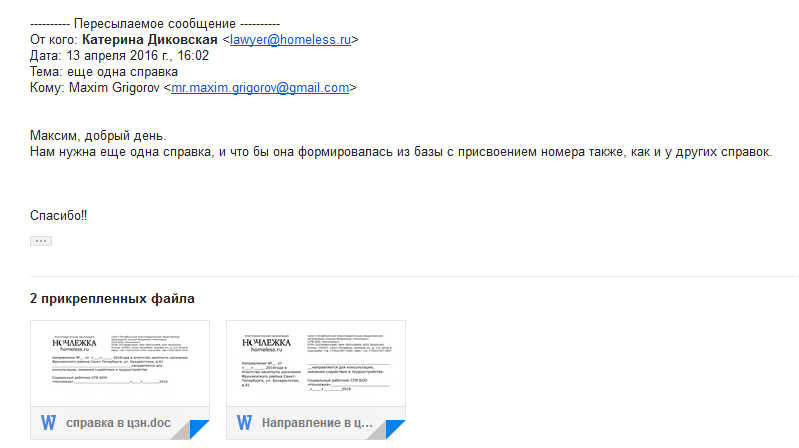
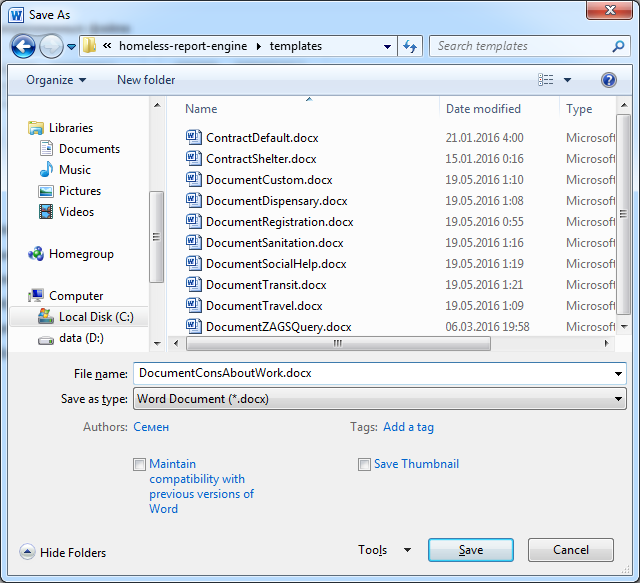
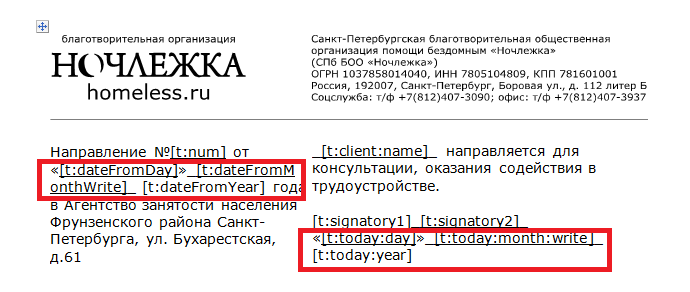
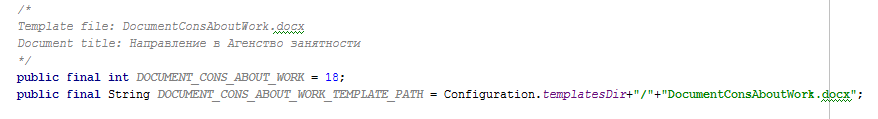
