**Servlet API.**

**Задание-10.**

**Игра «Крестики-Нолики».**

Основная цель данного задания – просто программировать – писать много Java кода реализующего не совсем тривиальные алгоритмы типа «один for, два if-а, один else». Стараться удержать в голове и реализовать в проекте все требования преподавателя (ну в будущем, заказчика ☺). Подумать над способом представления, хранения и передачи относительно большого количества разнородных данных и т. д. Одним словом, почувствовать себя разработчиком проекта, хоть как то похожего на настоящий, а не банальной лабораторной работы, которая никому кроме Вас самих не нужна. Однако, пожалуйста, не питайте излишних иллюзий – этот проект тоже не будет нужен никому кроме Вас ☺. Однако, все равно приступим.

В рамках текущей лабораторной работы Вам предстоит реализовать и протестировать развлекательное приложение, позволяющее пользователям играть в одну из самых популярных игр всех школьников советского союза – **крестики-ролики**. По требованиям текущей работы оппонентом пользователя в смертельной битве на виртуальном клетчатом поле ***выступит компьютер***.

Приложение должно состоять из трех сервлетов:

1. Сервлет для отображения страницы приветствия.
2. Сервлет, реализующий игровую карту.
3. Сервлет отображения статистики по окончании игры.

При первом доступе к приложению запрос пользователя должен переадресовываться на страницу приветствия, которая может выглядеть следующим образом:

***Примечание: изображение пока в процессе разработки.***

На странице приветствия, пользователь должен ввести свое имя в текстовое поле.

Так же пользователь может выбрать, какими фигурами он предпочитает сражаться – крестиками или ноликами. Выбор типа фигур должен осуществляться при помощи радио кнопок (см. Рис. 1). По умолчанию система должна предлагать пользователю играть крестиками и соответствующая радио кнопка должна быть выбрана при отображении страницы приветствия.

Последним компонентом на странице приветствия должен стать Checkbox, указывающий, должен ли меняться тип фигуры для пользователя после каждой игры. По умолчанию данный Checkbox должен быть включен, что означает смену фигур.

Заполнив форму приветствия, пользователь может нажать кнопку «Начать Турнир», после чего управление передается на страницу с игровой картой (см. ниже).

Однако если пользователь не заполнил поле с именем, либо ввел в качестве имени строку, состоящую только из пробельных символов, управление должно остаться на странице приветствия, и на экране должно отобразиться всплывающее окно, сообщающее об ошибке ввода имени.

Обратите внимание на то, что валидация имени должна быть реализована ***на клиентской стороне при помощи JavaScript-а***. Всплывающее окно в этом случае можно реализовать при помощи стандартного JS объекта **window.alert**.

Однако, *чуть более высшим пилотажем, чем низшим*, лучшим выбором будет внедрение сообщения об ошибке прямо в Web страницу рядом с полем для ввода имени. В этом случае сообщение об ошибке валидации может располагаться над полем для ввода имени, под этим полем или справа от него, и выделяться среди остальных текстовых элементов отличным цветом.

Довольно часто для сообщений об ошибках используют красный цвет и жирный шрифт. Также сообщение об ошибке можно по вкусу приправить симпатичной пиктограммой в виде крестика, восклицательного знака и т.д. однако последнее – целиком Ваше решение и обязательным требованием не является.

Если клиентская валидация прошла успешно, управление передается на сервлет, отображающий игровую карту.

Сервлет игровой карты, прежде всего, должен провести серверную валидацию данных и проверить имя пользователя по тем же правилам, по которым оно проверялось на стороне клиента (см. выше).

Серверная валидация должна стать первым этапом метода **service()** и выполняться на языке java. В случае если сервлет обнаружит, что имя пользователя не введено или состоит только из пробельных символов, дальнейшая логика сервлета игровой карты выполняться не должна, а вместо этого должно быть сформировано сообщение об ошибке и управление должно вернуться на страницу приветствия.

Страница приветствия в этом случае должна отобразить принятое от игровой карты сообщение об ошибке рядом с полем приветствия, так как было описано выше. JavaScript в этой ситуации использовать не потребуется.

