**Servlet API.**

**Задание-4.**

**Генератор рекомендательных писем.**

Друзья, это единственное задание из текущей темы, которое потребует от Вас создания *«нестандартных»* сервлетов. Дело в том, что при разработке всех остальных задач от Вас в качестве ответа будет требоваться ***HTML документ***, записанный в ***текстовый*** поток вывода. Но только сегодня, проездом в нашем городе, с единственным концертом и только для Вас… Неповторимое и леденящее кровь шоу… Вы готовы удивляться?.. ***Запись бинарного ответа в байтовый поток вывода***!!! Ну а попутно мы рассмотрим еще два (а если пожелаете, то и три) практических момента, которые, возможно, пригодятся Вам в реальной работе. Хотя, нет, соврал – один из этих моментов ***точно не пригодиться***, так что я бы на Вашем месте выбирал три, чтобы больше получилось ☺.

В этой задаче от Вас потребуется создание автоматического генератора рекомендательных писем для наиболее трудолюбивых слушателей нашего курса. Вы уже знаете, что после того, как наш курс закончится, тем из Вас кто работал больше и лучше своих коллег, наряду с сертификатом будет выдано рекомендательное письмо, которое Вы сможете предъявить в качестве дополнительного, пусть и неявного, и не всегда эффективного, но все же свидетельства Вашего конкурентного преимущества.

К сожалению, практика показывает, что огромное желание Вашего преподавателя составить отзыв, который бы наиболее точно и ярко раскрывал ***все*** положительные стороны именно того человека, для которого этот отзыв предназначается (*ну а еще конечно три ведра разгильдяйства вместо совести*), приводит к тому, что процесс выдачи этого письма затягивается на недели и даже месяцы! В результате, слушатели, заслужившие эти рекомендации, нервничают и обижаются, менеджмент тренинг центра – злится, а преподавателю искренне стыдно, но он из последних сил скрывает это, и к него это прекрасно получается. В результате, ситуация с рекомендациями – приятным, казалось бы, моментом – иногда превращается для слушателей в переполненную ожиданиями пытку! В общем, плохая получается ситуация… Я бы сказал – совсем беда…

Так давайте же вместе, общими усилиями не допустим этого! Давайте, в конце концов, сами бороться за свое светлое будущее! Давайте сыграем в простую игру – Вы сами себе напишете ту рекомендацию, которую, по Вашему мнению, Вы заслуживаете. А в конце курса, если все будет именно так, как Вы написали сейчас – я Вам ее выдам ☺. Но есть одно маленькое условие – напечатать Ваше рекомендательное письмо должна будет Ваша программа ☺! По тексту – не волнуйтесь – если будет нужно, подкорректируем в соответствии со всеми Вашими пожеланиями. Тем более как Вы сейчас узнаете, Ваша программа должна будет поддерживать загрузку шаблонного текста из внешнего файла, в который будут только лишь добавляться необходимые элементы.

Итак, обо всем по порядку. В рамках текущего задания, Вам необходимо реализовать сервлет, который отобразит для пользователя форму со следующими компонентами:

|  |  |  |
| --- | --- | --- |
| **№** | **Назначение компонента** | **Тип компонента** |
| 1 | Фамилия слушателя | Текстовое поле |
| 2 | Имя слушателя | Текстовое поле |
| 3 | Отчество слушателя | Текстовое поле |
| 4 | Дата начала занятий | DatePicker |
| 5 | Дата окончания занятий | DatePicker |
| 6 | Список тем, изученных самостоятельно. |  |
| 7 | Тема курсового проекта | Многострочное поле ввода – TextArea |
| 8 | Оценка курсового проекта | Combobox со следующими вариантами:   * Отлично * Хорошо * Удовлетворительно |
| 9 | Фамилия преподавателя | Текстовое поле |
| 10 | Имя преподавателя | Текстовое поле |
| 11 | Отчество преподавателя | Текстовое поле |
| 12 | Упаковать в архив | CheckBox |
| 13 | Тип архива | Радио кнопки со значениями:   * ZIP * JAR |
| 14 | Сгенерировать письмо | Кнопка submit. |

Приведем некоторые комментарии к таблице компонентов.