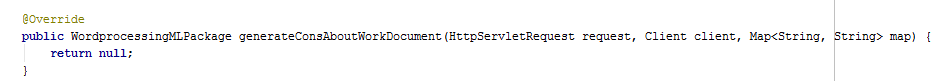
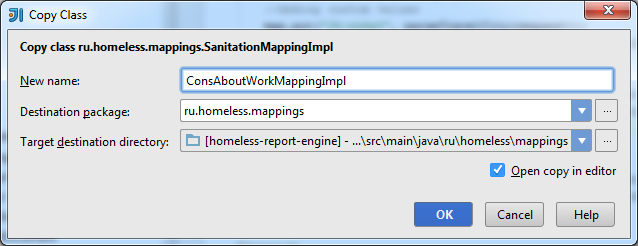
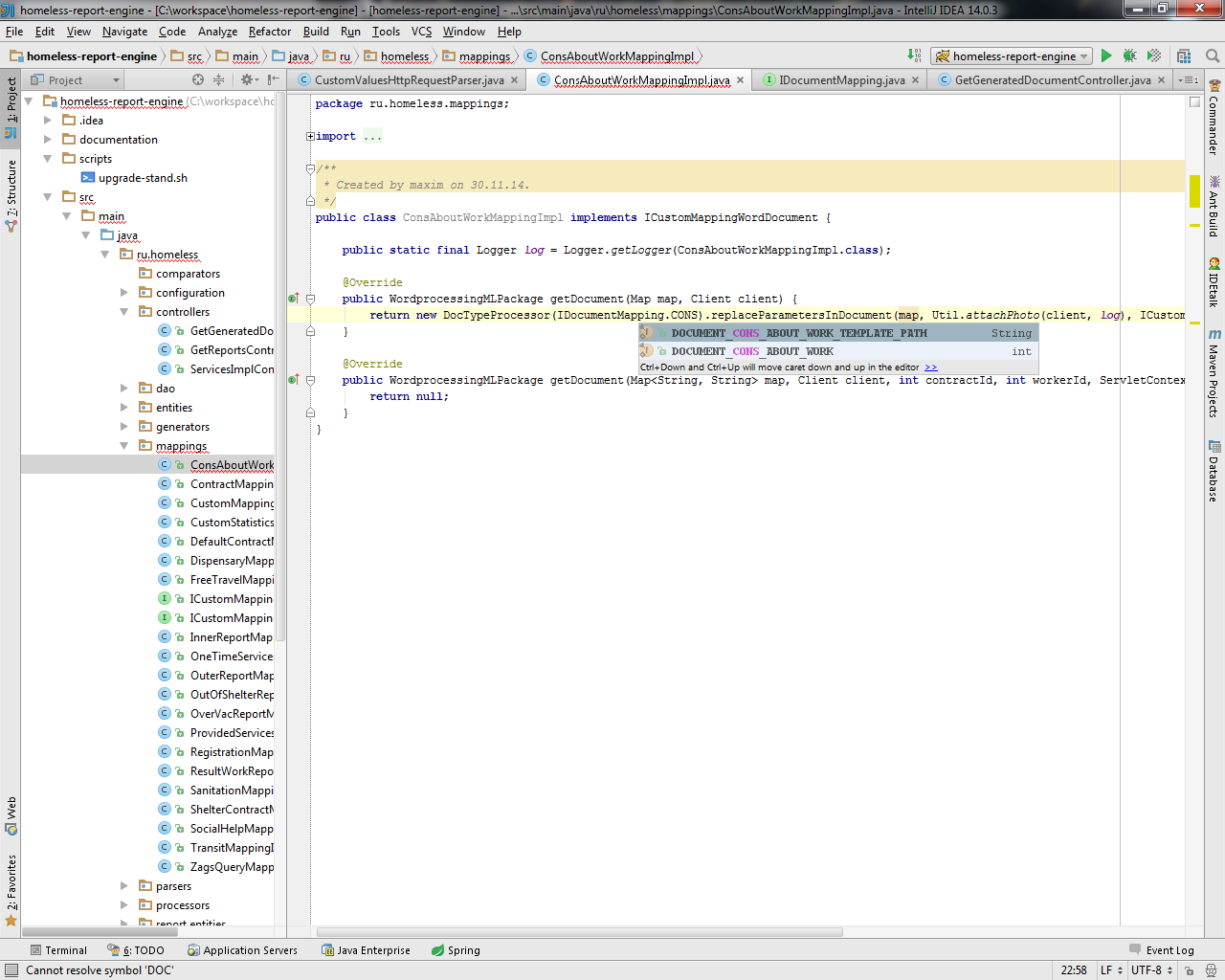
