Внедрение PHP-сценарисв в HTML-документ. Передача параметров формы PHP-сценарию. Передача значений переменных, соответствующих кнопкам формы

Задание 1. Создайте форму ввода данных о пользователе (ФИО, e-mail, телефон). Напишите скрипт PHP, который проверяет правильность заполнения полей формы.

Задание 2. Реализуйте тест:

ЗНАЕТЕ ЛИ ВЫ СЕБЯ. Ученые установили, что если из двух полушарий ведущим является правое полушарие мозга, то у человека преобладает эмоциональная сфера. Если ведущим оказывается левое полушарие, то у человека аналитический склад ума преобладает над эмоциональностью. Предлагаемый тест как раз и позволяет выявить ведущее полушарие. Это является признаком врожденным и, как правило, не меняется до конца жизни. (Правда/ ложь)

Вопрос N 1. Переплетите пальцы рук и вы заметите, что сверху всегда оказывается один и тот же палец, если левый - вы человек эмоциональный, правый - у вас преобладает аналитический склад ума.

Вопрос N 2. Попробуйте "прицелиться", выбрав себе мишень и глядя на нее через своеобразную мушку - карандаш или ручку. