Руководство по созданию Web приложения на Java с использованием стека технологий Java EE

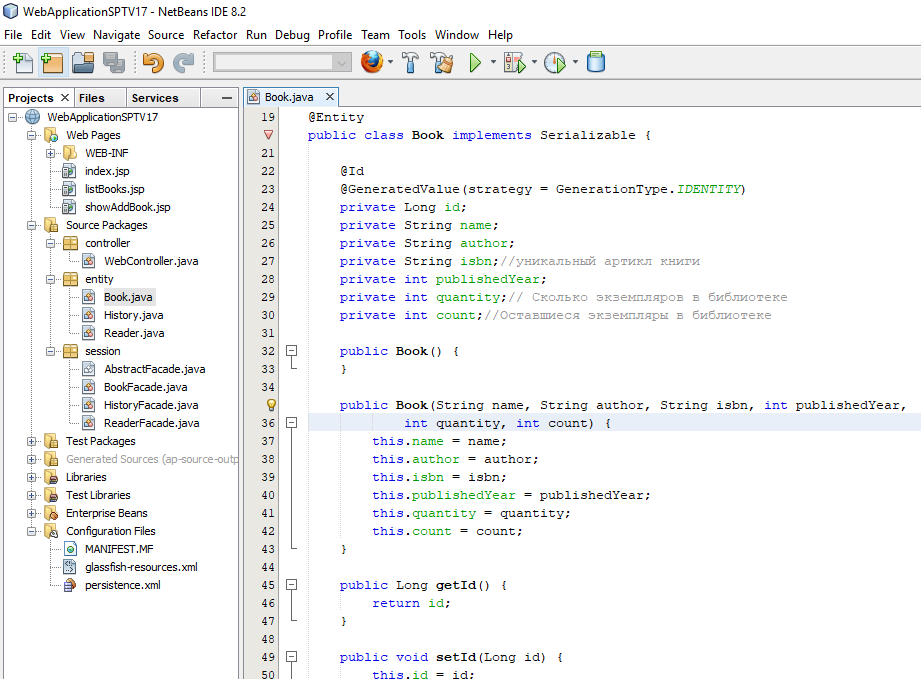
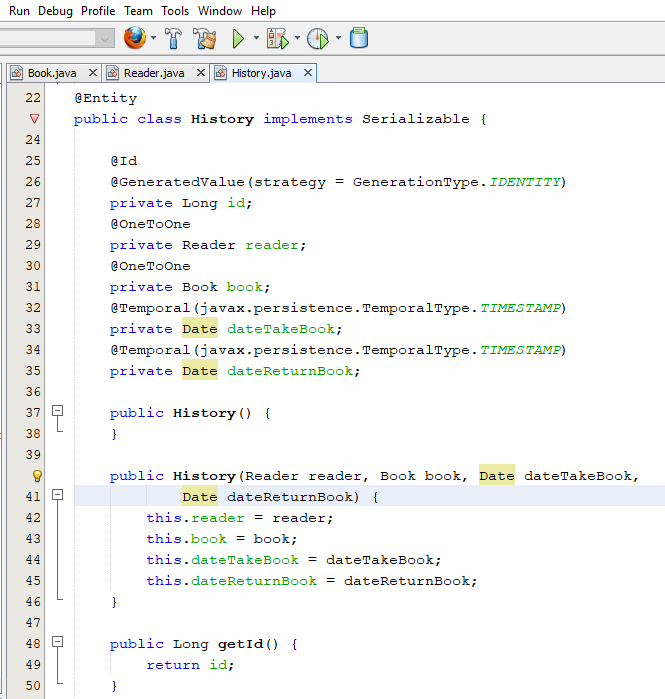
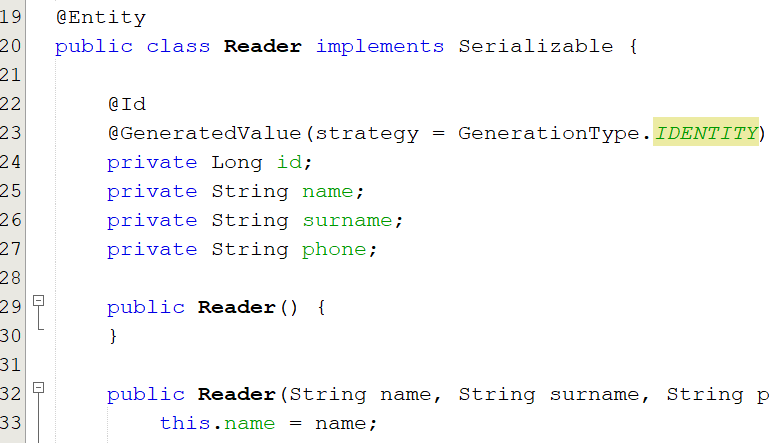
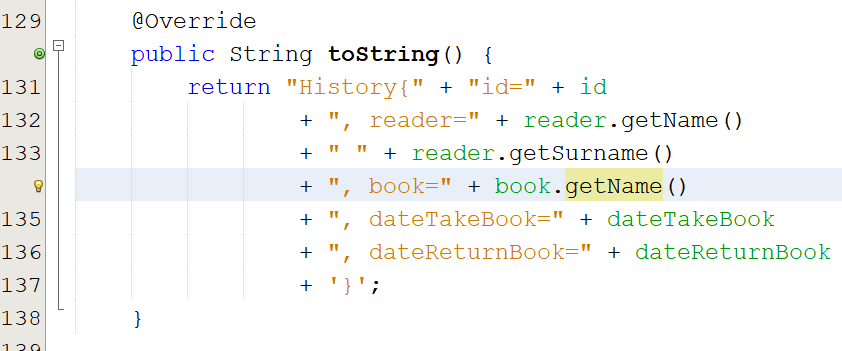
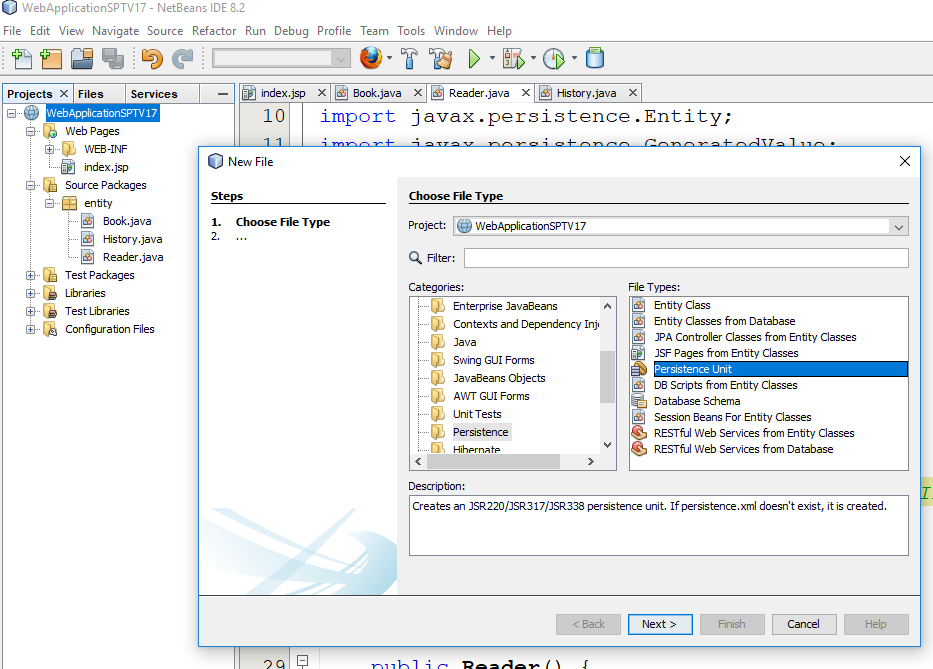
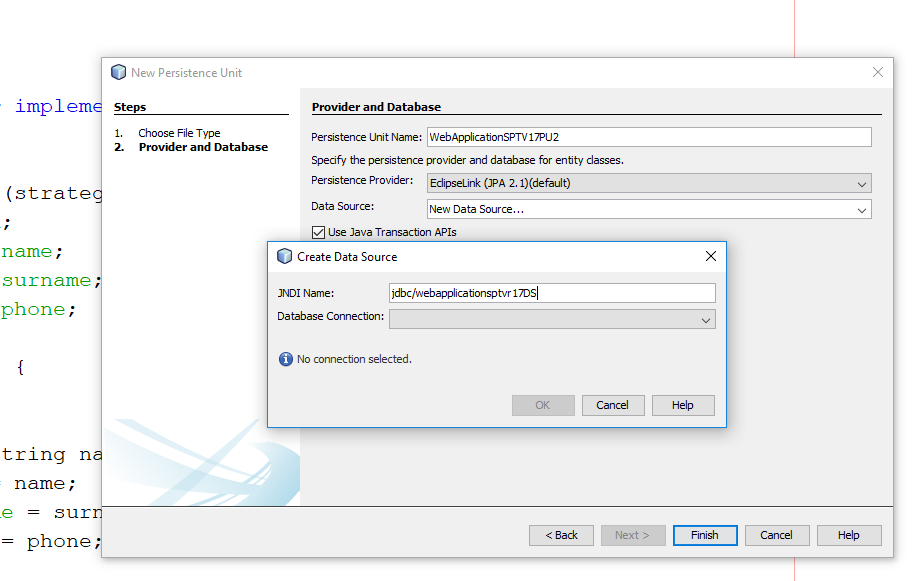