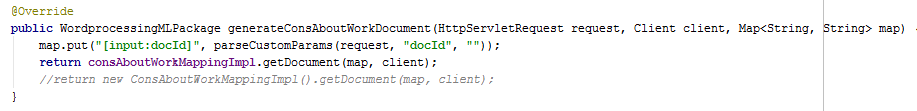
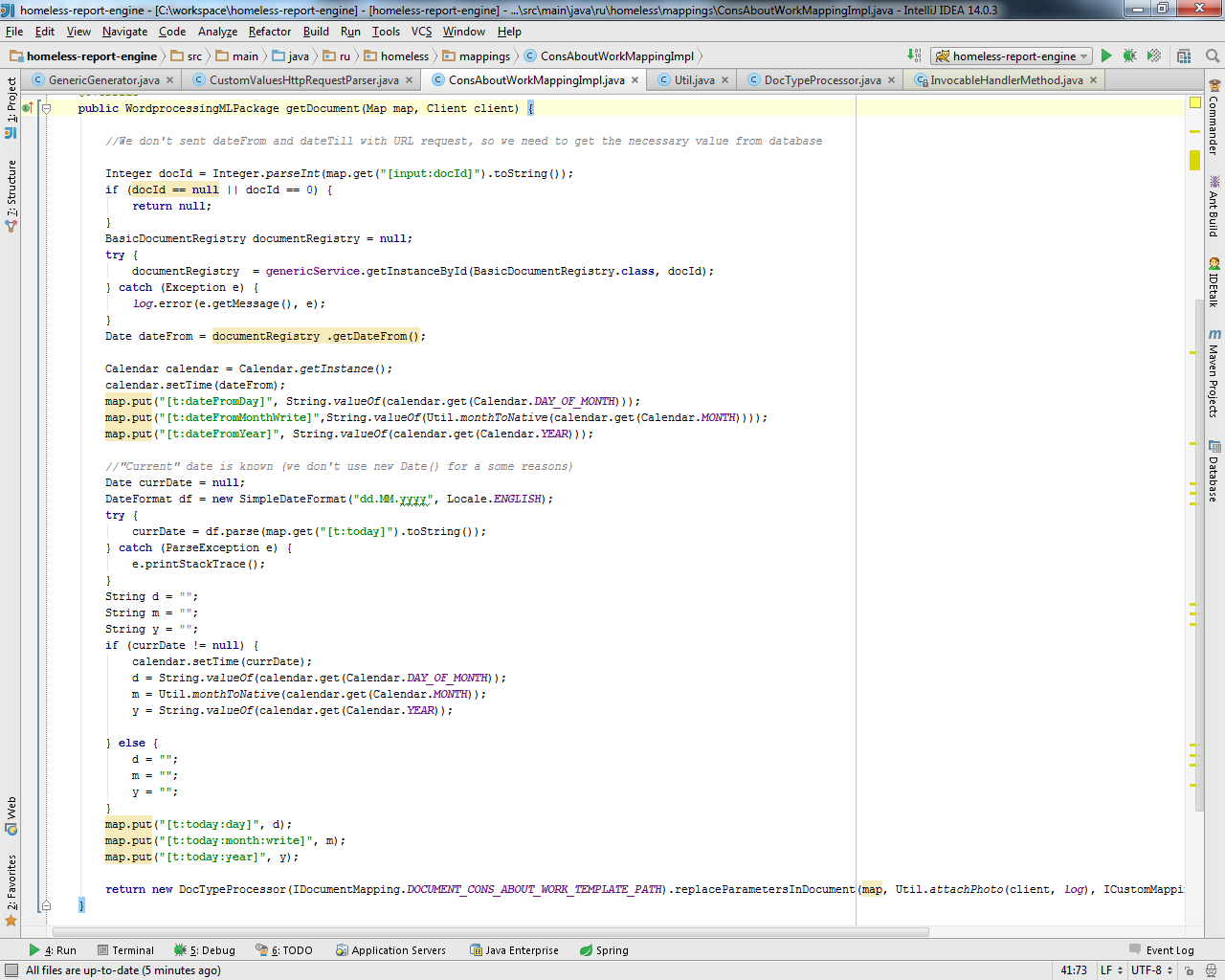
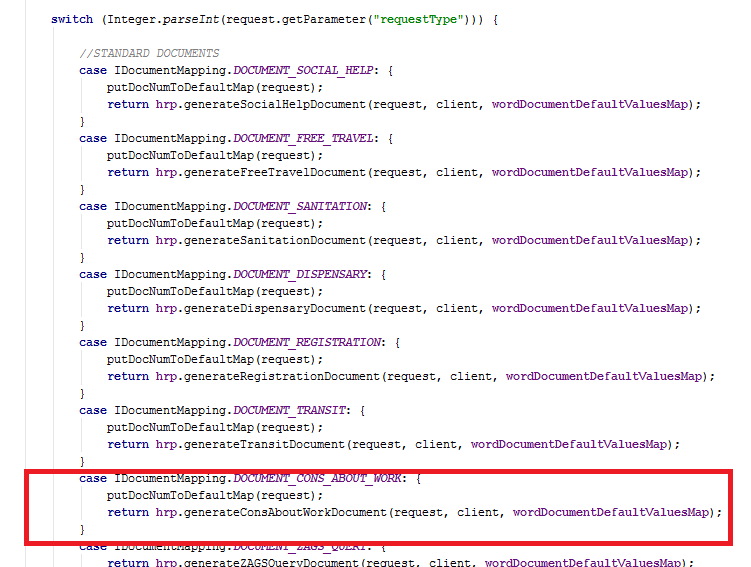
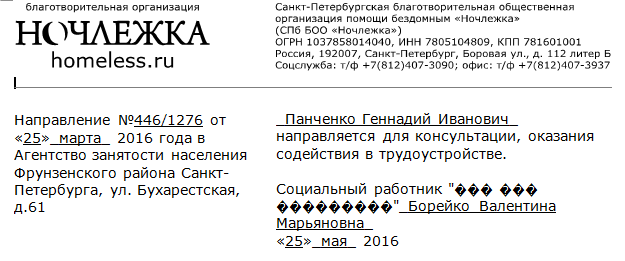
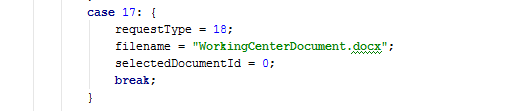
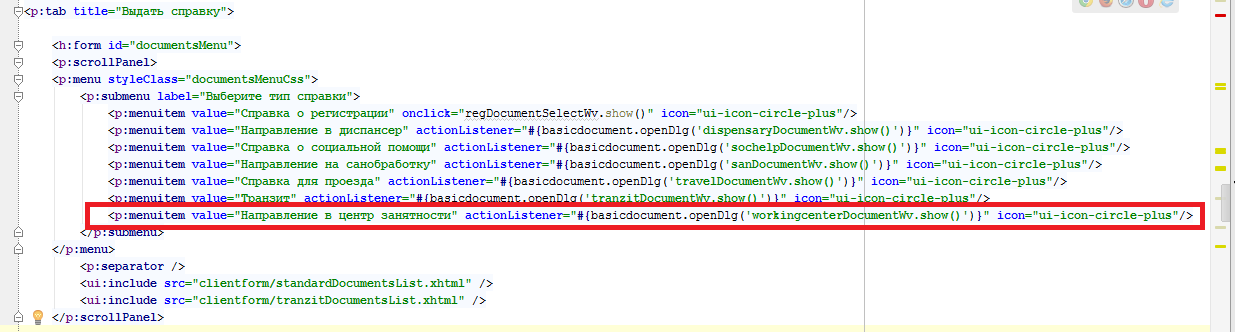
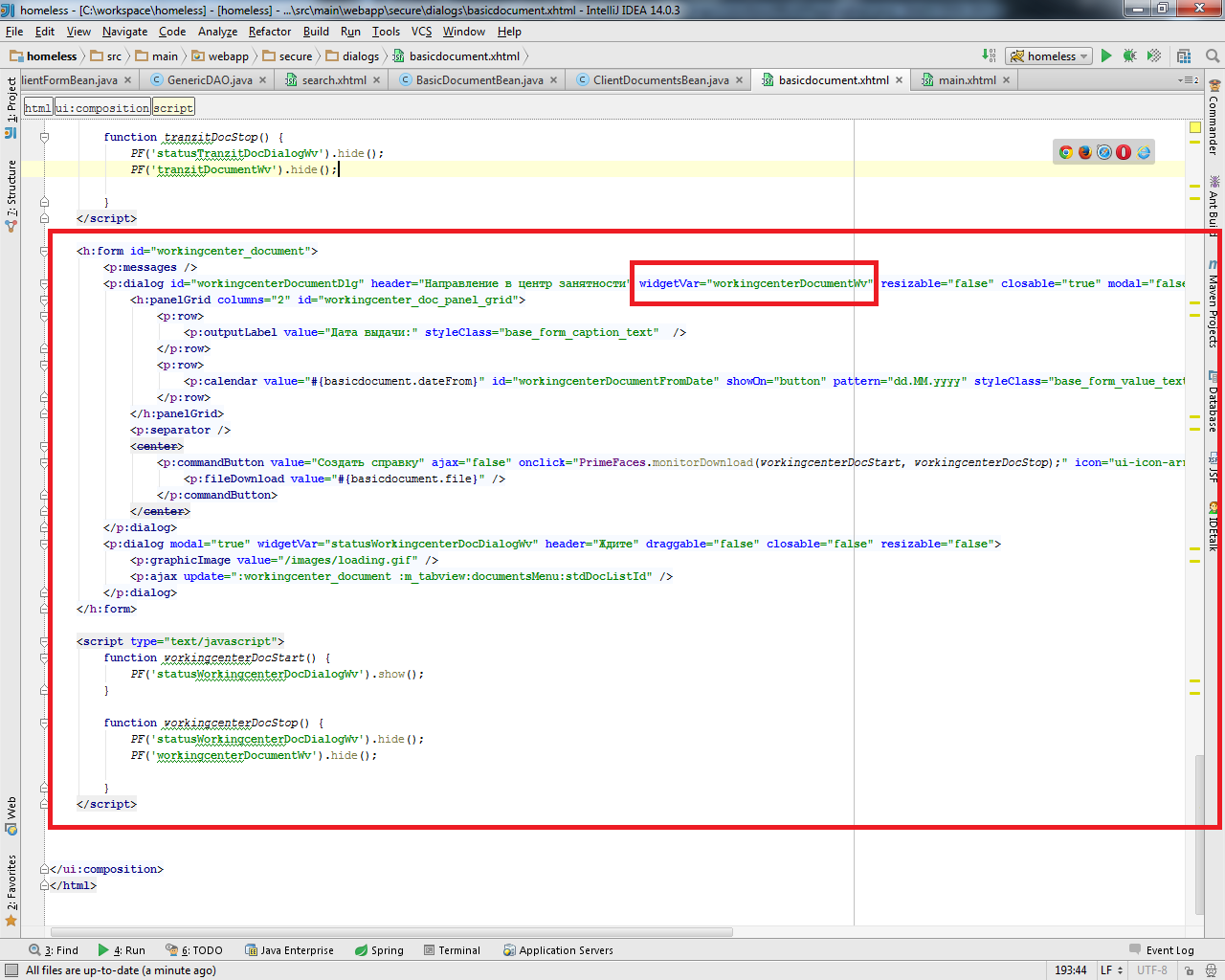
