

Инструкция по установке
Системы управления корпоративным
риском «Нейронавтика»

КОНСАЛТИКА

Москва, 2020

Содержание

1. Минимальные требования к аппаратному обеспечению	3
2. Требования к программному обеспечению	3
3. Инструкция по установке программного обеспечения	3
4. Описание процесса настройки конфигурационных файлов	6
4.1. Конфигурационный файл context.xml	6
4.2. Параметры веб-приложения web-app.properties	6
4.3. Конфигурационный файл app.properties.....	7
4.3.1. Параметры Ядра (backend) app.properties	7
4.3.2. Конфигурирование Backend for Frontend	8
4.4. Пример конфигурации	9
4.4.1. Пример конфигурации для продуктивной среды	9
5. Варианты развертывания программного обеспечения.....	10
5.1. Оба блока развернуты на одном сервере.....	10
6. Настройка СУБД	13

1. Минимальные требования к аппаратному обеспечению

Модуль	CPU Cores	RAM GB	HDD GB
Backend for Frontend + Backend	4	8	20
DataBase	4	8	10

2. Требования к программному обеспечению

1. Unix-подобные ОС (рекомендуется RedHat/CentOS 7/Astra Linux);
2. PostgreSQL версии 8.4+, свободная объектно-реляционная СУБД с открытым исходным кодом, распространяемая под лицензией PostgreSQL (<http://postgresql.org/ftp/source/>);
3. OpenJDK версии 1.8, свободное программное обеспечение с открытым исходным кодом, распространяемое под лицензией GNU GPL 2 (<https://openjdk.java.net/>);
4. Apache Tomcat версии 8.5+, свободное программное обеспечение с открытым исходным кодом, распространяемое под лицензией Apache License 2.0 (<https://tomcat.apache.org/download-80.cgi>).

3. Инструкция по установке программного обеспечения

1. Загрузить архив с экземпляром ПО

Загрузить архив с экземпляром ПО с использованием полученной от нас ссылки на архив с экземпляром ПО.

Разархивировать загруженный экземпляр ПО

Разархивировать загруженный экземпляр в любую директорию на жестком диске компьютера (сервера), на котором будет осуществляться установка экземпляра ПО, с использованием пароля.

2. Установить Java (OpenJDK) на сервер RedHat/CentOS версии 7

2.1. Открыть командную строку и выполнить команду:

```
yum -y install java-1.8.0-openjdk.x86_64 java-1.8.0-openjdk-devel.x86_64
```

3. Настроить Java (OpenJDK):

3.1. Проверить директорию, в которую установлена Java:

```
sudo update-alternatives --config java
```

3.2. Открыть файл `/etc/environment` с помощью команды `nano`:

```
nano /etc/environment
```

и добавить строку

```
JAVA_HOME="/usr/lib/jvm/java-1.8.0-openjdk-1.8.0.111-1.b15.e17_2.x86_64/jre"
```

3.3. Открыть файл профиля пользователя:

```
nano ~/.bash_profile
```

и добавить в него строки:

```
export JAVA_HOME=/usr/lib/jvm/java-1.8.0-openjdk-1.8.0.111-1.b15.e17_2.x86_64/jre
export PATH=$JAVA_HOME/bin:$PATH
```

3.4. Перезагрузить профиль пользователя:

```
source ~/.bash_profile
```

3.5. Убедиться в корректной установке Java с помощью команды `echo $JAVA_HOME`

4. Установить TomCat на сервер CentOS

4.1. Открыть командную строку и создать группу для установки веб-сервера с помощью команды

```
groupadd tomcat
```

4.2. Добавить пользователя с соблюдением требований безопасности:

```
useradd -s /bin/false -g tomcat -d /opt/tomcat tomcat
```

4.3. Скачать непосредственно установочный архив TomCat, выполнив команды:

```
cd /opt/
```

```
wget http://mirror.wanxp.id/apache/tomcat/tomcat-8/v8.5.6/bin/apache-tomcat-8.5.6.tar.gz
```

4.4. Распаковать архив и переименовать директорию для упрощения работы с помощью команд:

```
tar -xzvf apache-tomcat-8.5.6.tar.gz
mv apache-tomcat-8.5.6/* tomcat/
```

4.5. Установить владельца и группу с помощью команд:

```
chown -hR tomcat:tomcat tomcat
```

4.6. Проверить корректность установки с помощью запуска файла `startup.sh`:

```
cd /opt/tomcat/bin/
./startup.sh
```

4.7. Запустить следующую команду (TomCat использует порт по умолчанию 8080):

```
netstat -plntu
```

4.8. Убедиться, что на экране в одной строке **8080** и **Java** находятся в одной строке – это означает, что установка произведена корректно.