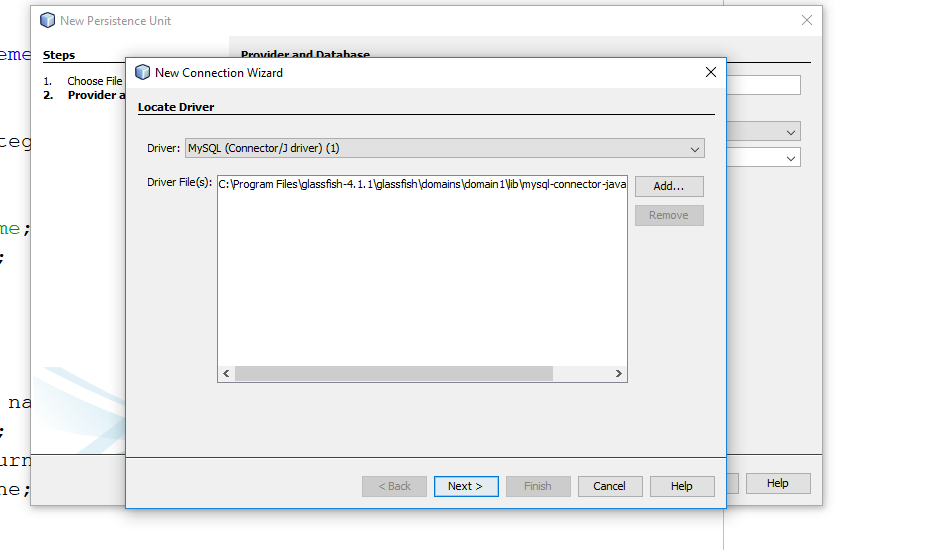
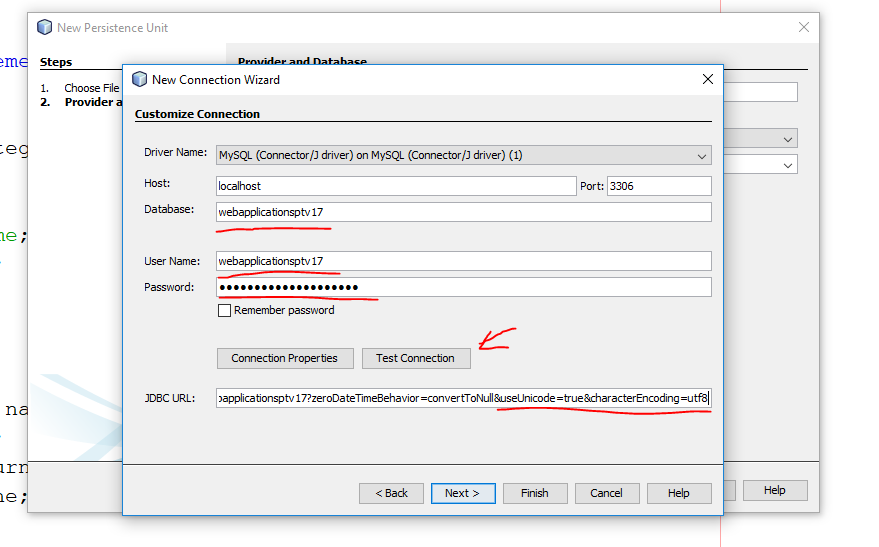
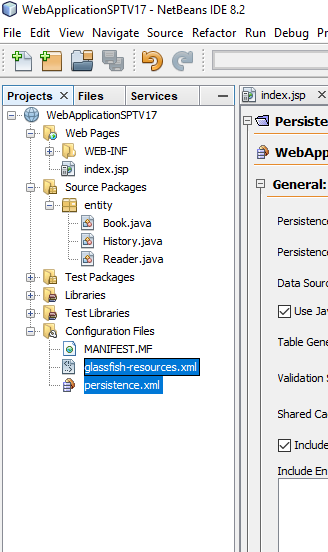
1. Загрузить инструменты: NetBeans, XAMPP, MySql, Apache-phpmyadmin.
2. В NetBeans создаем вебпроект, указываем название проекта с большой буквы и папку своей группы.  
   A screenshot of a social media post

