**Архитектура и конфигурирование**

Приложение homeless-report-engine — это стандартное J2EE web приложение. В качестве сервера приложений может использоваться любой контейнер сервлетов, но рекомендуется Apache Tomcat 7.0.47 и выше.

Приложение представляет собой вынесенную отдельно функциональность по генерированию справок, контрактов и отчетов для основного приложения homeless. Это повышает удобство использования и отладки, а также позволяет использовать функциональность в отдельных скриптах. Frontend отсутствует, поддерживаются лишь HTTP GET запросы.

В данный момент приложение не поддерживает аутентификацию, однако, это будет исправлено (необходимо поддержать глобальную авторизацию на tomcat или обеспечить SSO).

Приложение построено на стандартной платформе Spring, имеет общий контроллер для доступа к сервлетам, генерирует документы MS Office с помощью Apache POI и выдает их запросившему по HTTP GET. Вся необходимая информация для генерации документов берется из http request (передается в качестве параметров в строке GET запроса), а также из базы данных (используется та же база данных, что и для основного приложения homeless).

Конфигурация базы данных вынесена в конфигурацию сервера приложений (на примере Apache Tomcat 7). Для этого в нем необходимо в server.xml определить глобальный ресурс базы данных, в context.xml указать маппинг для использования в веб приложениях и в spring-config.xml приложения указать соответствующий JNDI ресурс.

В директорию lib Apache Tomcat 7 необходимо положить заранее скачанный mysql connector свежей версии.

В качестве примера (взятого из http://www.codingpedia.org/ama/tomcat-jdbc-connection-pool-configuration-for-production-and-development/), ниже представлены стандартные параметры ресурсов:

<Resource

name="jdbc/ProductionDB"

auth="Container"

type="javax.sql.DataSource"

factory="org.apache.tomcat.jdbc.pool.DataSourceFactory"

initialSize="34"

maxActive="377"

maxIdle="233"

minIdle="89"

timeBetweenEvictionRunsMillis="34000"

minEvictableIdleTimeMillis="55000"

validationQuery="SELECT 1"

validationInterval="34000"

testOnBorrow="true"

removeAbandoned="true"

removeAbandonedTimeout="55"

username="homeless"

password="homeless"

driverClassName="com.mysql.jdbc.Driver"

url="jdbc:mysql://localhost:3306/homeless?useUnicode=true&amp;characterEncoding=UTF-8"

/>

или

<Resource

name="jdbc/DevelopmentDB"

auth="Container"

type="javax.sql.DataSource"

factory="org.apache.tomcat.jdbc.pool.DataSourceFactory"

initialSize="5"

maxActive="55"

maxIdle="21"

minIdle="13"

timeBetweenEvictionRunsMillis="34000"

minEvictableIdleTimeMillis="55000"

validationQuery="SELECT 1"

validationInterval="34"

testOnBorrow="true"

removeAbandoned="true"

removeAbandonedTimeout="233"

username="homeless"

password="homeless"

driverClassName="com.mysql.jdbc.Driver"

url="jdbc:mysql://localhost:3306/homeless?useUnicode=true&amp;characterEncoding=UTF-8"

/>

Как можно видеть, мы определяем само подключение, пользователя и запускаем фабрику подключений, которая не даст приложению оторваться при неактивности пользователей (достичь wait\_timeout) и будет периодически посылать валидационный запрос в базу данных (SELECT 1), чтобы приложение отвечало в любое время дня и ночи, даже после новогодних праздников). На базе самого приложения это построить можно только с помощью подключения дополнительных фабрик или запуска keep alive потоков, в чем нет смысла.

Понятное дело, нам нужен только ProductionDB ресурс, поэтому добавляем в Production только его.

В context.xml необходимо определить ResourceLink для доступа из нашей Spring конфигурации:

<ResourceLink name="DevelopmentDB" global="jdbc/DevelopmentDB" type="javax.sql.DataSource"/>

или

<ResourceLink name="ProductionDB" global="jdbc/ProductionDB" type="javax.sql.DataSource"/>

Соответственно, в spring-context.xml необходимо определить

<jee:jndi-lookup jndi-name="DevelopmentDB" id="DataSource" expected-type="javax.sql.DataSource" />

или

<jee:jndi-lookup jndi-name="ProductionDB" id="DataSource" expected-type="javax.sql.DataSource" />

После чего конфигурацию доступа Tomcat к MySQL и работы приложения с MySQL можно считать завершенной.

Конфигурирование приложения относительно путей к шаблонам и директории с контрактами можно произвести, изменив файл application.properties внутри задеплоенного приложения при выключенном сервере приложений.

На данный момент существют следующие параметры конфигурации

**templatesDir** = абсолютный локальный путь на сервере к директории с шаблонами

**contractsDir** = абсолютный локальный путь на сервере к директории со сгенерированными контрактами

Имена файлов указываются только на этапе сборки в интерфейсе IDocumentMapping, если это необходимо (должны быть веские основания).

**Сборка и деплоймент**

Исходный код приложения в данный момент хостится на https://maxxon@bitbucket.org/maxxon/homeless-report-engine (для доступа обратиться к разработчику)

Сборка осуществляется стандартным Maven 3.0.5 (другие версии не тестировались). В случае необходимости изменить имена файлов шаблонов, перед сборкой необходимо изменить их в интерфейсе IDocumentMapping.

Сборка стандартная, через mvn clean package. В результате в target имеем собранный WAR, который можно отдеплоить на production.

Для деплоймента на Tomcat необходимо его остановить (можно использовать hot деплоймент, но это не тестировалось и, если требуются корректировки в application.properties, то их следует внести в WAR заранее). Рекомендуется переименовать директорию logs со старым логами и перезоздать ее пустую (чтобы отделить старые логи от новых).

Из webapps снести старое приложение homeless-report-engine (при необходимости, забэкапить настройки) и снести WAR, если он там присутствует.

После этого распаковать стандартным unzip новый WAR в директорию webapps или оставить его там не распакованным (будет распакован при первом запуске Tomcat).

В первом случае просто отконфигурировать приложение и запустить Tomcat. В случае, когда WAR не распаковывали, то запустить Tomcat и дождаться в логах сообщения о том, что приложение запущено (проверить через http что оно отвечает), затем остановить Tomcat, отконфигурировать приложение и снести уже не нужный WAR и снова запустить Tomcat.

После деплоймента убедиться что приложение позволяет успешно обратиться к существующим контрактам и сгенерировать любую справку (можно воспользоваться тестом, описанным на стандартной странице приложения).