**JSP API.**

**Задание-4.**

**Проверка знаний арифметики.**

Основная цель данного задания – избавить Вас от боязни отправлять форму Web-страницы на сервер чем-либо, кроме кнопки submit. Естественно, задание будет полезным только в том случае, если такая боязнь существует. В противном случае все выльется в банальную механическую работу по набору кода для очередной лабораторной работы. Так что, если Вы совершенно не боитесь таких понятий, как JavaScript-овый обработчик onClick – самое время начать его бояться – в конце текущего задания этот страх окупиться приливом огромного количества гормонов радости в Ваш организм ☺. Итак, задание:

Создайте JSP, которая проверяет знания арифметики у учащихся, а именно:

1. Сложение
2. Вычитание
3. Умножение
4. Остаток от деления

Остаток от деления в этом задании был выбран вместо обычного деления, потому что заставлять пользователей делить в уме и считать дробные числа – садизм.

Cервлет должен размещать следующие элементы (см. рис. 1):

1. Текстовую надпись приветствия.
2. Статистику прохождения тестов.
3. Тип текущего задания.
4. Строку с вопросом.
5. 4 радио-кнопки с ответами.
6. Строку, которая указывает, правильно ли ответил пользователь на предыдущий вопрос.
7. Кнопку «Следующее задание».
8. Кнопку «Закончить тест».

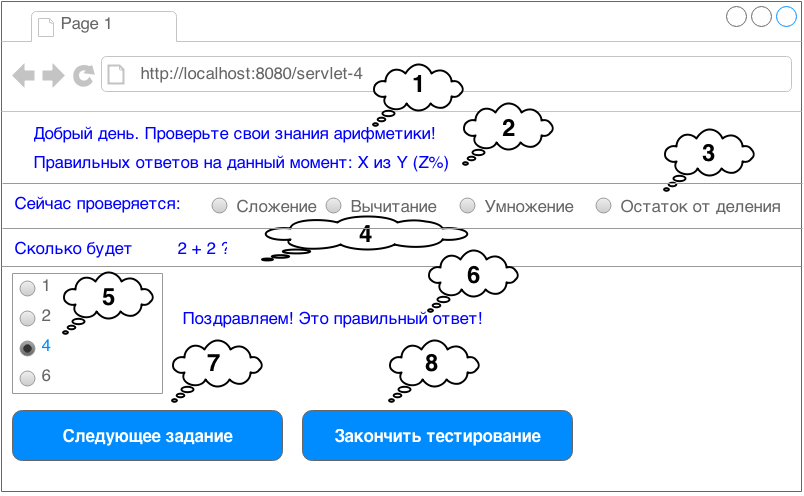


Рис. 1. Примерный вид интерфейса приложения.

Опишем эти элементы подробнее.

**1. Приветствие.** Н-да… описание как-то сразу не задалось ☺. В общем, тут все равно, что писать. Можете даже вообще не писать ничего - совершенно не принципиальный элемент.

**2. Статистика теста.** Это должна быть строка, из которой пользователь может увидеть:

* На сколько вопросов он уже ответил (X на рис. 1.).
* Сколько было правильных ответов (Y на рис. 1.).
* Каков процент правильных ответов (Z на рис. 1.).

Как Вы понимаете, значения X, Y и Z должны пересчитываться каждый раз, когда пользователь отвечает на вопрос.

**3. Тип задания.** Это опять-таки просто элемент декора. Но его интересно будет по программировать. Каждый раз, когда JSP генерирует новое задание, она должна определить тип задания – сложение, умножение, вычитание или остаток от деления. Тип задания каждый раз должен выбираться в ***произвольном порядке*** (т. е. здесь необходимо использовать генератор случайных чисел). И вот когда задание сгенерировано и JSP приступает к его отображению, одна из кнопок секции три должна быть выбрана. Это будет та кнопка, которая соответствует типу задания, которое «придумала» JSP страница. Обратите внимание на то, что сами кнопки должны быть не доступны для пользователя (disabled), так как пользователь не должен менять их положение – это только информативные кнопки.

**4. Строка с вопросом.** Эта строка так же формируется JSP при генерации ***каждого вопроса***. Здесь обратите внимание на два момента.

Во-первых, оба операнда (слагаемые, множители и т. д.) должны быть случайными числами.

Во-вторых, не забывайте про правильный знак для каждой операции. Я понимаю, что следующий текст Вас оскорбит до глубины души, но просто так, на всякий случай – таблица знаков для каждой операции ☺:

|  |  |  |
| --- | --- | --- |
| **№** | **Операция** | **Знак** |
| 1 | Сложение | + |
| 2 | Вычитание | - |
| 3 | Умножение | \* |
| 4 | Остаток от деления | / |

**5. Секция ответов.** Ответы представляют собой четыре радио-кнопки, среди которых обязательно один ответ должен быть правильным. Все остальные должны быть сгенерированы ***случайным образом***.

Также обратите внимание на следующий важный момент в этом пункте:

Если вопрос уже был сгенерирован JSP страницей, то управление должно передаваться на эту же страницу, как только пользователь нажимает на одну из радио-кнопок с ответами.

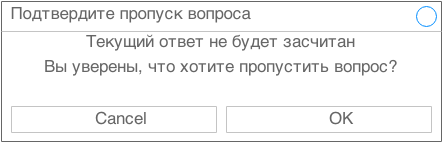
То есть, специальной submit кнопки, для того, чтобы отправить правильный ответ, на форме быть не должно. Вот тут вы и сможете прочувствовать всю мощь JavaScript-ового обработчика onClick.

**6. Комментарий к ответу.** Строка из пункта №6 должна отображаться на форме только в тех случаях, когда на предыдущем шаге JSP кроме генерации нового вопроса еще и ***проверяла ответ пользователя***. Должно быть две разных строки в зависимости от того, правильно пользователь ответил или нет. В качестве примера могут выступать следующие строки:

* *«Поздравляем! Это правильный ответ!»* – если пользователь ответил верно.
* И *«К сожалению, ответ не верный. Сосредоточьтесь!»* – в противном случае.

Но это только для примера. Вы можете придумать свои тексты.

**7. Кнопка «Следующее задание»** должна отобразить для пользователя следующий диалог для подтверждения:

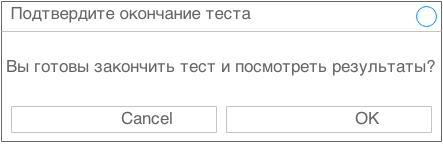


Если пользователь нажимает кнопку «Ok», то управление должно передаться JSP странице, которая засчитывает текущий ответ для пользователя, как ***неверный*** (обновляя, соответственно статистику теста из пункта №2) и генерирует новый вопрос.

Если же пользователь нажимает кнопку «Cancel», то управление должно остаться на текущей странице.

В этом пункте Вам так же придется использовать обработчик onClick, только уже в несколько нестандартном виде. В принципе, вопрос о том, как отменить submit формы, если пользователь нажимает кнопку «Cancel» достаточно легко гуглится, но если вдруг здесь возникнут проблемы, пожалуйста, не стесняйтесь и не медлите – сразу спрашивайте.

**8. Кнопка «Закончить тест»** опять же должна спросить у пользователя, согласен ли он закончить тестирование:



Если пользователь нажимает кнопку «Ok», то управление передается JSP, которая в ответ выдает пользователю страницу с результатами тестирования. Эта страница может выглядеть так, как показано на рис. 2:



Рис. 2. Страница результатов теста.

Как Вы видите, страница должна отобразить процент правильных ответов и строку соответствующую этому проценту. Таблица соотношений приведена ниже:

|  |  |  |
| --- | --- | --- |
| **№** | **Процент правильных ответов** | **Сообщение** |
| 1 | 0% – 25% | Неудовлетворительно |
| 2 | 26% – 50% | Удовлетворительно |
| 3 | 51% – 75% | Хорошо |
| 4 | 76% – 100% | Отлично |

Ну а если пользователь нажимает на кнопку **«Пройти тест еще раз»**, то Ваша JSP страница начинает делать заново все то, что было написано выше.