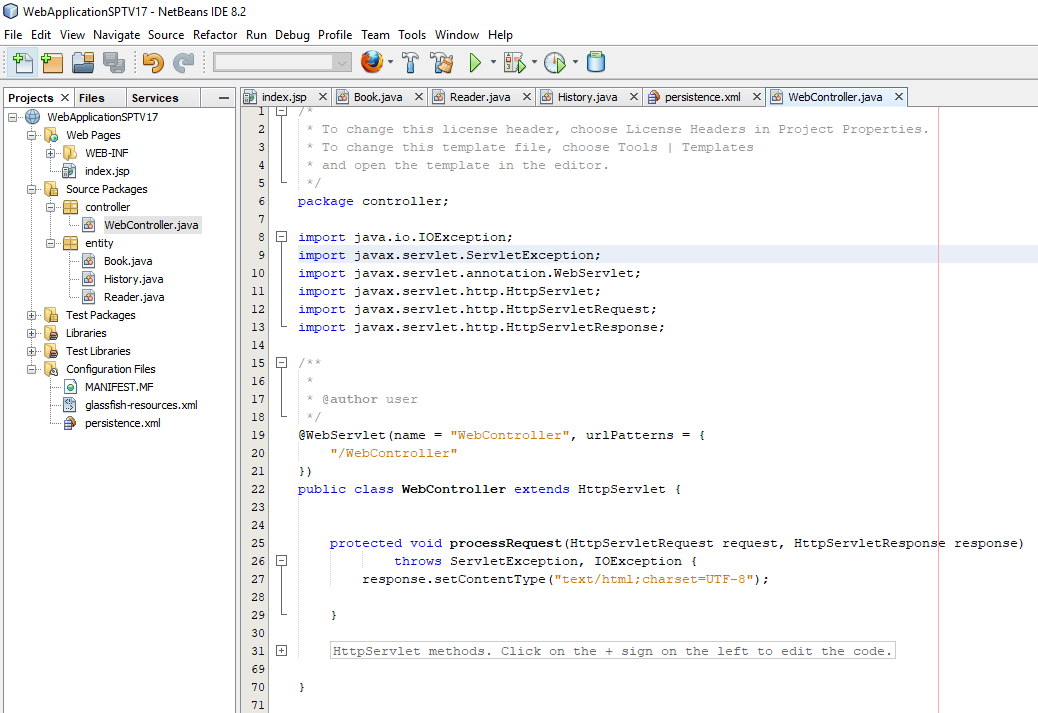
   Description automatically generated
3. В phpmyadmin cоздаем пользователя базы данных c именем соответствующим названию проекта, **только маленькими буквами (!)**

