, Float amount) {

this.id = id;

this.productLine = productLine;

this.item = item;

this.state = state;

this.branch = branch;

this.quantity = quantity;

this.amount = amount;

}

public Product(Long id, String productLine, String item, String state, String branch, Long quantity, Float amount, boolean showAsPrices) {

this(id, productLine, item, state, branch, quantity, amount);

this.showAsPrices = showAsPrices;

}

public String getBranch() {

return branch;

}

public void setBranch(String branch) {

this.branch = branch;

}

public Float getAmount() {

return amount;

// return new Float(quantity.floatValue());

}

public void setAmount(Float amount) {

this.amount = amount;

}

public Long getId() {

return id;

}

public void setId(Long id) {

this.id = id;

}

public String getItem() {

return item;

}

public void setItem(String item) {

this.item = item;

}

public String getProductLine() {

return productLine;

}

public void setProductLine(String productLine) {

this.productLine = productLine;

}

public Long getQuantity() {

return quantity;

}

public void setQuantity(Long quantity) {

this.quantity = quantity;

}

public String getState() {

return state;

}

public void setState(String state) {

this.state = state;

}

public Date getDate() {

return new Date();

}

public Boolean getIsAvailable(){

return Boolean.valueOf(random.nextBoolean());

}

public Boolean getShowAsPrices() {

return showAsPrices;

}

public void setShowAsPrices(Boolean showAsPrices) {

this.showAsPrices = showAsPrices;

}

static Random random = new Random();

public InputStream getImage() {

InputStream ret = this.getClass().getClassLoader().getResourceAsStream("images/emoticons/" + images[counter++]);

if (counter >= images.length)

counter = 0;

return ret;

}

public Code getCode() {

return code;

}

public class Code {

public String getCode() {

return "001-123ABC-HRC";

}

}

}

/\*

\* DynamicJasper: A library for creating reports dynamically by specifying

\* columns, groups, styles, etc. at runtime. It also saves a lot of development

\* time in many cases! (http://sourceforge.net/projects/dynamicjasper)

\*

\* Copyright (C) 2008 FDV Solutions (http://www.fdvsolutions.com)

\*

\* This library is free software; you can redistribute it and/or

\* modify it under the terms of the GNU Lesser General Public

\*

\* License as published by the Free Software Foundation; either

\*

\* version 2.1 of the License, or (at your option) any later version.

\*

\* This library is distributed in the hope that it will be useful,

\* but WITHOUT ANY WARRANTY; without even the implied warranty of

\*

\* MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU

\*

\* Lesser General Public License for more details.

\*

\* You should have received a copy of the GNU Lesser General Public

\* License along with this library; if not, write to the Free Software

\*

\* Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA

\*

\*

\*/

package info.smartkit.eip.test.reports.domain;

import java.util.ArrayList;

import java.util.Date;

import java.util.List;

public class Statistic {

private Date date;

private String name;

private float percentage;

private float average;

private float amount;

public static List dummy3 = new ArrayList();

static {

dummy3.add(new DummyLevel3("name1",new Long(1)));

dummy3.add(new DummyLevel3("name2",new Long(2)));

dummy3.add(new DummyLevel3("name3",new Long(3)));

dummy3.add(new DummyLevel3("name3",new Long(4)));

}

public Statistic() {

super();

}

public Statistic(Date date, String name, float percentage, float average, float amount) {

super();

this.date = date;

this.name = name;

this.percentage = percentage;

this.average = average;

this.amount = amount;

}

public Date getDate() {

return date;

}

public void setDate(Date date) {

this.date = date;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public float getPercentage() {

return percentage;

}

public void setPercentage(float percentage) {

this.percentage = percentage;

}

public float getAverage() {

return average;

}

public void setAverage(float average) {

this.average = average;

}

public float getAmount() {

return amount;

}

public void setAmount(float amount) {

this.amount = amount;

}

public List getDummy3() {

return dummy3;

}

}

package info.smartkit.eip.test.reports;

/\*

\* Copyright 2015 the original author or authors.