**Во-первых**, не забывайте, пожалуйста, что любая качественная программа должна заботиться о валидации введенных пользователем значений. Я думаю, что мы уже достаточно поговорили с Вами о валидации данных в задаче №1, поэтому здесь ограничимся лишь короткими тезисами – валидация данных должна присутствовать в Вашем сервлете, и ее правила Вы должны продумать самостоятельно.

**Во-вторых**, пункт номер №6 – «Список тем, изученных самостоятельно» – не дает точного типа компонента. Я бы на Вашем месте рассматривал два варианта реализации:

* Либо ListBox с возможностью выбора нескольких строк
* Либо красивая таблица с темами, у которой в пером столбце расположены Checkbox-ы, которыми можно отметить необходимые пользователю темы.

И выбрал бы второй вариант – он красивее, нагляднее и удобнее для пользователя.

**В-третьих**, обратите, пожалуйста, внимание на еще одно важное требование для пункта №6. Список тем ***не должен*** располагаться в Вашем исходном коде! Вместо этого, все темы должны загружаться ***из внешнего*** текстового файла, что позволит пользователям сервлета в любой момент изменить список предлагаемых тем либо дополнить его.

**В-четвертых**, пункт №12 – «Упаковать в архив». Это CheckBox, который не отправляется на сервер для обработки. Он служит только лишь для клиентской логики, и, если он выбран, то радио кнопки из пункта №13 становятся активными (enabled). И при помощи этих кнопок, пользователь может указать, что он хочет упаковать получившийся файл с рекомендацией в один из доступных архивов – ZIP или JAR.

Вот здесь мы вплотную и подошли к результатам работы нашей системы. После того, как пользователь корректно заполняет форму и наживает кнопку «Сгенерировать письмо», управление передается сервлету, который:

1. Считывает шаблон рекомендательного письма из внешнего файла (подробнее об этом будет рассказано ниже).
2. Располагает в этом шаблоне все данные, которые он получил от формы, описанной выше.
3. Записывает полученный результат в текстовый файл.
4. И отправляет этот файл, как бинарную информацию по средствам байтового потока вывода.
5. В результате чего на клиентской стороне браузер открывает пользователю стандартное диалоговое окно, которое спрашивает у пользователя, что он хочет сделать с полученным файлом – открыть его одной из стандартный программ либо сохранить на файловой системе.

Можно сказать все – работа сделана. Но у нас еще остается несколько уточняющих моментов для обсуждения.

**Архивация файла с рекомендательным письмом.** Как Вы могли видеть выше, у нас есть требование – результирующий файл может архивироваться либо в ZIP формат, либо в JAR. Это, кстати и есть первый из тех ***практических моментов***, о которых мы говорили вначале. В общем-то, добавлен он был для того, чтобы Вы вспомнили Java SE и поработали с такими классами, как

java.util.zip.ZipEntry

java.util.zip.ZipOutputStream

java.util.jar.JarEntry

java.util.jar.JarOutputStream

Обратите внимание, все классы, приведенные выше, лежат в пакете java.util.\*, а значит, для их использования не нужно предпринимать никаких дополнительных усилий кроме импорта. А еще обратите внимание на то, что в архивы в Java – это потоки вывода. Соответственно, хотите записать файл – пишите в файл, хотите записать в архив – пишите в архив, не забывайте только, что в обоих случаях писать нужно в поток ☺.

**Формат выходного файла – практический момент №2.** Выше мы описали требование, что форматом выходного файла с рекомендательным письмом должен быть ***текстовый файл***. Но в реальном мире текстовые файлы Вы вряд ли будете отдавать пользователям. Они потребуют от Вас PDF-ов, Excel-ей, и прочих Word-ов! Так почему бы ни подготовится к их требованиям заранее? В результате получаем новую редакцию требования:

Файлом результата может быть, конечно, и текстовый формат, но лучше, если это будет файл в формате Microsoft Word.

И вот тут возникает вопрос – как этот Word файл создать? Ответ очень прост – для этого Вам потребуется использовать какой-либо сторонний API, например **Apache POI**, домашняя странница которого расположена по адресу <http://poi.apache.org/>. А дальше? А дальше сами и с гуглом – ведь это же не обязательное требование, а так – рекомендация ☺.

