Использование GET-параметров в ссылках. Виртуальная клавиатура.

Лабораторная работа № А-3.

ЦЕЛЬ РАБОТЫ

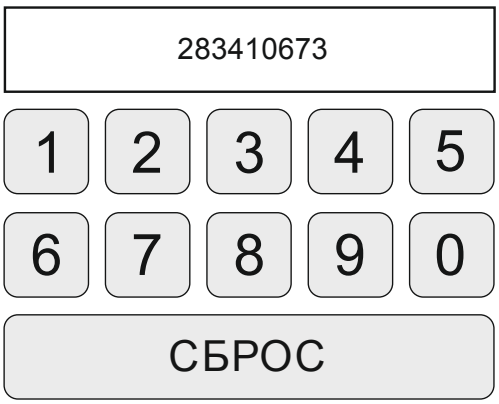
Изучение принципов работы с GET-параметрами, возможностей по вводу и хранению данных с помощью параметров, понимание принципов построения многостраничных веб-сервисов и динамических страниц сайта.

ПРОДОЛЖИТЕЛЬНОСТЬ

2 академических часа (1 занятие)

РЕЗУЛЬТАТ РАБОТЫ

Размещенный на Веб-сервере и доступный по протоколу http документ (страница сайта) с кнопками цифр от 0 до 9, кнопкой сброса и окном просмотра результата.



ДОПОЛНИТЕЛЬНЫЕ ТРАБОВАНИЯ К РАБОТЕ

Вся функциональность реализуется исключительно с помощью PHP. Кнопки формируются в виде ссылок с необходимыми стилями. Окно просмотра результата формируется из блока (тег `<div>`), выключка текста в блоке – по центру. Внешний вид кнопок и окна просмотра результата соответствует рисунку. Скрипт должен функционировать следующим образом.

- При первой загрузке в браузер окно просмотра результата пусто.
- При нажатии на кнопку "СБРОС" страница перезагружается, в окне просмотра результата ничего нет.
- При нажатии кнопки с цифрой страница перезагружается, к строке в окне просмотра результата добавляется соответствующая цифра.
- В подвале отображается общее число нажатий любых кнопок с момента первой загрузки страницы.

РЕКОМЕНДАЦИИ К СТРУКТУРЕ ПРОГРАММЫ

Основной задачей данной лабораторной работы является такая программа, которая бы формировала нужный результат без промежуточного хранения данных. Если бы такое хранилище у нас имелось, то описание алгоритма выглядело бы следующим образом.

- Если хранилища еще нет (первая загрузка страницы) – создаем пустое хранилище.
- Если нажата кнопка с цифрой – добавить в хранилище соответствующую цифру.
- Если нажата кнопка сброс – очистить хранилище.
- Вывести содержимое хранилища.

Предположим, что переменная `$STORE` как раз и является нашим хранилищем. Тогда программу можно представить следующим образом.

Листинг А-3. 1

```
<?php                                   // начало PHP-программы
      if( !isset($STORE) ) $STORE = '';   // если хранилища еще не – создаем его

      if( isset($_GET['num']) )               // если была нажата кнопка с цифрой
          $STRORE .= $_GET['num'];            // сохранить цифру в хранилище

      if( isset($_GET['reset']) )         // если была нажата кнопка СБРОС
          $STRORE = '';                   // очистить хранилище

      echo '<div class="result">'.$STORE.'</div>';  // выводим содержимое хранилища
?>
<a href="/?num=1">1</a>
...
<a href="/?num=0">0</a>
<a href="/?reset=Y">СБРОС</a>
```

Кнопки реализованы в виде ссылок, как и требуется в задании. Причем в адресе каждой ссылки в качестве GET-параметра передается значение кнопки (цифра или "*reset*"). При нажатии кнопки ее значение попадает в PHP и может быть прочитано в суперглобальном массиве `$_GET` с соответствующим ключом: если в адресе ссылки указан параметр "`some_param_name=...`", то его значение будет доступно в `$_GET[some_param_name]`. В данном примере кнопки с цифрами используют один и тот же параметр *num*, но с разными значениями: если нажата одна из них, то в PHP появляется элемент массива *`$_GET['num']`*, хранящий переданной значение. Кнопка "СБРОС" использует параметр *reset* – если нажали ее, то появляется элемент массива *`$_GET['reset']`*.