5. Связать Apache и TomCat для завершения установки Apache TomCat

5.1. Создать в системной директории новый файл конфигурации:

```
cd /etc/systemd/system/  
nano tomcat.service
```

5.2. Вставить в данный файл следующую информацию:

```
[Unit]  
Description=Apache Tomcat 8 Servlet Container  
After=syslog.target network.target  
[Service]  
User=tomcat  
Group=tomcat  
Type=forking  
Environment=CATALINA_PID=/opt/tomcat/tomcat.pid  
Environment=CATALINA_HOME=/opt/tomcat  
Environment=CATALINA_BASE=/opt/tomcat  
ExecStart=/opt/tomcat/bin/startup.sh  
ExecStop=/opt/tomcat/bin/shutdown.sh  
Restart=on-failure  
[Install]  
WantedBy=multi-user.target
```

5.3. Сохранить файл, выйти и перезапустить сервисы с помощью команд:

```
systemctl daemon-reload  
systemctl start tomcat  
systemctl enable tomcat
```

5.4. Проверить корректность работы

```
systemctl status tomcat
```

6. Поместить файлы `demo.war` и `demo-core.war` в папку `tomcat/webapps/`, после чего они автоматически распакуются веб-сервером TomCat.

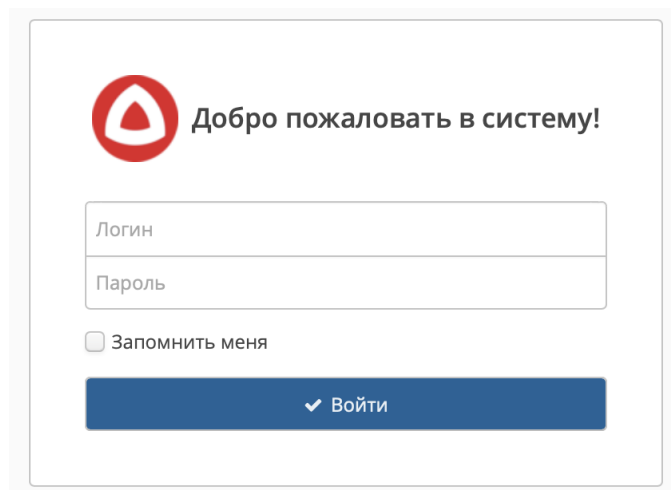
7. Настроить конфигурационные файлы `context.xml`, `web-app.properties` и `app.properties` (описано в разделе 4 инструкции)

8. Перезапустить веб-сервер TomCat с помощью команды:

```
sudo systemctl restart tomcat
```

9. Открыть веб-браузер и перейти, написать в адресной строке `localhost:8080` и нажать **Enter**

На экране браузера по адресу `http://localhost:8080` должно появиться окно входа в Систему (как показано на скриншоте ниже):



4. Описание процесса настройки конфигурационных файлов

4.1. Конфигурационный файл context.xml

Файл context.xml является дескриптором развертывания приложения на сервере **Apache Tomcat**. В развернутом приложении этот файл располагается в подкаталоге META-INF каталога веб-приложения или WAR-файла, например, tomcat/webapps/app-core/META-INF/context.xml.

Основное предназначение файла для блока Middleware - определить JDBC источник данных и поместить его в JNDI под именем.

Пример определения источника данных для **PostgreSQL**:

```
<Resource
  name="jdbc/CubaDS"
  type="javax.sql.DataSource"
  driverClassName="org.postgresql.Driver"
  username="dbuser"
  password="dbpassword"
  url="jdbc:postgresql://localhost/sales"/>
```

где name – имя источника данных, driverClassName – имя класса, username – имя пользователя, password – пароль, url – строка подключения

4.2. Параметры веб-приложения web-app.properties

По назначению свойства приложения можно классифицировать следующим образом:

Конфигурационные параметры - задают наборы конфигурационных файлов и некоторые параметры пользовательского интерфейса, т.е. определяют функциональность приложения. Значения конфигурационных параметров обычно задаются при разработке приложения.

