Настройка Вэб приложения.

Вэб проект содержит папку **web**, где хранятся страницы проекта. В папке **web** имеется папка **WEB-INF**, где можно скрыть файлы от внешнего просмотра, т. е. файлы html, которые находятся в этой папке невозможно вызвать набором их имени в адресной строке браузера. Для управления файлами, находящимися в папке **WEB-INF** необходимо создать и настроить сервлет.
Сервлет, это пользовательский класс, который расширяет класс **HttpServlet**, созданный разработчиками Java.
Анотация перед объявлением пользовательского класса,
@WebServlet(name = "library", urlPatterns = {"/page1","/page2"}),
дает имя сервлету и указывает шаблоны страниц, запросы которых он будет обрабатывать.

Среда Netbeans в пользовательском классе создает метод
processRequest(HttpServletRequest request, HttpServletResponse response).

Он принимает запросы GET и POST, которые может послать браузер при переходе пользователя по ссылке, или клике на кнопку submit формы.

Поговорим о параметрах метода **processRequest.**

В переменной-параметре **request** лежит объект класса **HttpServletRequest**, в котором содержится вся информация запроса, а в **HttpServletResponse response** содержится все, что сервлет отошлет браузеру.

Давайте в теле метода processRequest создадим выражение
String path = request.getServletPath();
Пусть браузер пришлет запрос соответствующий содержанию адресной строки

Это приведет к тому, что в переменной path будет лежать строка «page1»
Таким образом, метод **request.getServletPath()** возвращает строку, которая находится сразу за названием приложения в строке браузера.
Если теперь мы напишем строку, содержащую цепочку вызовов методов:
request.getRequestDispatcher(path + ".jsp").forward(request, response),
то заставим сервлет, с помощью метода **getRequestDispatcher(path + ".jsp"),** сформировать путь к существующей странице page1.jsp, и, далее, передать эту страницу методу **forward(request, response)**, который обработает request и, по протоколу HTTP, перешлет ответ браузеру.
Иными словами, если мы укажем в переменной p**ath** путь к jsp файлу, то этот файл будет послан браузеру.

Например:
p**ath = "/WEB-INF/page2"**, выведет страничку **page2.jsp** из защищенной от прямого доступа папки **WEB-INF**.

Таким образом, в этом методе мы можем получить запрос (request) отосланный пользователем (клиентом), обработать его и отправить пользователю ответ (response).

Чтобы обработать запрос пользователя (browser) и сформировать адекватный ответ, мы можем обратиться к базе данных, считать какие-то данные указанные в запросе, взять страницу jsp, вставить в нее данные из базы и отослать ее пользователю.
Чтобы web приложение могло работать с базой, его надо определенным образом настроить.

Настройка взаимодействия с базой данных.

Настройка сервера приложений glassfish.

Добавьте в папку glassfish->glassfish->domains->domain1->lib драйвер базы данных.
Теперь необходимо создать подключение сервера приложений к базе данных. Для этого лучше сделать пул (pool) подключений, тогда сервер сможет **создать несколько подключений к базе данных при развертывании приложения и раздавать эти подключения из пула** всем, кому это понадобится. Это улучшает быстродействие приложения, т. к. создание подключения к базе данных является очень трудоемкой операцией.

Создание пула подключений

Нам потребуется умение работать в командной строке.
В терминале следует запустить asadmin - утилиту glassfish. Эта утилита находится в папке bin корневой директории сервера приложений.
Когда начнет работать утилита, в терминале появится приглашение:

asadmin>

Запишите следующую команду:

asadmin>start-domain

Дождитесь, когда запустится сервер.
Следующая команда создаст пул подключений:

asadmin>**create-jdbc-connection-pool --datasourceclassname com.mysql.jdbc.jdbc2.optional.MysqlDataSource --restype javax.sql.DataSource --property portNumber=3306:password=ktvr17weblibrary:user=ktvr17weblibrary:serverName=localhost:databaseName=ktvr17weblibrary:useUnicode=true:characterEncoding=utf8 ktvr17weblibraryPool**

Эту команду следует ввести одной строкой. В команде красным цветом выделены атрибуты подключения к раннее созданной базе данных, это пароль, пользователь и название базы данных.
Следующий этап — создание DataSource:

asadmin> create-jdbc-resource --connectionpoolid ktvr17weblibraryPool jdbc/ktvr17weblibraryDS

где **ktvr17weblibrary Pool** — выше созданный идентификатор пула, **ktvr17weblibrary DS** — созданный DataSource.
Осталось только перезапустить сервер приложений glassfish

asadmin> restart-domain

После перезапуска сервера можно выйти из утилиты **asadmin** набрав команду «exit».

Настройка классов-сущностей нашего приложения

Создайте классы сущностей :).
Или скопируйте их из уже созданного раннее приложения в проект.

Создание сессионных компонентов классов-сущностей

Выберите мастер создания сессионных компонентов.
Новый файл → Enterprise JavaBeans → сеансовые компоненты для сущностных классов.

Укажите новый пакет в мастере, назовите его «**session**»
Выберите все классы-сущности, или поставьте галочку, которая задействует функцию автоматического добавления сущностей и нажмите кнопку «Готово».
Теперь у нас есть компоненты EJB, которые возьмут на себя заботу по операциям с базой данных.

Настройка сервлета для использования сессионных компонентов
Чтобы иметь доступ к созданным компонентам, в нашем (выше созданном и настроенном) сервлете, сразу после объявления класса, необходимо добавить специальную аннотацию и объявить переменные нужных нам сессионных компонентов.

```
@EJB
BookFacade bookFacade;
@EJB
ReaderFacade readerFacade;
```

Следующий шаг опциональный и заключается в переопределении init метода, который будет выполняться при запуске нашего веб приложения.
Клик правой кнопкой мыши → вставка кода → переопределение метода → GenericServlet → init.
Вот что должно получиться:

```
@Override
public void init() throws ServletException {
    getServletContext().setAttribute("books", bookFacade.findAll());
}
```

Что делает строчка, добавленная в переопределенный метод init()?
Метод **getServletContext()** возвращает интерфейс ServletContext.
Этот интерфейс определяет набор методов, которые сервлет использует для связи с контейнером. Для любого web-приложения существует один и только один ServletContext, поэтому любую информацию в нем можно считать глобальной и доступной из любой точки приложения (если приложение НЕ является распределенным, т. е. размещенном на разных компьютерах).
Метод setAttribute запускает указанный в его параметрах метод EJB компонента (bookFacade.findAll()), который ищет все книги в базе и найденное кладет в переменную "books" в виде списка объектов book.

Теперь мы можем получить доступ к этой переменной (books) в соответствующей jsp страничке с помощью языка JSTL. Про язык JSTL смотри методический материал на siseveeb.ee в разделе Java.