Ну и наконец, мы подошли к основному требованию данной работы:

Текст рекомендательного письма не должен быть жестко определен в Вашем исходном коде. Вместо этого, шаблон текста должен загружаться из внешнего текстового файла, и наполняться теми данными, которые сервлет получил в качестве параметров.

В результате имеем классический подход при генерировании разного рода отчетов – получая некий шаблонный текст, программа должна наполнить его актуальными данными. Ниже я приведу Вам текст шаблона, который Вы можете использовать, но помните, Вы работаете над своим будущим рекомендательным письмом – не пожалейте времени и придумайте тот текст, который наиболее точно и красочно подчеркнет все Ваши достоинства ☺. Итак текст рекомендательного письма может быть следующим:

Рекомендация.

**Петров Сидор Иванович** обучался на курсах по предмету «Java Enterprise Edition» в период с **1 января 2014 г.** по **1 декабря 2017 г.** За время обучения **Сидор** проявил большое желание стать специалистом по проектированию и разработке программного обеспечения.

К занятиям готовился ответственно. Всегда выполнял домашнее задание. Самостоятельно, с использованием дополнительных материалов, освоил темы не входящие в состав предлагаемого курса такие как:

**- Тема-1**

**- Тема-2**

**- Тема-3**

**- …**

и применил полученные знания, как при выполнении домашних заданий, так и во время разработки итогового проекта. Курсовой проект был выполнен по теме «**Тема курсового проекта**» и сдан с оценкой «**отлично**».

По стилю работы **Сидор** склонен к работе в команде, демонстрируя ответственность, внимательность к деталям, а так же способность быстро и чётко выражать свои мысли.

В целом **Сидор** характеризуется как прилежный слушатель и перспективный сотрудник. Я буду рад, если в будущем мы сможем работать не как «преподаватель-слушатель», но как коллеги.

Преподаватель **Сидоров Иван Петрович**

Обратите внимание, что в тексте выше все элементы, которые Ваш сервлет должен заменить данными из запроса, выделены ***полужирным курсивом***. И вот здесь возникает вопрос – ***как реализовать эту замену***?

Я предложу Вам один из наиболее распространенных вариантов решения подобных задач, но возможно у Вас на примете есть и свой собственный подход. В любом случае, это требование Вы можете реализовать как угодно, главное, чтобы оно было реализовано.

Суть подхода, который я хочу Вам предложить, заключается в следующем. В тексте шаблона в тех местах, в которые должны будут вставляться реальные данные, Вы располагаете так называемые ***тэги*** (иногда их еще называют якорями) – последовательность символов, которую сможет распознать Ваш сервлет. И вместо которой, он вставит в текст актуальную информацию – имя, фамилию, тему курсового проекта и т. д.

Важно следить за тем, чтобы текст самого тэга был как можно более специфическим. Это нужно для того, чтобы текст тэга для имени, например, не пересекался с одним из реальных имен, которое может потребоваться при подготовке рекомендации. Иначе у алгоритма может слегка «поехать крыша». Достигается эта специфичность, как правило, путем добавления символов, которые в реальных именах и фамилиях уж точно не встретятся – фигурные и угловые скобки, нижние подчеркивания, звездочки и т. д.

В результате, если я для имени слушателя, например, буду использовать тэг {STUDENT\_FIRST\_NAME}, а для темы курсового проекта – тэг {COURSE\_TASK\_NAME}, то текст моего шаблона получится следующим:

Рекомендация.

**{STUDENT\_LAST\_NAME} {STUDENT\_FIRST\_NAME} {STUDENT\_MIDDLE\_NAME}** обучался на курсах по предмету «Java Enterprise Edition» в период с **{START\_COURSE\_DATE}** по **{END\_COURSE\_DATE}** За время обучения **{STUDENT\_FIRST\_NAME}** проявил большое желание стать специалистом по проектированию и разработке программного обеспечения.

К занятиям готовился ответственно. Всегда выполнял домашнее задание. Самостоятельно, с использованием дополнительных материалов, освоил темы не входящие в состав предлагаемого курса такие как:

**{ADDITIONAL\_TOPIC\_NAMES}**

и применил полученные знания, как при выполнении домашних заданий, так и во время разработки итогового проекта. Курсовой проект был выполнен по теме «**{COURSE\_TASK\_NAME}**» и сдан с оценкой «**{COURSE\_TASK\_MARK}**».