Таким образом, обработка нажатия кнопки с цифрой скриптом очень проста (обработка нажатия кнопки СБРОС производится абсолютно аналогично):

- если в массиве есть элемент с ключом *num*, значит был передан параметр с именем *num*;
- если передан параметр *num*, значит был осуществлен переход по ссылке, которая его содержит;
- если был осуществлен переход по такой ссылке – значит была нажата кнопка с цифрой;
- если была нажата кнопка с цифрой – значит значение параметра и есть цифра на кнопке;
- цифру на нажатой кнопке необходимо добавить в хранилище.

В данном примере используются два параметра: *num* и *reset*. Но исходя из логики программы, если передан один параметр, то другой не будет передан никогда. Поэтому, имеет смысл заменить два параметра одним, назовем его *key*.

Листинг А-3. 2

```
<?php                                   // начало PHP-программы
      if( isset($_GET['key']) )               // если кнопка была нажата
      {
          if( $_GET['key'] == 'reset' )   // если нажата кнопка СБРОС
               $STORE = '';                   // очистить хранилище
          else                            // если нажата другая кнопка
               $STRORE .= $_GET['key'];       // сохранить цифру в хранилище
      }
      else
          $STORE = '';    // если хранилища еще не – создаем его
echo '<div class="result">'.$STORE.'</div>';    // выводим содержимое хранилища
?>
<a href="/?key=1">1</a>
...
<a href="/?key=0">0</a>
<a href="/?key=reset">СБРОС</a>
```

В отличии от предыдущего примера, получение параметра *key* (наличие элемента массива *`$_GET['key']`*) означает что была нажата либо цифра, либо кнопка "СБРОС". Поэтому после проверки наличия параметра, необходимо проверить его значение: если это "reset" – была

нажата кнопка "СБРОС", иначе – была нажата кнопка с цифрой. Кроме того, если параметр не передан вовсе – это означает что кнопка не была нажата вовсе, что возможно только в одном случае: первая загрузка страницы. Поэтому создание параметра переносится из начала программы. Получившийся код гораздо более оптимален и понятен, а значит вероятность логической ошибки в программе уменьшается.

К несчастью, в данной лабораторной работе мы не можем пользоваться хранилищами данных, поэтому придется "накапливать" информацию в параметрах. Итак, при каждом вызове страницы, перед тем как обработать переданные параметры, нам необходимо знать результат предыдущей обработки, т.е. ту строку, которая выводилась до этого. В листингах А-3.1 и А-3.2 для этого использовалось хранилище, но оно отсутствует. Единственный выход – ПЕРЕДАВАТЬ В ПРОГРАММУ РЕЗУЛЬТАТ ПРЕДЫДУЩЕЙ ОБРАБОТКИ КАК ЕЩЕ ОДИН ПАРАМЕТР.

Листинг А-3. 3

```
<?php                                  // начало PHP-программы
      $STORE='';                            // создаем пустое хранилище
      if( isset($_GET['store']) )           // если передано предыдущее значение
            $STORE= $_GET['store'];         // сохраняем его в хранилище

      if( isset($_GET['key']) )                 // если кнопка была нажата
      {
          if( $_GET['key'] == 'reset' )   // если нажата кнопка СБРОС
               $STORE = '';                     // очистить хранилище
          else                          // если нажата другая кнопка
               $STRORE .= $_GET['key'];         // сохранить цифру в хранилище
      }

      echo '<div class="result">'.$STORE.'</div>';  // выводим содержимое хранилища
?>
<a href="/?key=1&store=<?php echo $STORE; ?>">1</a>
...
<a href="/?key=0&store=<?php echo $STORE; ?>">0</a>
<a href="/?key=reset&store=<?php echo $STORE; ?>">СБРОС</a>
```

Это и делается в начале программы А-3.3: создается пустое хранилище и, если был передан параметр *store* с информации об его предыдущем содержимом – оно переносится в него. Интересно, что, если информация передана – значит была нажата какая-либо кнопка. Следовательно, если информация не была передана, то кнопка не была нажата, т.е. была осуществлена первая загрузка страницы. В этом случае хранилище все равно будет проинициализировано, а значит создание пустого хранилища после проверки на отсутствие параметра *key* – дублирование функционала и уже не нужно.