A screenshot of a cell phone

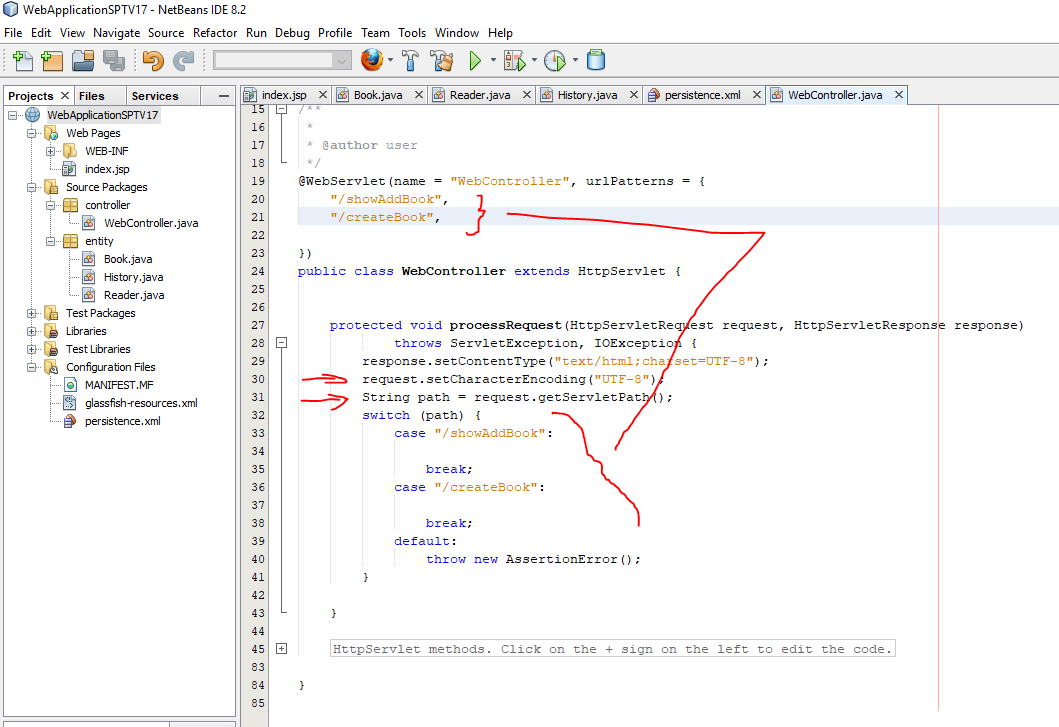
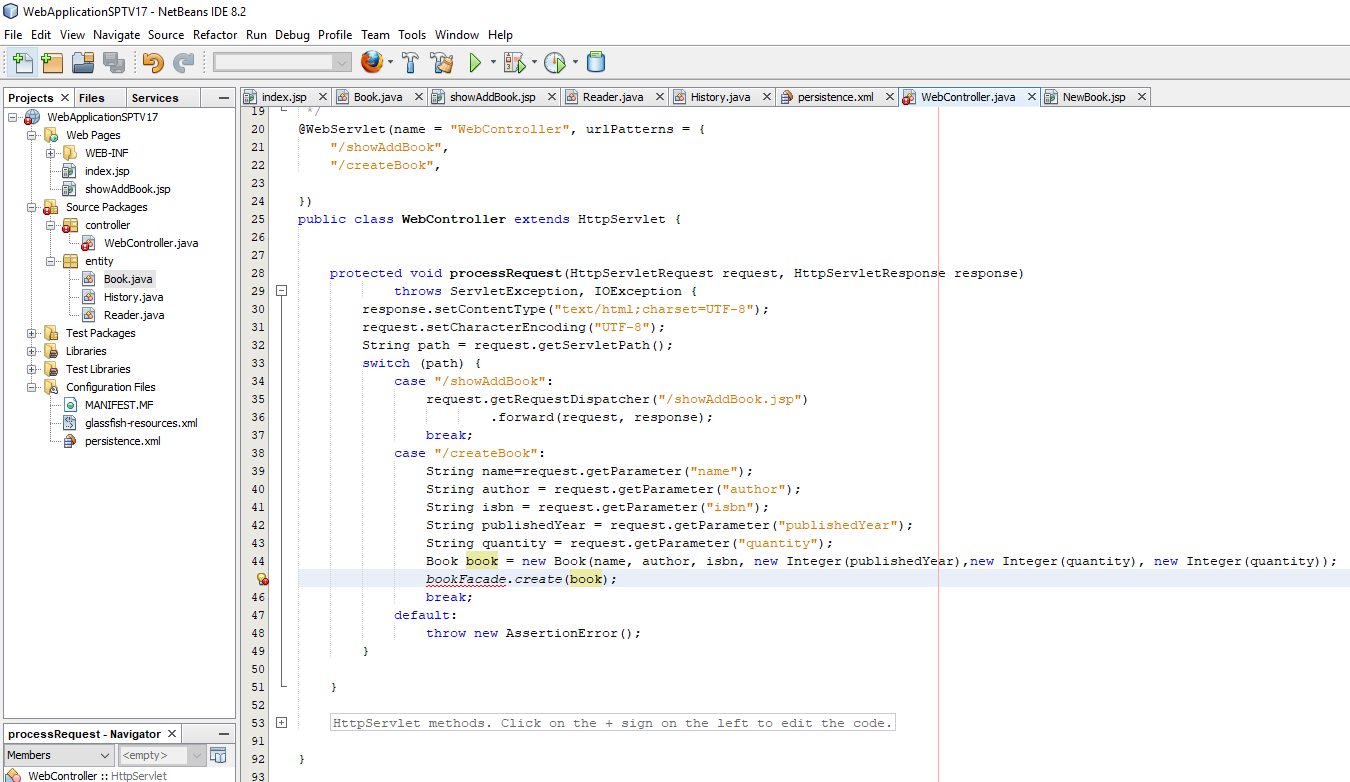
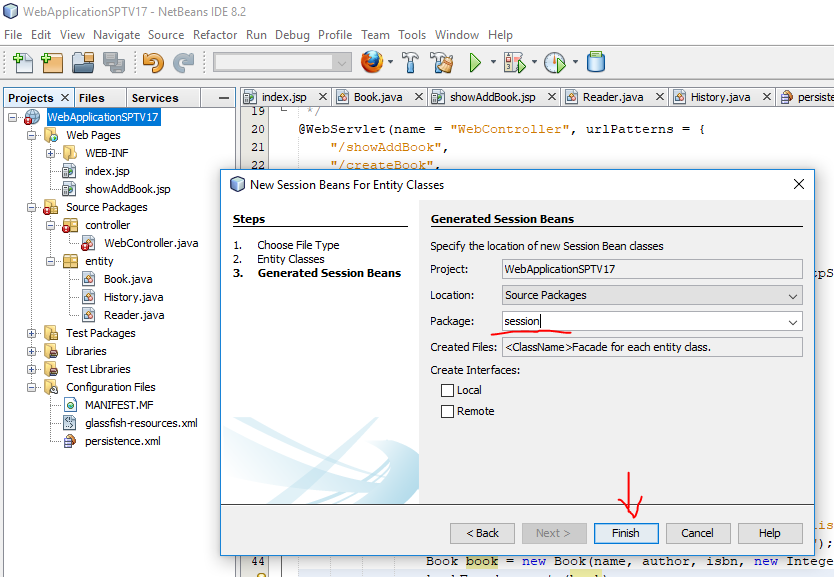
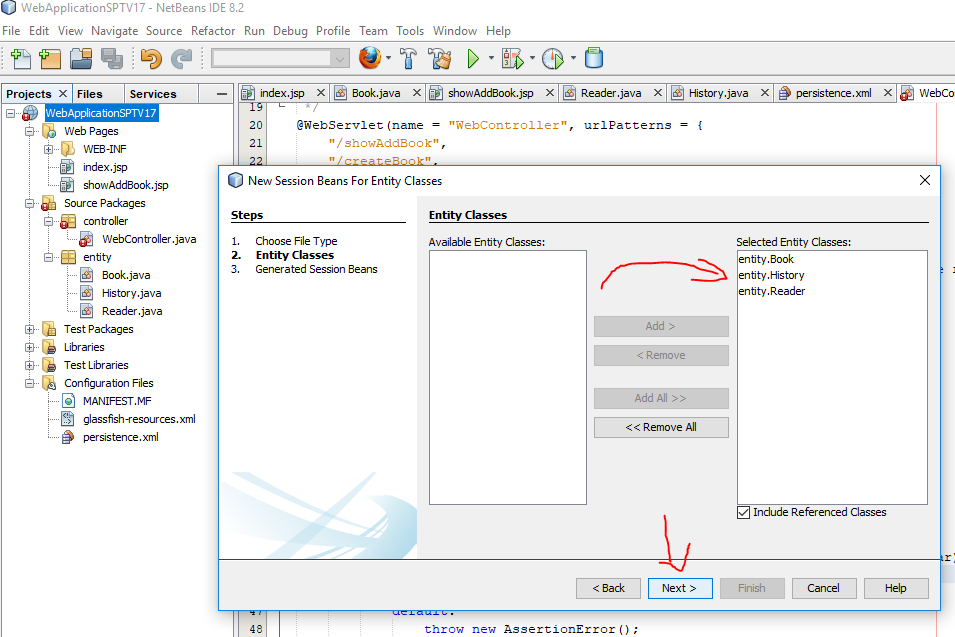
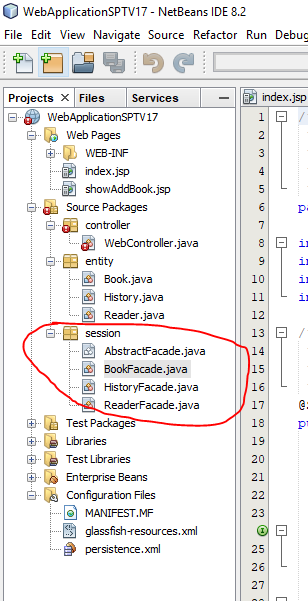
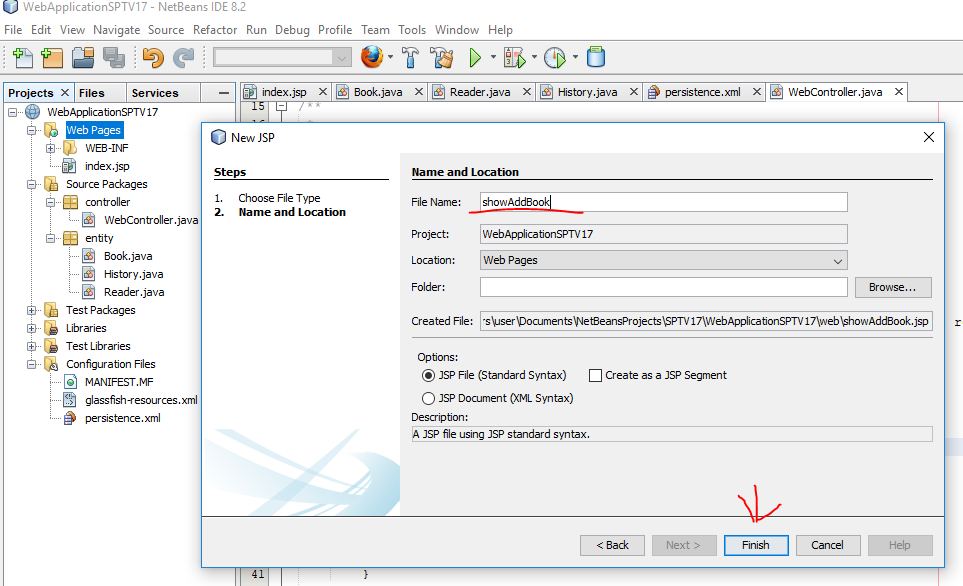
Description automatically generated