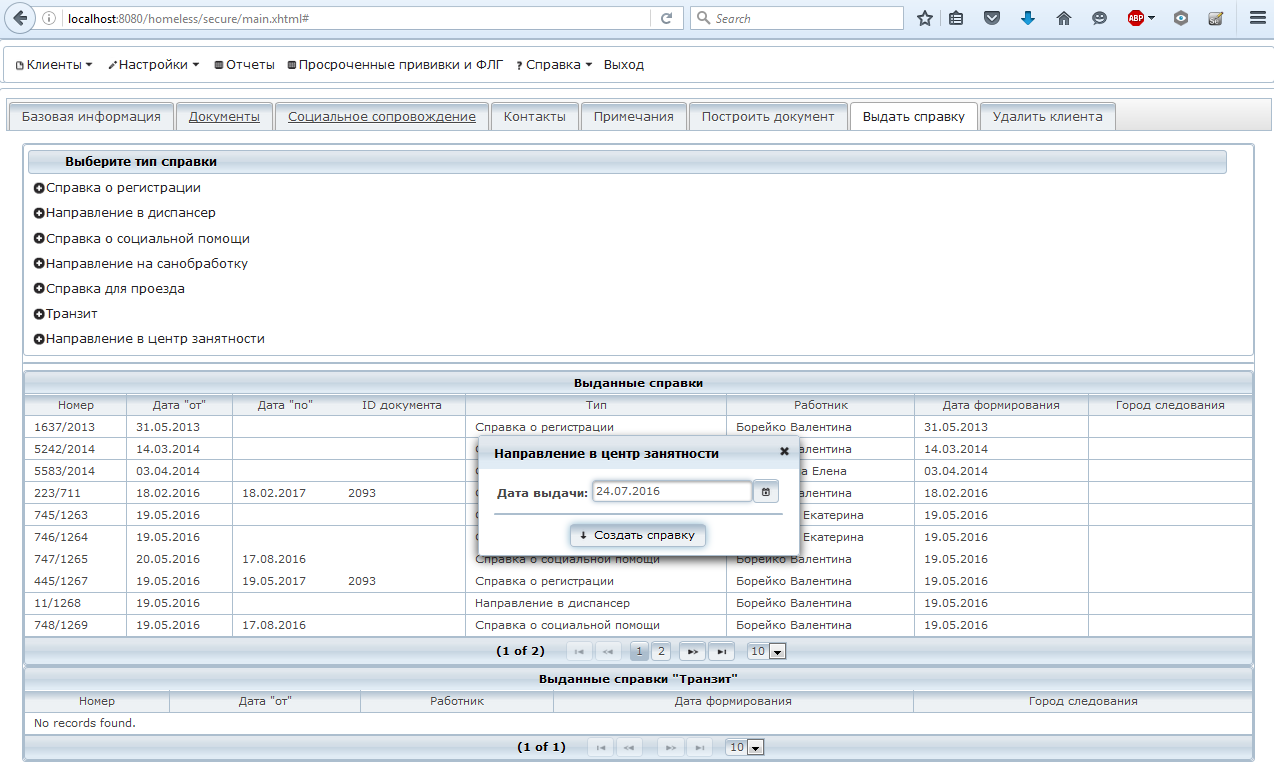
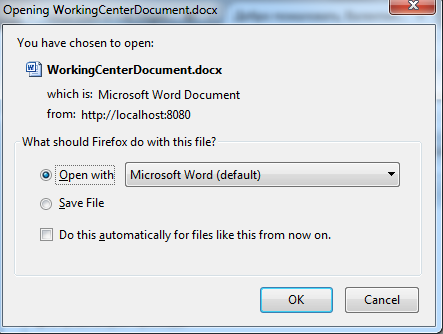
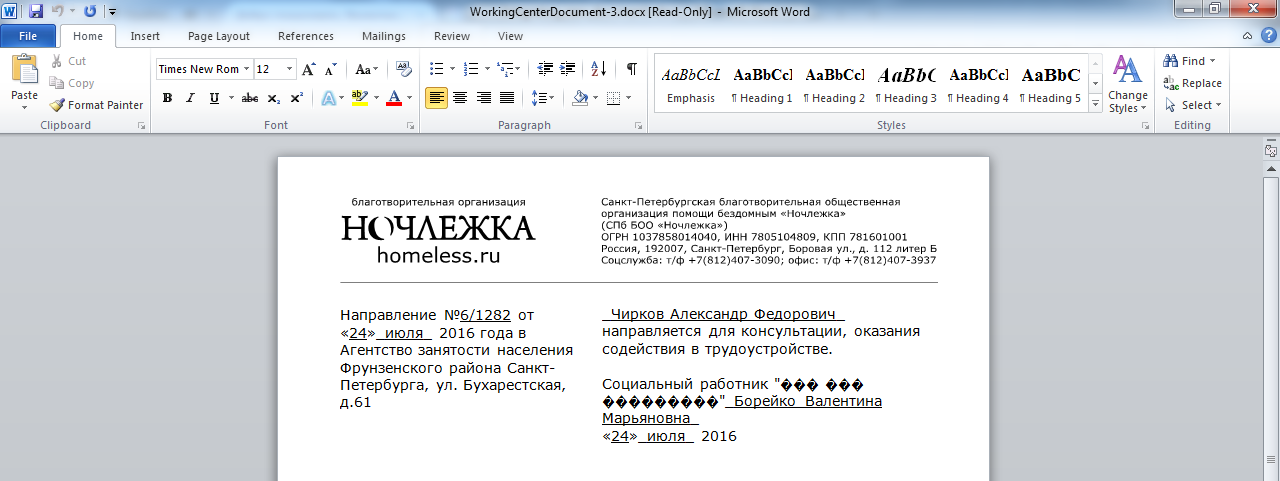
Интеграция новых справок в систему