Параметры развертывания - различные URL для соединения блоков приложения, тип используемой БД, настройки безопасности и т.д. Значения параметров развертывания обычно зависят от окружения, в котором устанавливается данный экземпляр приложения.

Параметры времени выполнения - активность аудита, параметры отсылки email и т.д. Параметры времени выполнения могут быть изменены при необходимости во время работы приложения без его перезапуска.

Значения свойств приложения могут быть заданы в базе данных, в файлах свойств, через системные свойства Java или переменные окружения ОС. Если свойство с некоторым именем задано в нескольких источниках, его значение определяется в следующем приоритете:

1. Системное свойство Java (высший приоритет)
2. Переменная окружения ОС
3. Файл свойств
4. База данных (низший приоритет)

Полный список параметров конфигурационного файла https://doc.cuba-platform.com/manual-7.2-ru/app_properties_reference.html

4.3. Конфигурационный файл `app.properties`

4.3.1. Параметры Ядра (backend) `app.properties`

- `cuba.dbmsType=postgres`
`cuba.dbmsType` - задает тип СУБД.
- `cuba.webContextName=operrisk-core`
`cuba.webContextName` - конфигурационный параметр, задающий имя контекста веб-приложения. Как правило, эквивалентен имени каталога или WAR-файла, содержащего данный блок приложения.
Используется в блоках Middleware, Web Client, Web Portal.
- `cuba.availableLocales=Russian|ru;English|en`
Список поддерживаемых языков интерфейса.
Формат свойства:
`{название_языка1}|{код_языка_1};{название_языка2}|{код_языка_2};`
 - `{название_языка}` – это название, которое будет отображаться в списках доступных языков. Например, в окне входа в систему, в экране редактирования пользователя.

- {код_языка} – соответствует коду, возвращаемому методом `Locale.getLanguage()`. Используется как суффикс для формирования имен файлов пакетов сообщений. Например, `messages_fr.properties`
- `bpm.activiti.asyncExecutorEnabled=true`
Возможные значение: `true` или `false`. Определяет, включен ли Job executor для выполнения задач таймеров и асинхронных задач. По умолчанию значение равно `false`.
- `cuba.webAppUrl=http://localhost:9090/operrisk`
URL, по которому доступен Web Client приложения.
Используется, в частности, для формирования ссылок на экраны приложения извне, а также классом `ScreenHistorySupport`.
- `cuba.restApiUrl=http://localhost:9090/operrisk-portal/api`
URL, по которому Backend for Frontend обращается к Backend (ядру) платформы.
- `cuba.security.minimalRoleIsDefault=false`
- `cuba.dataSourceProvider=jndi`
Способ получения источника данных задается данным свойством приложения: его значение может быть либо `application`, либо `jndi`.
 - Значение `application` указывает, что источник данных должен быть сконфигурирован свойствами приложения.
 - Значение `jndi` указывает, что источник данных должен быть получен из JNDI.

4.3.2. Конфигурирование Backend for Frontend

- `cuba.connectionUrlList=http://localhost:9090/operrisk-core`
Задает список URL для подключения клиентских блоков к серверам Middleware. Значением свойства должен быть один или несколько разделенных запятой URL вида `http[s]://host[:port]/app-core`, где `host` - имя сервера, `port` - порт сервера, `app-core` - имя веб-приложения, реализующего блок Middleware.
- `cuba.useLocalServiceInvocation=true`
При установке данного свойства в `true` блоки Web Client и Web Portal вызывают сервисы Middleware в обход сетевого стека, что положительно сказывается на производительности системы. Это возможно в случае быстрого развертывания, единого WAR и единого Uber JAR. В других вариантах развертывания данное свойство необходимо установить в `false`.
- `cuba.webContextName=operrisk`
Конфигурационный параметр, задающий имя контекста веб-приложения. Как правило, эквивалентен имени каталога или WAR-файла, содержащего данный блок приложения.