1. В NetBeans в проекте удаляем файл index.html и создаем index.jsp  
   A screenshot of a cell phone

   Description automatically generated
2. Создаем в проекте классы сущности.  
   Сущность Book будет выглядеть примерно так:  
     
     
   *Не забываем, в сущности должны присудствовать анотации и необходимые члены класса:  
   - поля, в которых хранится состояние сущности  
   - пустой конструктор  
   - геттеры и сеттеры для всех полей  
   - переопределение equals & hashCode  
   - переопределение метода toString()*
3. Создаем также сущности Reader и History  
     
     
     
   Не забываем, что перед классом в поле должны быть прописаны анотации, которые указывают на связь между таблицами. В данном случае @OneToOne и @Temporal(…)  
   И еще, в сущности History метод toString() я советую сделать как то так:  
     
     
     
   Это позовлит видеть состояние этой сущности более компактно и достаточно информативно.
4. Создаем подключение нашего проекта к базе данных.  
   Для этого создаем Persistence Unit с помощью помощника NetBeans  
     
     
     
   Создаем новый DataSource с именем jdbc/webapplicationsptv17DS (название нашей базы данных с постфиксом «DS».  
     
     
   Выбираем драйвер MySql   
     
   И делаем подключение к нашей базе данных  
   
5. В результате наших действий должно быть создано два файла  
     
     
     
   Файл glassfish-resources.xml я вам покажу. Вот он:  
     
   Очень много букв, но если вы присмотритесь, то увидите здесь атрибуты подключения к базе данных: имя базы, пользователь, пароль …  
   Для нас будет важна строка которая на изображении выделена  
     
    <property name="URL" value="jdbc:mysql://localhost:3306/webapplicationsptv17?zeroDateTimeBehavior=convertToNull**&amp;**useUnicode=true**&amp;**characterEncoding=utf8"/>  
     
   Причем, это именно одна длинная строка, без всяких новых строчек. То что выделено в этой строке должно присутствовать. Если у вас нет такого, то отредактируйте файл glassfish-resources.xml руками и впишите недостающие символы.  
     
   Теперь у нас есть подключение проекта к базе данных и мы можем работать с бизнес логикой
6. Чтобы управлять запросами пользователя создадим класс сервлет в пакете controller и назовем его WebController  
   A screenshot of a cell phone

   Description automatically generated  
     
   

Уберем все ненужное и оставим то, что нам пригодится (см. скрин)

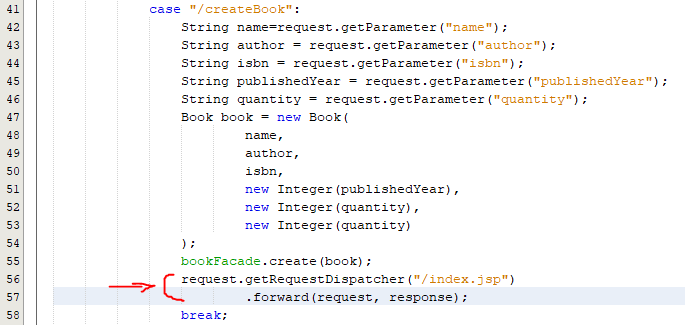
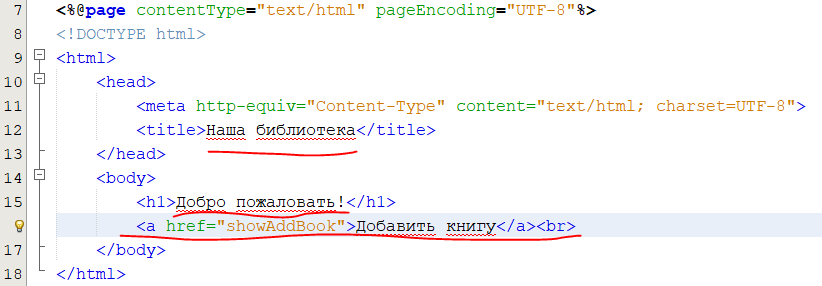
1. Заставим наш сервлет говорить на нормальном языке (UTF-8) и создадим механизм обработки запросов  
   
2. Создание логики приложения: добавление книги  
     
   объекта bookFacade у нас еще нет, поэтому строчка подсвечивается красной лампочкой (ошибка). Продолжаем работу и эта ошибка изчезнет, когда мы попросим EJB контейнер внедрить нужный объект.
3. Создадим инструменты легкой работы с базой данных   
     
     
   У нас появятся четыре класса, использующие паттерн Facade.  
     
   Их мы будем использовать для работы с базой данных
4. Создадим jsp страничку с названием showAddBook.jsp  
     
     
   