1. Имеем запрос от кого-то из Ночлежки в YouTrack или через почту. Пример ниже:  
     
   
2. Сначала готовим шаблон. Идем в проект homeless-report-engine. Обновляем на актуальное состояние. Директория templates содержит всякие шаблоны.
3. Копируем файлы, полученные от Ночлежки, предварительно сохранив их как docx, если они в doc формате.  
   
4. Редактируем шаблон, указав существующие плейсхолдеры или добавив новые. Рекомендуется использовать существующие. Например,  
   [t:num] – номер справки в стандартно принятной нумерации  
   [t:photo] – фотография клиента  
   [t:client:name] – имя клиента  
   [t:signatory1] – стандартная левая часть подписи  
   [t:signatory2] – стандартная правая часть подписи  
   [t:client:birth] – дата рождения клиента  
   clientWhereWasBorn – место рождения клиента (без квадратных скобок)  
   [t:client:document] – документ, на основании которого выдается справка  
   [t:reg:date] – дата регистрации по документу  
   [t:dateFrom]– дата начала действия справки  
   [t:dateTill] – дата окончания действия справки  
   [t:date] или [t:today]– дата выдачи справки  
   [t:city] – куда едет клиент  
     
   Пример (помечены красным те плейсхолдеры, которых еще нет, созданы в процессе создания этого документа):  
   