Обратите внимание на то, что хорошим тоном по отношению к пользователю является ***полное совпадение*** сообщений об ошибках валидации на клиентской и на серверной стороне для одной и той же ошибки. Тем самым Вы гарантируете, что не будете заставлять пользователя тратить ресурсы его мозга на изучение ***двух различных*** сообщений, которые при этом описывают ***одну и ту же*** ошибку. Тем самым Вы помогаете пользователю быстрее привыкнуть к Вашему приложению. Тем самым Вы заставляете пользователя быстрее начать любить Ваше приложение и отказываться от его аналогов, разрабатываемых конкурентами… Однако мы отвлеклись, и после небольшого примечания, а скорее даже вопроса Вам, мы обязательно продолжим.

*Примечание-вопрос: Пожалуйста, подумайте над вопросом, зачем в приложении нужно организовывать* ***два уровня проверки – и на стороне клиента и на стороне сервера****? Приведите как минимум один пример (основываясь на текущем задании), когда только лишь клиентской проверки данных будет не достаточно. Если Вы приведете* ***два примера*** *(или больше?), и, более того, для второго примера сможете написать код, который докажет его состоятельность – я на ближайшем же занятии представлю Вас группе, как человека, которому данный курс не нужен, и с радостью поменяюсь с Вами местами, чтобы самому чему-нибудь у Вас научиться ☺.*

*Ответ на этот вопрос Вы можете высылать вместе с работой в тексте письма. Если до того, как Вы читаете этот вопрос, мы уже успели обсудить его с Вами – что ж отвечайте в письме что-то вроде – «Не волнуйтесь, я уже в курсе этой проблемы, и программируя, я регулярно бдю как комфорт клиента, так и спокойствие сервера». Ну, или ничего не отвечайте, но главное – не оставляйте этот вопрос без ответа.*

Ну а мы продолжаем…

В случае, когда серверная проверка данных завершилась успешно, сервлет должен отобразить Web-страницу с игровой картой, которая может выглядеть следующим образом:

***Примечание: изображение пока в процессе разработки.***

Как Вы можете видеть, игровая карта должна включать в себя следующие элементы:

1. Кнопку «Новая Игра»
2. Игровое поле.
3. Таблицу статистики турнира.
4. Таблицу истории турнира.
5. Кнопку «Закончить турнир»

Опишем эти элементы подробнее.

**Кнопка «Новая Игра»** запускает новое соревнование между пользователем и компьютером. В этот момент должны выполниться следующие условия:

1. Игровое поле должно очиститься от ранее расставленных фигур.
2. Количество игр в таблице статистики турнира должно увеличиться на единицу (см. ниже).
3. В таблице истории турнира должна появиться новая запись, содержащая информацию о:
   1. Порядковом номере игры.
   2. Времени начала игры.
   3. Нулевом количестве сделанных ходов.
4. Время начала игры должно отображаться в формате **«DD-MM-YYYY HH:MM:SS»**
5. Сама кнопка «Новая Игра» должна стать недоступной (disabled) до тех пор, пока текущая игра не закончится.
6. Кнопка «Закончить турнир» должна стать недоступной (disabled) до тех пор, пока текущая игра не закончится.

**Игровое поле** должно представлять собой матрицу 3\*3, в каждой клетке которой может отображаться одна из фигур – либо крестик либо нолик.

*Примечание: обратите внимание, что когда я говорю «матрица 3\*3», то, скорее всего, меня снова захлестывает либо пафос, либо ностальгия по универу. Уверен, Вы прекрасно понимаете, что в нашем случае, все эти «мега матрицы 3 на 3» – это банальный HTML элемент* ***<table>*** *с тремя строками и тремя столбцами.*

Игровое поле должно очиститься от ранее расставленных фигур, если была нажата кнопка «Новая игра» (см. выше). После этого пользователь и компьютер могут начинать игру.

Игра представляет собой последовательность ходов, в каждом из которых пользователь выполняет щелчок мышью по любой свободной клетке поля.