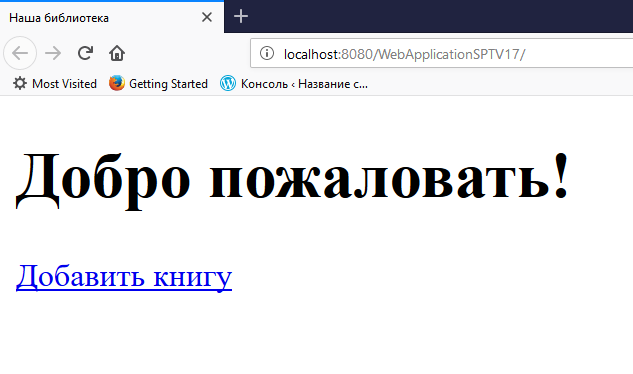
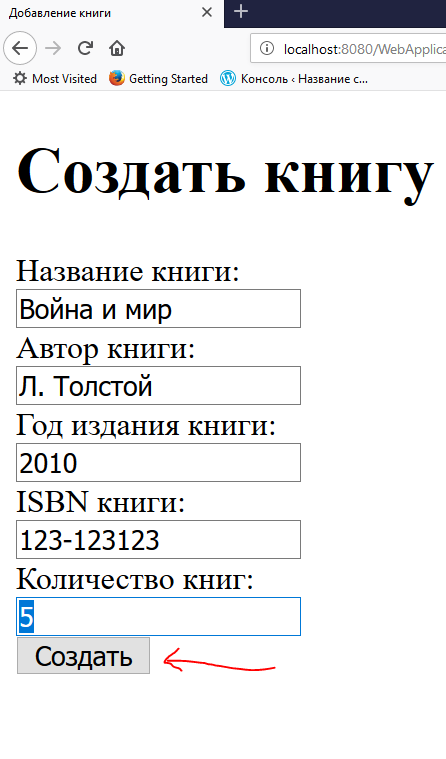
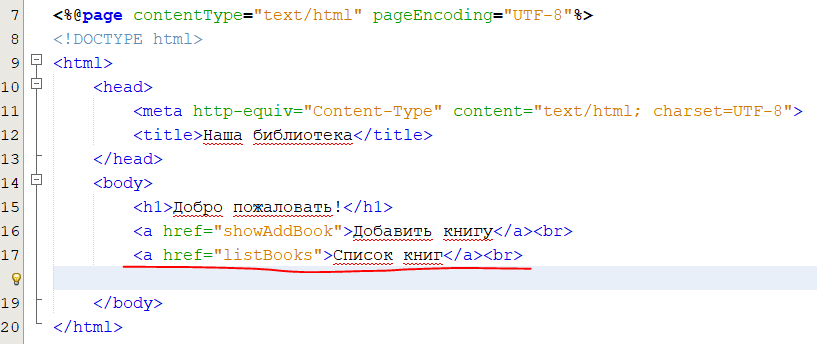
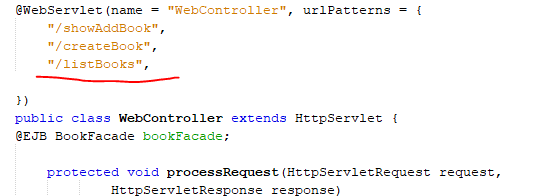
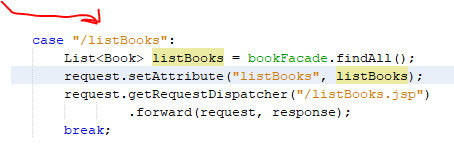
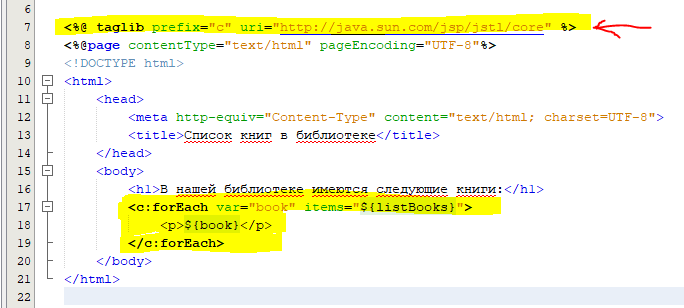
Обратите внимание, параметр action тега form содержит шаблон, который будет перехватывать наш сервлет (WebController), потому что такой же паттерн прописан в анотации @WebServlet(…), в параметре urlPatterns(“createBook”,…)

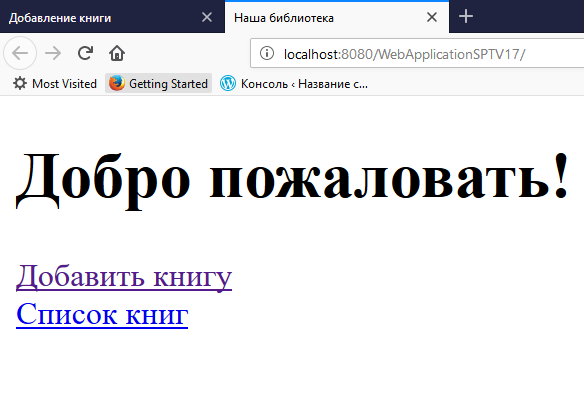
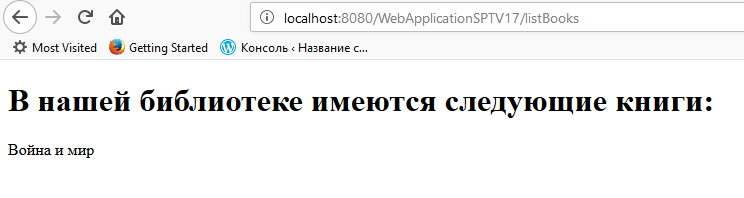
1. И последнее, подключим созданный класс фасад к нашему контроллеруA screenshot of a social media post

   Description automatically generated

Напомню, в верхней строке мы просим EJB контейнер внедкить объект bookFacade в поле сервлета с помощью анотации @EJB. После этого ошибка изчезнет.