- `cuba.httpSessionExpirationTimeoutSec=1800`
Задает таймаут бездействия HTTP-сессии в секундах.
- `cuba.web.foldersPaneEnabled=true`
Включает отображение панели папок для экрана и использование горячих клавиш в папках.
- `cuba.availableLocales=Russian|ru;English|en`
Список поддерживаемых языков интерфейса.
Формат свойства:
`{название_языка1}|{код_языка_1};{название_языка2}|{код_языка_2};`
 - `{название_языка}` – это название, которое будет отображаться в списках доступных языков.
 - `{код_языка}` – соответствует коду, возвращаемому методом `Locale.getLanguage()`. Используется как суффикс для формирования имен файлов пакетов сообщений.
- `cuba.localeSelectVisible=true`
Включает или отключает возможность пользователя выбрать язык интерфейса при входе в систему.
Если `cuba.localeSelectVisible=false`, то локаль пользовательской сессии выбирается следующим образом:
 - если для данного экземпляра сущности `User` установлен атрибут `language`, то устанавливается локаль для этого языка;
 - если язык операционной системы пользователя присутствует в списке доступных (заданных свойством `cuba.availableLocales`), то выбирается он;
 - в противном случае выбирается язык, заданный первым в свойстве `cuba.availableLocales`.
- `cuba.themeConfig=com/haulmont/cuba/havana-theme.properties`
Задает набор файлов `*-theme.properties`, в которых описаны переменные тем, такие как размеры диалоговых окон и ширина полей ввода по умолчанию.

4.4. Пример конфигурации

4.4.1. Пример конфигурации для продуктивной среды

```
cuba.web.appFoldersRefreshPeriodSec=30
cuba.rest.client.id=rest_client
cuba.rest.client.secret={noop}rest_client
cuba.rest.apiUrl=http://localhost:9090/operrisk-portal/api
cuba.rest.securityScope=GENERIC_UI
```

```
cuba.rest.productionMode=true
cuba.web.productionMode=true
cuba.web.resourcesCacheTime=0
```

3.3.2. Пример конфигурации для LDAP

```
cuba.web.requirePasswordForNewUsers=false
cuba.web.ldap.enabled=true
cuba.web.ldap.urls=ldap://192.168.101.2:389
cuba.web.ldap.base=OU=Staff,OU=Domain
Users,DC=mycompany,DC=local
cuba.web.ldap.user=basil.holl@mycompany.local
cuba.web.ldap.password=Qawsed123
cuba.web.standardAuthenticationUsers=admin
```

```
cuba.web.appFoldersRefreshPeriodSec=30
cuba.rest.client.id=rest_client
cuba.rest.client.secret={noop}rest_client
cuba.rest.apiUrl=http://localhost:9090/operrisk-portal/api
cuba.rest.securityScope=GENERIC_UI
```

```
cuba.rest.productionMode=true
cuba.web.productionMode=true
cuba.web.resourcesCacheTime=0
```

5. Варианты развертывания программного обеспечения

5.1. Оба блока развернуты на одном сервере

В данном случае обеспечивается максимальная производительность передачи данных между блоками **Web Client** и **Middleware**, так как при включенном свойстве приложения `Cuba.useLocalServiceInvocation` сервисы **Middleware** вызываются в обход сетевого стека.

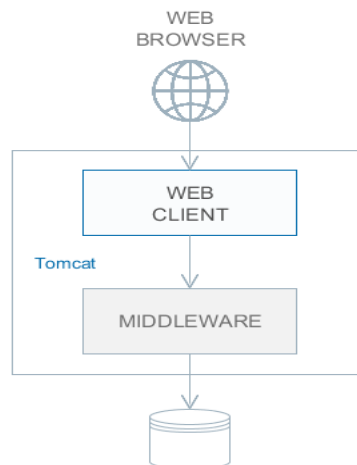


Рис.1 Блоки Middleware и Web Client развернуты на отдельных серверах приложения.

Данный вариант позволяет распределить нагрузку между двумя серверами приложения и более оптимально использовать ресурсы серверов. Кроме того, в этом случае нагрузка от веб-пользователей меньше сказывается на выполнении других процессов. Под другими процессами здесь понимается обслуживание средним слоем других типов клиентов, выполнение задач по расписанию и, возможно, интеграционные задачи.

Требования к ресурсам серверов:

- Tomcat 1 (Web Client):
 - Объем памяти - пропорционально количеству одновременно подключенных пользователей.
 - Мощность CPU - зависит от интенсивности работы пользователей.
- Tomcat 2 (Middleware):
 - Объем памяти - фиксированный и относительно небольшой.
 - Мощность CPU - зависит от интенсивности работы пользователей и других процессов.