По стилю работы **{STUDENT\_FIRST\_NAME}** склонен к работе в команде, демонстрируя ответственность, внимательность к деталям, а так же способность быстро и чётко выражать свои мысли.

В целом **{STUDENT\_FIRST\_NAME}** характеризуется как прилежный слушатель и перспективный сотрудник. Я буду рад, если в будущем мы сможем работать не как «преподаватель-слушатель», но как коллеги.

Преподаватель **{TUTOR\_LAST\_NAME} {TUTOR\_FIRST\_NAME} {TUTOR\_MIDDLE\_NAME}**

Уверен, что о функциональном назначении других тэгов Вы легко догадаетесь. Ну, все, наша проблема решена. Остаться решить проблему, которая появилась сразу же после решения предыдущей проблемы ☺. ***Как эффективно осуществить замену тэгов в шаблоне, чтобы получить результирующий текст рекомендательного письма***?

В этом случае вариантов решения уже гораздо больше, и Вы опять-таки сможете выбрать любой. Самый простой из них – читать текст шаблона построчно, и для каждой строки применять алгоритм, который посредствам простыни из условных операторов **if**, проверит – а не попался ли в только что прочитанной строке какой-нибудь тэг, который нужно заменить параметром из запроса? И если вдруг попался – производить замену.

Однако такой подход может сыграть с Вами злую шутку – если **if**-ов будет действительно много – есть большая вероятность того, что не примется такая работа.

Поэтому, пожалуйста, помните о том, что алгоритм замены тэгов в шаблоне рекомендательно письма – это, можно сказать основной объем работы по текущей задаче. Отнеситесь к нему со всей серьезностью. Подумайте о том, как сделать его простым для понимания Ваших коллег, удобным в плане расширения (добавления новых тэгов) и стабильным в работе.

Если позволите, я подскажу Вам одну из технологий, которую Java использует для того, чтобы парсить XML документы. Это **Java Simple API for XML (SAX)**. Главу из Java Tutorial-а по этой теме можно найти по следующей ссылке – <http://docs.oracle.com/javase/tutorial/jaxp/sax>. А в гугле можно найти бесконечность примеров и статей на тему того, как парсить XML документы при помощи SAX алгоритма на Java. Обратите внимание на то, что Java SAX API входит в стандартную поставку Java SE, так что никакие дополнительные библиотеки Вам не потребуются.

Так вот, Вы могли бы, изучив эту технологию, позаимствовать у нее основную идею, и на основе этой идеи реализовать свой парсер, который разберется с шаблоном Вашего рекомендательного письма. Неявным бонусом для Вас в этом случае станет знакомство с SAX-парсерами – довольно распространенной технологией, к слову сказать. Но еще раз – решать Вам ☺.

Все – почти закончили. Три последних требования в этой задаче остались. Первое просто, чтобы напомнить Вам, что:

Дополнительные темы, которые изучил слушатель и текст шаблона рекомендательного письма должны храниться во внешних файлах.

Более того:

Эти файлы не должны располагаться в Вашем приложении, они должны храниться в неком внешнем каталоге файловой системы, не принадлежащем серверу.

Ну а это, в свою очередь, рождает третье требование:

Путь к внешнему каталогу и имена файлов, как для тем, так и для шаблона, должны конфигурироваться при помощи *параметров инициализации* Вашего сервлета.

Обратите внимание, последнее требование в задаче – это и есть тот ***неявный практический момент №3***, который Вы ***опробуете сейчас***, и который, скорее всего, не будете применять на практике ***никогда*** – использование тэгов <init-param> при конфигурировании сервлета.

Почему никогда не будете использовать? Потому что это непрактичный и… *…и вот тут не придумывается никакое другое слово кроме как…* корявый подход, который никто в реальной работе не использует.

Почему его нужно попробовать сейчас? Потому что **параметры инициализации сервлета** – это одна их наиболее популярных теоретических тем, которая описывается во всех книгах по сервлетам в самом начале. Так почему бы ее и не попробовать тогда?

Ну, вот и все – все требования описаны. Остается только пожелать Вам успехов и вдохновения во время работы!