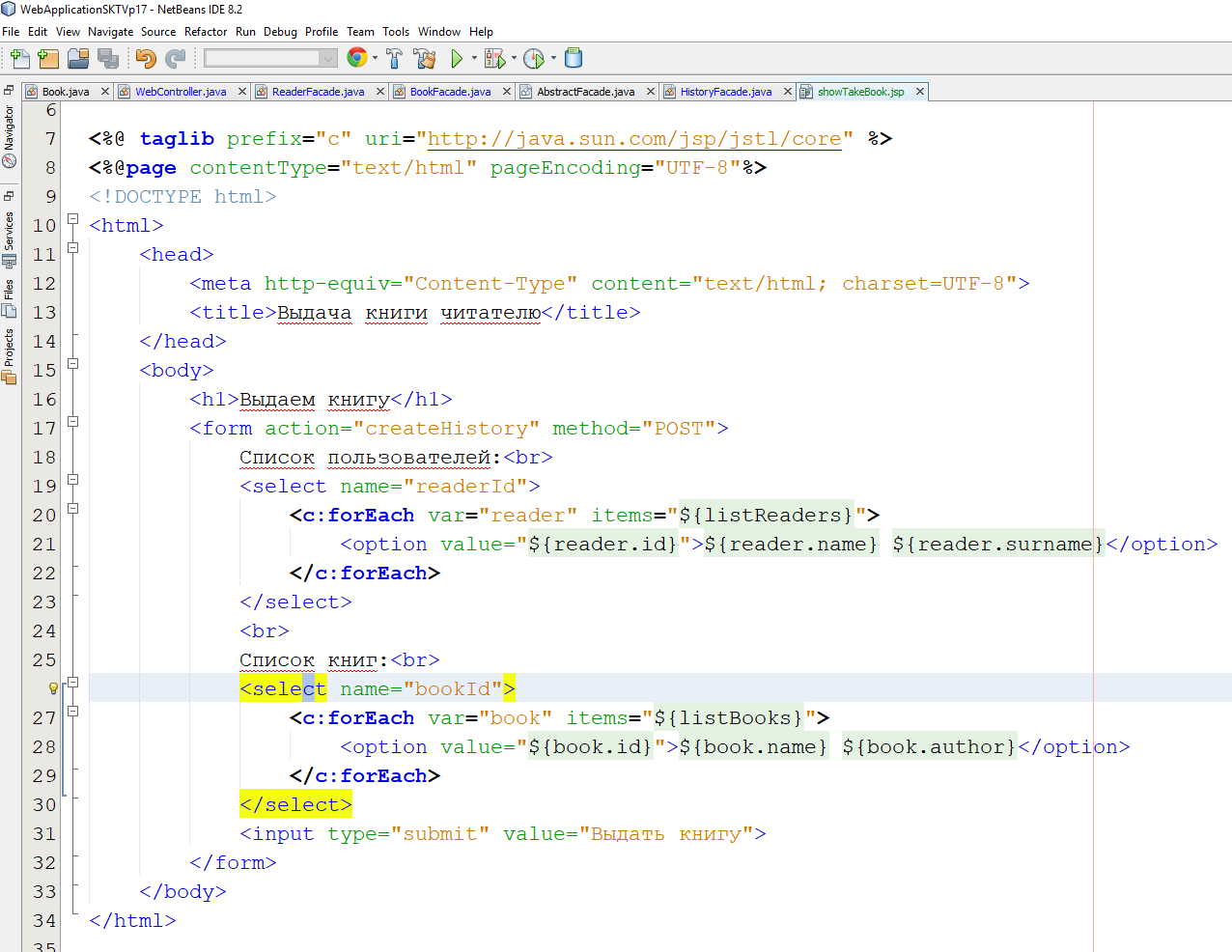
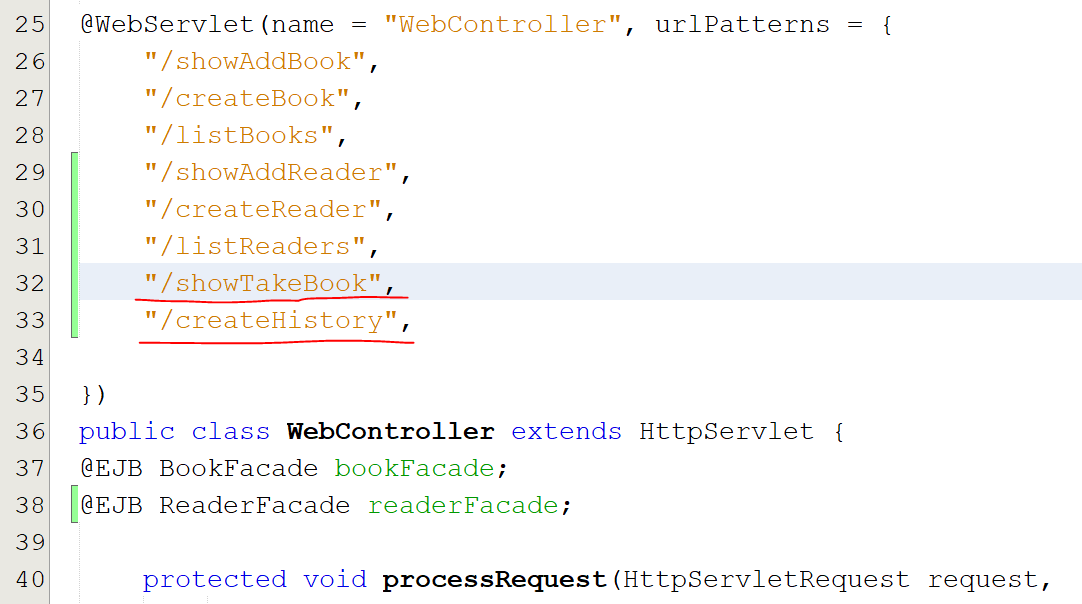
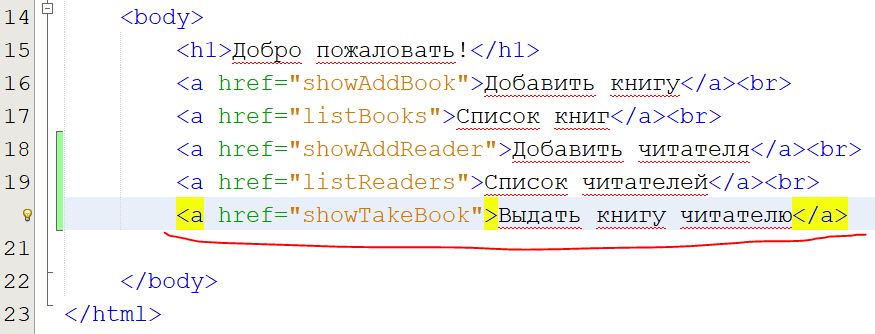
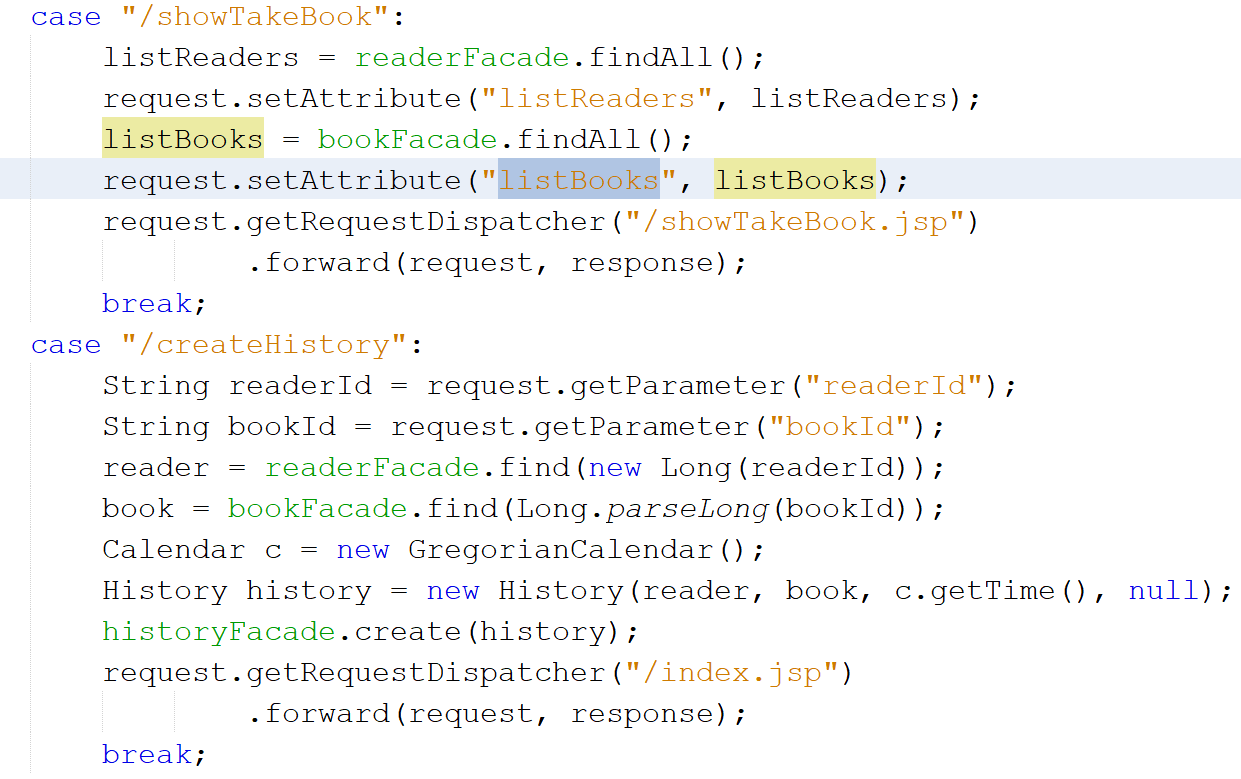
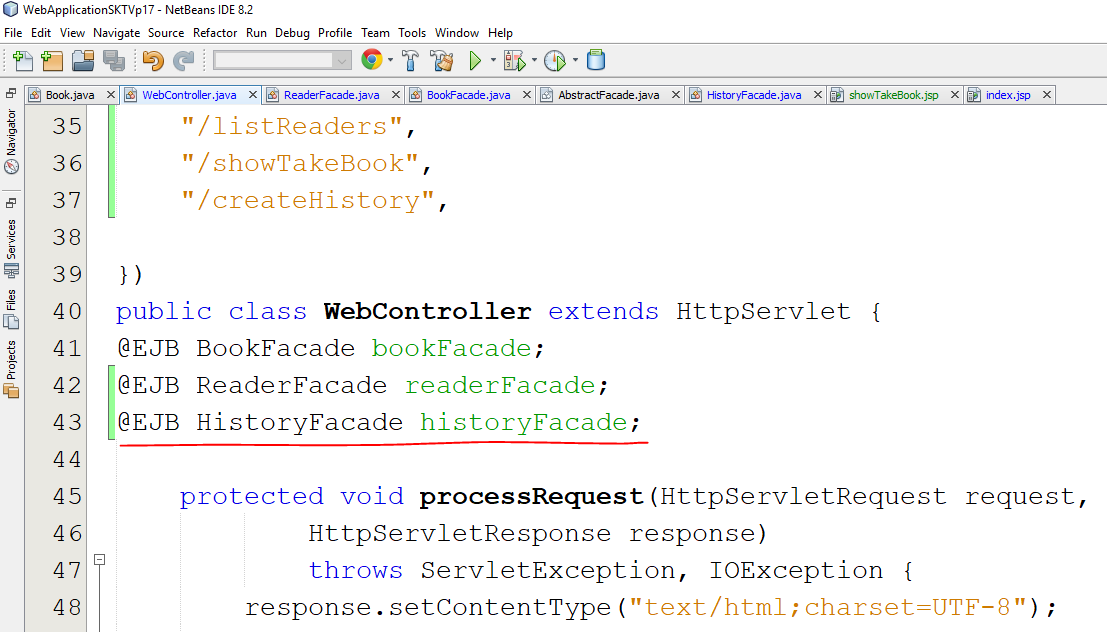
1. В результате у нас получится такой case, который будет записывать книгу в базу данных и отправлять пользователю индекную страницу  
   