В этом и более сложных вариантах развертывания в блоке Web Client свойство `cuba.useLocalServiceInvocation` должно быть установлено в `false`, а свойство `cuba.connectionUrlList` должно содержать URL блока Middleware.

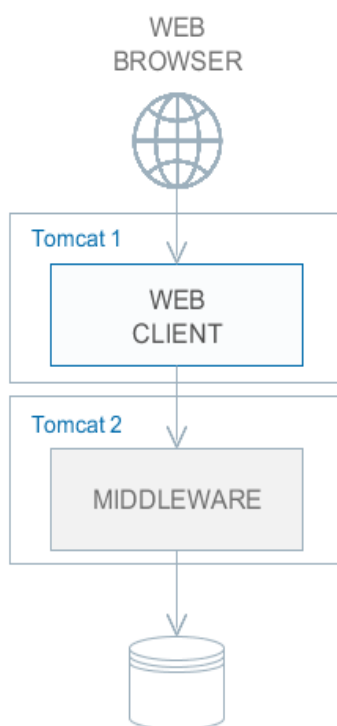


Рис.2 Кластер серверов Web Client работает с одним сервером Middleware.

Данный вариант применяется, когда вследствие большого количества одновременно подключенных пользователей требования к памяти для блока Web Client превышают возможности одной JVM. В этом случае запускается кластер (два или более) серверов Web Client, и подключение пользователей производится через Load Balancer. Все серверы Web Client работают с одним сервером Middleware.

Дублирование серверов Web Client автоматически обеспечивает отказоустойчивость на этом уровне. Однако, так как репликация HTTP-сессий не поддерживается, при незапланированном отключении одного из серверов Web Client все пользователи, подключенные к нему, вынуждены будут выполнить новый логин в приложение.

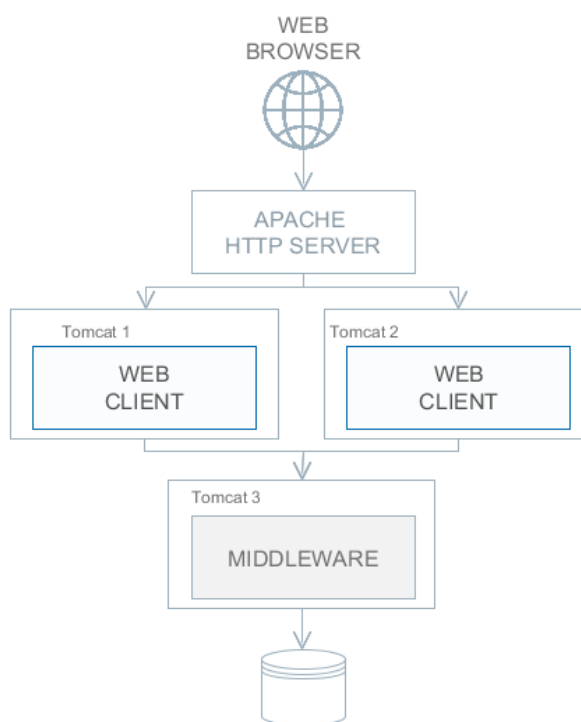


Рис.3 Кластер серверов Web Client работает с кластером серверов Middleware.

Это максимальный вариант развертывания, обеспечивающий отказоустойчивость и балансировку нагрузки для Middleware и Web Client.

Подключение пользователей к серверам Web Client производится через Load Balancer. Серверы WebClient работают с кластером серверов Middleware. Для этого им не требуется дополнительный Load Balancer - достаточно определить список URL серверов Middleware в свойстве `cuba.connectionUrlList`. Можно также использовать дополнение для интеграции с Apache ZooKeeper для динамического обнаружения серверов среднего слоя.

В кластере серверов Middleware организуется взаимодействие для обмена информацией о пользовательских сессиях, блокировках и пр. При этом обеспечивается полная отказоустойчивость блока Middleware - при отключении одного из серверов выполнение запросов от клиентских блоков продолжается на доступном сервере прозрачно для пользователей.

6. Настройка СУБД

Тип используемой СУБД определяется свойствами приложения `Cuba.dbmsType` и (опционально) `cuba.dbmsVersion`, которые влияют на поведение механизмов, зависящих от типа базы данных.