\*

\* Licensed under the Apache License, Version 2.0 (the "License");

\* you may not use this file except in compliance with the License.

\* You may obtain a copy of the License at

\*

\* http://www.apache.org/licenses/LICENSE-2.0

\*

\* Unless required by applicable law or agreed to in writing, software

\* distributed under the License is distributed on an "AS IS" BASIS,

\* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.

\* See the License for the specific language governing permissions and

\* limitations under the License.

\*

\* All rights reserved.

\*/

import java.util.ArrayList;

import java.util.HashMap;

import java.util.Map;

import net.sf.jasperreports.engine.JRException;

import net.sf.jasperreports.engine.JRExporterParameter;

import net.sf.jasperreports.engine.JasperCompileManager;

import net.sf.jasperreports.engine.JasperExportManager;

import net.sf.jasperreports.engine.JasperFillManager;

import net.sf.jasperreports.engine.data.JRBeanCollectionDataSource;

import net.sf.jasperreports.engine.export.JRXlsExporter;

import info.smartkit.eip.models.DataBeanList;

/\*\*

\* Application entry point

\*

\* @author yangboz

\*/

public class JasperReportFill

{

public static String getClassPath()

{

return "/Users/yangboz/Documents/Git/north-american-adventure/RushuMicroService/eip-rushucloud/src/main/resources/";

}

private final static String JASPER\_REPORT\_BASE = getClassPath() + "reports/jasper\_report\_template";

private final static String JRXML\_SOURCE\_FILE = JASPER\_REPORT\_BASE + ".jrxml";

private final static String JASPER\_DEST\_FILE = JASPER\_REPORT\_BASE + ".jasper";

@SuppressWarnings("unchecked")

public static void main(String[] args) throws JRException

{

// JasperCompileManager.compileReportToFile(JRXML\_SOURCE\_FILE, JASPER\_DEST\_FILE);

String jasperDestFile = JasperCompileManager.compileReportToFile(JRXML\_SOURCE\_FILE);

//

String printFileName = null;

DataBeanList DataBeanList = new DataBeanList();

ArrayList dataList = DataBeanList.getDataBeanList();

JRBeanCollectionDataSource beanColDataSource = new JRBeanCollectionDataSource(dataList, true);

Map parameters = new HashMap();

long start = System.currentTimeMillis();

try {

printFileName = JasperFillManager.fillReportToFile(JASPER\_DEST\_FILE, parameters, beanColDataSource);

if (printFileName != null) {

/\*\*

\* 1- export to PDF

\*/

JasperExportManager.exportReportToPdfFile(printFileName, JASPER\_REPORT\_BASE + ".pdf");

/\*\*

\* 2- export to HTML

\*/

JasperExportManager.exportReportToHtmlFile(printFileName, JASPER\_REPORT\_BASE + ".html");

/\*\*

\* 3- export to Excel sheet

\*/

JRXlsExporter exporter = new JRXlsExporter();

exporter.setParameter(JRExporterParameter.INPUT\_FILE\_NAME, printFileName);

exporter.setParameter(JRExporterParameter.OUTPUT\_FILE\_NAME, JASPER\_REPORT\_BASE + ".xls");

exporter.exportReport();

}

} catch (JRException e) {

e.printStackTrace();

}

System.err.println("JasperReport Filling time : " + (System.currentTimeMillis() - start) + printFileName);

}

}

/\*

\* DynamicJasper: A library for creating reports dynamically by specifying

\* columns, groups, styles, etc. at runtime. It also saves a lot of development

\* time in many cases! (http://sourceforge.net/projects/dynamicjasper)

\*

\* Copyright (C) 2008 FDV Solutions (http://www.fdvsolutions.com)

\*

\* This library is free software; you can redistribute it and/or

\* modify it under the terms of the GNU Lesser General Public

\*

\* License as published by the Free Software Foundation; either

\*

\* version 2.1 of the License, or (at your option) any later version.