Для собственно передачи предыдущего значения в ссылки кнопок добавляется второй параметр, который и хранит значение хранилища. Для этого адрес ссылок частично формируется динамически: в ссылку добавляется текущее вычисленное значение, которое для следующей загруженной страницы станет предыдущим.

Оптимизируем рассмотренный код, сократив по возможности передаваемые параметры, используемые переменные и т.д.

Листинг А-3. 4

```
<?php                                  // начало PHP-программы
      if( !isset($_GET['store']) )          // если НЕ передано предыдущее значение
          $_GET['store'] = '';              // создаем пустое хранилище
     else                               // иначе
      if( isset($_GET['key']) )                 // если кнопка была нажата
          $_GET['store'].= $_GET['key'];        // сохранить цифру в хранилище

     // выводим содержимое хранилища
      echo '<div class="result">'.$_GET['store'].'</div>';
?>
```

```
<a href="/?key=1&store=<?php echo $_GET['store']; ?>">1</a>
...
<a href="/?key=0&store=<?php echo $_GET['store']; ?>">0</a>
<a href="/">СБРОС</a>
```

Первое что следует сделать – убрать переменную `$STORAGE`, которая в программе полностью дублирует элемент массива `$_GET['store']`. Это упрощает начальную инициализацию хранилища: если параметр не передан, то мы инициализируем его пустой строкой. Кроме того, нажатие кнопки "СБРОС" на данном этапе полностью соответствует начальной загрузке страницы, поэтому эта ссылка не передает никаких параметров вовсе. Следовательно, и обработку значения *reset* тоже можно убрать.

Убедитесь, что если вручную в URL страницы передать любое значение в параметре *key*, то оно добавится в окно просмотра результата. Самостоятельно доработайте программу согласно требованиям лабораторной работы.

СПРАВОЧНАЯ ИНФОРМАЦИЯ

Передача одного параметра в ссылке	`<a href="?param=value"></a>` `<?php echo $_GET['param'];  // выводит value  ?>`
Передача двух параметров в ссылке	`<a href="?p1=v1&p2=v2"></a>` `<?php echo $_GET['p1'].'_'.$_GET['p2'];  // v1_v2  ?>`
Передача трех параметров в ссылке	`<a href="?p1=v1&p2=&p3=3"></a>` В качестве значения параметра p2 передана пустая строка
Существование переменной	`isset( $v ); // true если переменная была инициализирована`
Наличие в массиве элемента с ключом	`array_key_exists( $key, $array )`

КОНТРОЛЬНЫЕ ВОПРОСЫ К ЛАБОРАТОРНОЙ РАБОТЕ

Для успешной защиты работы помимо соответствующего требованиям результата необходимо уверенно отвечать на нижеперечисленные и другие вопросы, а также на контрольные вопросы всех предыдущих лабораторных работ.

1. Как передать GET-параметр через ссылку?
2. Что такое $_GET?
3. Как передать в ссылке несколько параметров?
4. Как проверить существование переменной?
5. Как узнать, был ли передан параметр в скрипт?
6. Как узнать значение переданного в сценарий параметра?
7. Изменится ли работа программы А-3.1, если между двумя условными операторами поставить else? Если да – то как?
8. Изменится ли работа программы А-3.4, если между двумя условными операторами убрать else? Если да – то как?
9. Изменится ли работа программы А-3.4, если убрать второй условный оператор, оставив else? Если да – то как?