# Notifications

Сервис состоит из двух частей: сервиса и API для вызова удаленных методов сервиса

## Развертывание сервиса

Требования

* jdk1.7
* Tomcat8
* Cервер RabbitMQ
* Сервер MySQL

### Настройка

Переименовывать notify.war в notify.zip
Заполннить файлы ./WEB-INF/classes/META-INF/staging.properties и ./WEB-INF/classes/META-INF/production.properties внутри архива данными для staging и production контуров.

Заполнить параметры:

view\_checker\_url={url приложения доступный снаружи}/viewchecker/

unsubscribe\_url={url приложения доступный снаружи}/unsubscribed/

jdbc.jdbcUrl={url до базы данных}

jdbc.user={аккаунт базы данных с правами создавать таблицы, создавать, читать, редактировать, удалять записи в таблицах}

jdbc.password={пароль аккаунта базы данных}

mail.host={адрес SMTP сервера, через который будет идти "белая" рассылка - ожидаемые сообщения}

mail.port={порт SMTP сервера, через который будет идти "белая" рассылка - ожидаемые сообщения}

spam.mail.host={адрес SMTP сервера, через который будет идти "серая" рассылка - коммерческие предложения, которые некоторые пользователи могут пометить как спам}

spam.mail.port={порт SMTP сервера, через который будет идти "серая" рассылка - коммерческие предложения, которые некоторые пользователи могут пометить как спам}

rabbit.hostname={адрес сервера RabbitMQ}

rabbit.username={аккаунт для доступа к серверу RabbitMQ}

rabbit.password={пароль для доступа к серверу RabbitMQ}

Переимновать notify.zip в в notify.war

Ппри запуске tomacat использовать флаг -Dspring.profiles.active=staging для staging контура и -Dspring.profiles.active=production для production контура.

Произвести dsrkflre notify.war в Tomcat, в контекст notify

## API

Сервис, который захочет выполнять рассылку через notificator, должен использовать API notifications-api-0.34.jar и реализовать java-конфиг вида:

@Configuration

public class TestConfig {

 @Bean

 public ConnectionFactory rabbitConnectionFactory() {

 CachingConnectionFactory connectionFactory = new CachingConnectionFactory(raabitHost);

 connectionFactory.setUsername(rabbitUserName);

 connectionFactory.setPassword(rabbitPassword);

 return connectionFactory;

 }

 @Bean

 public AmqpAdmin rabbitAdmin() {

 return new RabbitAdmin(rabbitConnectionFactory());

 }

 @Bean

 public MessageConverter messageConverter() {

 return new Jackson2JsonMessageConverter();

 }

 @Bean

 public AmqpTemplate rabbitTemplate() {

 RabbitTemplate rabbitTemplate = new RabbitTemplate(rabbitConnectionFactory());

 rabbitTemplate.setMessageConverter(messageConverter());

 rabbitTemplate.setReplyTimeout(60\*1000);

 return rabbitTemplate;

 }

 @Bean

 public ApiProvider apiProvider() {

 ApiProvider apiProvider = new ApiProvider();

 apiProvider.setExchange(NotificationsApi.EXCHANGE\_NAME);

 return apiProvider;

 }

 @Bean

 public NotificationsApi notificationApi() {

 return apiProvider().getApi(NotificationsApi.class);

 }

}

Вызов удаленных методов осуществляется по-примеру:

public class TestConsumer {

 @Autowired private NotificationsApi notificationsApi;

 public void myMethod() {

 NotificationDto notificationDto = new NotificationDto();

 NotificationResponse notificationResponseWhite = sendMessage(notificationDto);

 List<NotificationDto> notificationDtoList = new ArrayList<NotificationDto>();

 List<NotificationResponse> notificationResponseWhiteList = notificationsApi.sendMessages(notificationDtoList)

 SpamMessageDto grayMessageDto = new SpamMessageDto();

 List<NotificationResponse> notificationResponseGrayList = sendSpamMessage(grayMessageDto);

 List<NotificationTagsRequest> requests = new ArrayList<NotificationTagsRequest>();

 List<String> requestsIds = findScheduledByTags(requests);

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

 boolean isExists = existsScheduledByTags(tags);

 void rejectNotificationsByTags(tags);

 Date date = new Date();

 void rejectNotificationsByTagsAndDate(tags, date);

 }

}