5. Увы, практика показала, что даже если отключена проверка орфографии, Word генерит всякую ересь внутри наших плейсхолдеров и она мешает реплейсу параметров. Поскольку мы не используем API Office, а, фактически, правим голый XML и перепаковываем архив с набором XML (docx – это в чистом виде ZIP файл со всякими файлами внутри). В файле Word\_Template\_Preparation.docx описано как править шаблон так, чтобы после этого его можно было использовать.
6. Далее интегрируем шаблон в движок отчетов. Сначала описываем его в главном интерфейсе IDocumentMapping  
   Там же внизу определяем имплементирующий метод для генерации документа  
   
7. Далее вспоминаем что работа с этим движком начинается с контроллера, затем глубже. Идем в главный контроллер для справок GetGeneratedDocumentController, находим метод для справок getGeneratedWordDocument и спускаемся по классам вглубь.  
     
   Следующим будет метод GenericGenerator:generateWordDocument. Сначала он парсит нестандартные параметры, переданные через URL (CustomValuesHttpRequestParser). Проваливаемся в его конструктор и видим что IDEA ругается что не имплементирован новый метод. Имплементируем его, сделав стаб.  
     
     
     
   Поскольку для каждого типа документа мы имеем свою реализацию, нам нужно ее создать, прежде чем двигаться дальше. Идем в пакет mappings и создаем новый класс, который будет нам генерировать новую справку. Проще всего скопировать любой другой справочный класс \*Impl, например, SanitationMappingImpl  
     
     
   В скопированном файле правим путь к шаблону  
     
   Не забываем пометить аннотацией @Component наш новый класс, иначе его будет не заинжектить куда надо.  
   Все остальное можно оставить по умолчанию.  
     
   Возвращаемся в CustomValuesHttpParser и правим наш стаб:  
     
   Если мы будем работать внутри нового класса с заинжектированными бинами, нам необходимо заинжектить его самого в CustomValuesHttpParser  
     
     
     
   Если ничего из базы не потребуется доставать, то инжектить новый класс сюда не нужно и просто делаем так, как написано в закомментированной строчке.   
   Здесь можем положить в мапу кастомные параметры из реквеста. Дальше реквест будет недоступен и на следующем шаге будем класть только те значения в мапу, которые сделали из уже доступных или из базы данных.  
     