\*

\* This library is distributed in the hope that it will be useful,

\* but WITHOUT ANY WARRANTY; without even the implied warranty of

\*

\* MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU

\*

\* Lesser General Public License for more details.

\*

\* You should have received a copy of the GNU Lesser General Public

\* License along with this library; if not, write to the Free Software

\*

\* Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA

\*

\*

\*/

package info.smartkit.eip.test.reports.utils;

import java.lang.reflect.InvocationTargetException;

import java.util.Comparator;

import java.util.List;

import org.apache.commons.beanutils.PropertyUtils;

import org.apache.commons.logging.Log;

import org.apache.commons.logging.LogFactory;

/\*\*

\* @author Alejandro Gomez

\* Date: Feb 26, 2007

\* Time: 10:27:32 AM

\*/

public class MultiPropertyComparator implements Comparator {

private static final Log LOGGER = LogFactory.getLog(MultiPropertyComparator.class);

private List info;

/\*\*

\*

\* @param \_info List<SortInfo>

\*/

public MultiPropertyComparator(final List \_info) {

info = \_info;

}

public int compare(final Object \_o1, final Object \_o2) {

int result = 0;

for (int i = 0; result == 0 && i < info.size(); i++) {

final SortInfo sortInfo = (SortInfo)info.get(i);

try {

final String propertyName = sortInfo.getPropertyName();

final Comparable value1 = getValue(\_o1, propertyName);

final Comparable value2 = getValue(\_o2, propertyName);

result = compare(value1, value2) \* (sortInfo.isAscending() ? 1 : -1);

} catch (IllegalAccessException ex) {

LOGGER.warn(ex);

} catch (InvocationTargetException ex) {

LOGGER.warn(ex);

} catch (NoSuchMethodException ex) {

LOGGER.warn(ex);

}

}

return result;

}

private static Comparable getValue(final Object \_object, final String \_field) throws NoSuchMethodException, IllegalAccessException, InvocationTargetException {

final Object value = PropertyUtils.getProperty(\_object, \_field);

if (value == null)

return null;

if (!(value instanceof Comparable))

throw new RuntimeException("Objects are not Comparable, class " + value.getClass().getName());

return (Comparable) value;

}

private static int compare(final Comparable \_value1, final Comparable \_value2) {

if (\_value1 == null) {

if (\_value2 == null) {

return 0;

} else {

return -1;

}

} else if (\_value2 == null) {

return 1;

} else {

return \_value1.compareTo(\_value2);

}

}

}

/\*

\* DynamicJasper: A library for creating reports dynamically by specifying

\* columns, groups, styles, etc. at runtime. It also saves a lot of development

\* time in many cases! (http://sourceforge.net/projects/dynamicjasper)

\*

\* Copyright (C) 2008 FDV Solutions (http://www.fdvsolutions.com)

\*

\* This library is free software; you can redistribute it and/or

\* modify it under the terms of the GNU Lesser General Public

\*

\* License as published by the Free Software Foundation; either

\*

\* version 2.1 of the License, or (at your option) any later version.

\*

\* This library is distributed in the hope that it will be useful,

\* but WITHOUT ANY WARRANTY; without even the implied warranty of

\*

\* MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU

\*

\* Lesser General Public License for more details.

\*

\* You should have received a copy of the GNU Lesser General Public

\* License along with this library; if not, write to the Free Software

\*

\* Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA

\*

\*

\*/

package info.smartkit.eip.test.reports.utils;

/\*\*

\* @author Alejandro Gomez

\* Date: Feb 26, 2007

\* Time: 10:27:04 AM

\*/

public class SortInfo {

private String field;

private boolean ascending;

public SortInfo(final String \_field, final boolean \_ascending) {

field = \_field;

ascending = \_ascending;

}

public String getPropertyName() {

return field;

}

public boolean isAscending() {

return ascending;

}

}