2. Следующий шаг, это добавление ссылок в индексных файл. Это нужно для того, чтобы вызвать у приложения те функции, которые оно может выполнять. А наше приложение уже может выполнить две функции: показать страничку добавления книги и записать книгу в базу данных. Чтобы их вызвать нам нужно сначала вызвать первую, а затем вторую.   
   Внесем в индексную страницу следующие изменения.  
     
   
3. Теперь остается только откомпилировать, собрать проект и запустить  
   A screenshot of a social media post

   Description automatically generated
4. Проект запустится и в браузере покажется индексная страничка  
     
     
   после нажатия на ссылку покажется следующая страничка  
     
   нажмите на «Создать». После этого покажется индексная страничка снова.
5. Добавим третью функцию нашему приложению. Создадим вывод списка книг имеющихся в библиотеке.   
   Для этого надо добавить в индексную страницу ссылку на новую функцию: “listBooks”.  
     
   
6. Теперь добавим изменения в контроллер   
     
     
     
     
   наш контроллер сработает на этот шаблон и выполнит case.  
   Возьмет из базы список объектов Book, передаст его в контекст обработки ответа и отправит в браузер страничку listBooks.jsp
7. Создание страницы listBooks.jsp  
     
   новые добавления отмечены маркером
8. Компилируем и запускаем проект  
   A screenshot of a social media post

   Description automatically generated  
     
     
   После нажатия на ссылку «Список книг» получаем:  
     
   Все правильно, в нашей базе есть только одна книга, именно та, что мы добавили выше

Точно так же как мы добавляли книгу, создайте страничку showAddReader, создайте код, который будет брать информацию о читателе со странички и записывать ее в базу данных, после чего выведет список всех читателей в страничку listReaders.

Настала пора выдавать книги пользователям. За эту операцию будет отвечать сущность History. Именно в ней есть поля, которые хранят объект книги и объект читателя, плюс еще и время выдачи книги читателю и время, когда читатель вернет книгу обратно в библиотеку.

1. Создадим страничку showTakeBook.jsp где выведем два списка: listReaders и listBooks с помощью специальной библиотеки тегов JSTL.
2. Добавим шаблон запроса формы в сервлет WebController  
     
   Добавим также шаблон запроса для показа странички showTakeBook.jsp  
   И пропишем его в тег <a> на индексной страничке (index.jsp).  
   
3. Следующий шаг, создать case-ы для обработки этих двух шаблонов запроса  
     
   Первый выводит страничку со списком читателей и книг (страничке нужны списки книг и читателей, мы умеем их добыть).   
   Второй получает идентификаторы выбранного читателя и книги, находит по ним объекты.  
   Далее мы создаем объект history, инициируем его полученными данными, и добавляем еще время полученное с помщью класса Calendar.   
   Остается только записать с помощью historyFacade историю в базу.   
   Чтобы воспользоваться historyFacade необходимо его объект внедрить в поле контроллера с помощью анотации @EJB  
   

Переходим на страничку index.jsp  