     
   Далее вспоминаем что у нас есть новые параметры. Они не передаются снаружи, мы их создаем прямо в коде. Также думаем: нужны ли они еще для других справок потенциально? Если нет, то в новом классе рисуем такой код:  
     
     
     
   Если все же нужны, то идем в GenericGenerator:putDefaultValuesInMap и делаем это там.
8. Далее, если вернемся в GenericGenerator:generateWordDocument мы видим как готовится мапа со стандартными параметрами и затем в зависимости от типа реквеста мы генерим тот или иной документ. Добавляем новый.  
   
9. Считаем что интегрировали справку. Запускаем и тестируем:  
     
   Поскольку мы еще не внедрили в основное приложение вызов нашей генерации, то сымитируем вызов. Правда, если у нас используется стандартная нумерация и какие-то параметры из базы данных (BasicDocumentRegistry), то надо что-то там создать. Например, можно создать справку о регистрации, затем переиспользовать эту строку из базы в наших целях.  
   Вот пример: я выдал справку о регистрации (requestType=12) для клиента clientId=13000, его документ (основание для выдачи справки) docId=15660, issueDate – текущая дата, итд итп. А в базу также занеслись параметры from и till (срок действия справки). Их и будем доставать.  
     
   Далее я вижу в логе вызов URL для генерации, меняю там requestType на новый и voila, у меня сгенерированная новая справка.  
     
   <http://localhost:9090/homeless-report-engine/getGeneratedWordDocument?requestType=18&clientId=13000&docId=15660&issueDate=25.05.2016&workerId=2&docNum=446/1276&travelCity>=  
     
   (порт может отличаться)  
     
     
   (корявые буквы – причина небольшого несоответствия кодировок, на production сервере будет все в порядке)
10. Ок, теперь надо это интегрировать в основное приложение. Тут все намного проще.  
      
    Открываем исходники приложения homeless  
    Также открываем MySQL Workbench и коннектимся к локальной отладочной базе  
      
    Делаем  
      
    *use homeless;*

*select \* from BasicDocumentRegistryType;*  
Видим различные типы справок. Под новым номером добавляем новую справку (номер любой, например, сейчас это 17.  
  
*insert into BasicDocumentRegistryType (id, caption) values (17, 'Направление в центр занятности');*

1. Идем в код. BasicDocumentBean класс в методе export содержит уже все необходимое.   
     
   Добавляем блок кода для новой справки (selectedDocument = 0 потому что мы не предоставляем выбор документа (как для справки о регистрации)  
     
     
   17 здесь – это ID типа документа в базе, 18 – код справки в приложении homeless-report-engine (интерфейс IdocumentMapping)  
     
   На всякий случай, отмечу, что справки учитываются только в таблице BasicDocumentRegistry и в ReceivedServices не попадают. Эти таблицы не связаны.
2. Отлично. Теперь пора нарисовать пункт меню. Открываем main.xhtml. Где-то там перечислены все справки, добавляем новую.  
     
   
3. Она будет показывать новый диалог. Далее идем в basicdocument.xhtml. Там создаем новый блок (копируем похожий и меняем под новые имена и указываем id новой справки в нашем бине (здесь это 17)  
     
   
4. Проверяем что нет ошибок и запускаем основную аппликацию.  
     
     
     
     
     
   
5. Далее добавляем новый темплейт в Mercurial и коммитим и пушим изменения по обоим проектам. Ждем пока пройдут обе сборки и обновят демо стенд.   
     
   ***Добавляем новую строчку в BasicDocumetRegistryType в демо базу после ее разворачивания в процессе сборки***

Затем тестируем то же самое на демо стенде (<http://10.2.0.9/demo-homeless>)

1. Заливаем на прод (http://10.2.0.9/homeless).   
     
   ***Добавляем новую строчку в BasicDocumetRegistryType на базу, используемую для прода***  
     
   Тестируем что все работает и по пути ничего не поломали.

На самом деле все достаточно просто. Другие документы и отчеты интегрируются похожим образом, достаточно хотя бы один раз это проделать вручную.