**Servlet API.**

**Задание-5.**

**Регистрация, авторизация и…**

**…RequestDispatcher.**

*Пойди туда, куда скажу. Принеси то, что попрошу!*

*Девиз ReqestDispatcher-а.*

Вообще говоря, основная цель данного практического задания – изучение объекта javax.servlet.RequestDispatcher и применение его возможностей при разработке проектов. Однако это задание так же скрывает в себе немножко халявы и один неявный бонус, о которых Вы узнаете походу его прочтения.

Суть задания заключается в следующем – Вам необходимо при помощи сервлетов разработать следующий набор страниц и организовать взаимодействие между ними:

1. Домашнюю страницу приложения.
2. Страницу регистрации.
3. Страницу авторизации.
4. Страницу фотоальбома.

Ниже представим некоторые уточняющие моменты по данным страницам:

При первом доступе к приложению должна отображаться ***домашняя страница***.

Содержание страниц Вы можете продумать и реализовать самостоятельно, но общая структура (верстка) для ***всех страниц*** должна быть ***одинаковой*** и выглядеть так, как показано на рис. 1.

Обратите внимание на то, что каждая страница обязательно должна включать в себя четыре блока, три из которых будут одинаковыми ***для всех*** страниц. Эти блоки помечены номерами на рис. 1. Ими являются:

1. **Заголовок страницы**.
2. **Боковое меню приложения**.
3. **Нижняя часть страницы**. На английском языке это называется Footer, Web-разработчики эту часть еще называют «подвалом». Но «нижняя часть страницы» – это совсем отвратительное название, просто ничего другого в голову не приходит.
4. **Содержательная часть страницы**. Тоже название отвратительное. Проще сказать – content.

Уверен, Вы догадались, что одинаковыми на всех страницах будут блоки ***№№ 1, 2 и 4***. Блок ***№4*** будет содержать уникальную для каждой страницы верстку.

Обратите внимание на то, что главное меню Вашего приложения должно быть представлено в двух разных блоках – первом и втором. Рис. 1 показывает, что внешний вид главного меню отличается в этих блоках - оно либо горизонтальное, либо вертикальное, но при этом функциональные элементы (ссылки) в меню должны быть ***одинаковыми***.

Блоки ***№№ 1, 2 и 4*** должны быть реализованы при помощи ***отдельных*** сервлетов, и включаться в сервлеты основных страниц при помощи метода include() объекта javax.servlet.RequestDispatcher. Во время работы над этим пунктом задачи я настоятельно рекомендую Вам изучить паттерн [Composite View](http://www.oracle.com/technetwork/java/compositeview-137722.html), который широко применяется при верстке страниц в реальных приложениях. Можно сказать, что метод include() – это наша первая попытка реализовать Composite View. Позже по ходу курса мы увидим еще несколько способов, как это реализовать. И даже будем знакомиться со специальным фреймворком, который решает только эту задачу. Однако какие бы технологии мы ни применяли, в основе каждой из них все равно будет лежать метод include(). Не даром, сервлеты – фундаментальная технология JEE.… Хоть и паскуд#@я, но все-таки – фундаментальная ☺.

Ссылка ***«авторизация»*** должна быть не видима для пользователей до тех пор, пока пользователь не выполнит хотя бы одной регистрации в проекте.

Пользователь может регистрироваться в приложении сколько угодно раз, при этом приложение должно запоминать ***всех зарегистрированных*** пользователей. Однако обратите внимание на следующий момент – в текущем задании зарегистрированные пользователи должны сохраняться только на время работы с приложением. Другими словами, не используйте для этого, файлы, базу данных и т. д., храните зарегистрированных пользователей в сессии пользователя – все постоянные хранения нас еще ждут впереди.

Если пользователь зарегистрировался, или выполнил вход в приложение (авторизовался), то ссылки **«регистрация»** и **«авторизация»** в главном меню должны исчезнуть, а вместо них должны появиться ссылки:

1. **Фотоальбом** – переводит пользователя на его фотоальбом. Реализация самого фотоальбома меня не интересует – тут Вы либо проявляете максимум Вашей фантазии, либо просто пишете в середине страницы – «фотоальбом» – оба варианта будут приняты с одинаковым восторгом.
2. **Выход** – по нажатию на эту ссылку должен осуществиться выход из приложения (и тут все дружно вот это вот банальное – СПАСИБО КЭП!! ☺). Т. е. должно произойти следующее:
   * Авторизованный пользователь должен быть удален из сессии приложения.
   * Управление должно передаться на домашнюю страницу.
   * Ссылки «Фотоальбом» и «Выход» должны стать невидимыми.
   * Ссылки «регистрация» и «авторизация» – наоборот – видимыми.

Предпоследнее требование задачи. Все переходы между страницами должны осуществляться при помощи ссылок. Однако в этом приложении у Вас будут ситуации, когда один сервлет должен будет передать управление другому сервлету. Например, когда пользователь жмет ссылку «Выход», то изначально управление передается сервлету LogoutServlet (имена сервлетов здесь только для примера, Ваши имена могут быть ***любыми***). Однако после того, как LogoutServlet отработает, управление из этого сервлета должно передаться сервлету HomePageServlet, для того чтобы он отрисовал домашнюю страницу. Так вот, в текущей задаче все передачи управления от сервлета к сервлету должны выполняться при помощи метода forward() объекта javax.servlet.RequestDispatcher.

Последнее требование задачи. Как Вы могли заметить, в текущей задаче Вам придется очень активно работать с объектом javax.servlet.http.HttpSession. Так давайте, имея такой шанс, закрепим на практике еще один теоретический вопрос, связанный с этим объектом – ***конфигурирование времени жизни сессии***. Пожалуйста, добавьте в файл web.xml конфигурацию, которая позволит задать время жизни сессии в минутах. По истечению этого времени сессия пользователя должна уничтожаться, а, следовательно, нажатие пользователем на любую ссылку приложения в этом случае должно обрабатываться точно так же как и при нажатии на ссылку ***«Выход»***. В реальной жизни Вы могли видеть примеры такого подхода, например, в приложениях интернет-банкинга – если пользователь не проявляет активность в приложении интернет-банкинга определенное время, то любое его действие после этого ведет его на страницу авторизации. Следовательно, время жизни сессии – это один из способов защиты данных пользователя. Если он отошел от компьютера, забыв при этом выйти из приложения, то спустя какое-то время система сделает это самостоятельно, а то мало ли кто вокруг ходит. Хотя, как правило, вокруг никто не ходит, сам пользователь, скорее всего, отошел чайку попить, а вернувшись, он говорит совсем не это: *«Ох, молодец мое приложение! Вот оно капец как заботиться о моей безопасности!! Одним словом – качественная компьютерная система!!!»*, а что-то вроде *«@#$%$#@#$%$# авторизация @#&#ная!»*. Но нам это не важно – порядок, есть порядок!

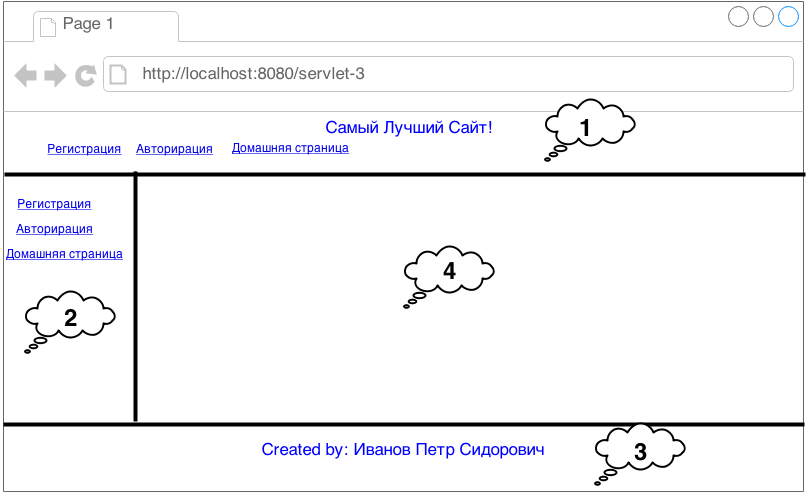


Рис. 1. Пример одной из страниц приложения.

Ах, да, чуть не забыли. Так где же тут была халява, и в чем заключается бонус? Ну, халява в том, что Вы уже сделали и сдали работу по Web-программированию – HTML, CSS, JS – помните? Соответственно думать о том, как верстать большинство страниц текущего приложения Вам не нужно – верстка уже готова и ждет дальнейшего применения. Согласен, халявы здесь мало. Тем более, вся она будет нейтрализована тем простым фактом, что всю эту верстку Вам придется внедрить в сервлеты, а это, как Вы уже, наверное, успели заметить – ад. Но все равно – она, все-таки, есть ☺.

Ну и бонус. Все Вы делаете курсовой проект. А какой курсовой проект может обойтись без главных функций – регистрация, авторизация, главная страница, logout? А тут – тара-тара-таааара-та-та-та – у Вас уже все это есть ☺. Остается только придумать, в какой именно системе Вы все это будете использовать, разработать модель данных этой системы, добавить поддержку БД, и переписать все, что Вы сделали в текущем задании к бубеням. И все – курсач готов! Какой же это бонус, скажете Вы, если все равно все переписать придется? Ну да придется, ну и что… Архитектура-то в голове уже сидит, все остальное – дело техники ☺.