В этот момент в клетке по которой кликнул пользователь должна отобразиться фигура, которой пользователь играет в текущей игре. В этот же момент свой ход делает компьютер, отображая свою фигуру в любой свободной клетке поля. Обратите внимание, что компьютер должен выбирать свободную клетку **в случайном порядке**.

После каждого хода счетчик ходов в таблице истории турнира должен увеличиваться на единицу (см. ниже).

Игра продолжается до тех пор, пока на поле есть свободные клетки. Игра может закончиться либо победой пользователя, либо победой компьютера, либо ничьей.

Победа присуждается одной из сторон в том случае, когда одинаковыми фигурами последовательно заполнены три клетки по вертикали, горизонтали либо диагонали.

Ничья признается тогда, когда на поле больше нет свободных клеток, при этом ни одна из сторон не смогла одержать победу за предыдущие ходы.

После того, как текущая игра закончена, должны выполниться следующие условия:

1. В таблице истории турнира должно отобразиться время окончании игры в следующем формате: **«DD-MM-YYYY HH:MM:SS»**
2. Кнопка «Новая Игра» должна стать доступной (enabled).
3. Кнопка «Закончить турнир» должна стать доступной (enabled).
4. Если пользователь победил в текущей игре, на странице должна отобразиться красивая большая зеленая надпись – **«Поздравляем, Вы победили!»**
5. Если пользователь победил в текущей игре, в таблице статистики турнира **количество побед** должно увеличиться на единицу.
6. Если случилась ничья в текущей игре, на странице должна отобразиться красивая большая желтая *(ну вряд ли желтая – ее ж не видно будет… Хотя – все зависит от того, как раскрасить ;))* надпись – **«Что-то пока не понятно… Нужно сыграть еще!»**
7. Если случилась ничья в текущей игре, в таблице статистики турнира **количество ничьих** должно увеличиться на единицу.
8. Если пользователь проиграл в текущей игре, на странице должна отобразиться красивая большая красная надпись – **«Ну да, конечно, не повезло… Но ведь можно же взять реванш!»**
9. Если пользователь проиграл в текущей игре, в таблице статистики турнира **количество поражений** должно увеличиться на единицу.

**Таблица статистики турнира** должна отображать следующую информацию:

1. Количество всех сыгранных игр в рамках текущего турнира.
2. Количество побед пользователя.
3. Количество поражений пользователя.
4. Количество ничьих.

**Таблица истории турнира** должна отображать статистику каждой сыгранной игры выглядеть следующим образом:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Дата начала игры | Дата окончания игры | Количество ходов | Результат |
| 1 | 01-01-2014 00:00:00 | 01-01-2014 00:05:15 | 255 | Победа |
| 2 | 01-01-2014 00:16:00 | 01-01-2014 00:17:05 | 2 | Поражение |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

**Кнопка «Закончить турнир»**. Если пользователь нажимает на эту кнопку, то управление передается на сервлет отображения статистики турнира.

Данный сервлет должен еще раз отобразить таблицы статистики турнира и истории турнира, поздравить пользователя с окончанием турнира, а так же написать красивые сообщения, на подобии тех, которые мы выводили с Вами в конце каждой игры. На каких правилах должны основываться эти сообщения, я уверен – Вы догадаетесь сами.

Дамы и Господа, успехов и позитива Вам на все время выполнения этой работы!!

*P. S. Есть еще одно примечание… Я, конечно, понимаю, что мы с Вами изучаем программирование на платформе Java EE, и все это очень серьезно… Поэтому я, конечно, не могу требовать от Вас того, что написано ниже. Но все-таки у нас ТУРНИР! А турниры – они ведь всегда смертельные, пусть и на тетради в клеточку!!! В общем, что тут ходить вокруг да около - если Вы в процессе оформления Web-страниц найдете место для маленькой иконки черного дракона в огненном круге, ну или может там для небольшой фотографии Саб-Зи́ро, я буду крайне этому рад. Ну а если уж вместо нудной кнопки «Начать Турнир» я увижу что-то вроде –* ***«And Let Mortal Kombat Begin!»*** *­– я Ваш фанат на всю жизнь ☺*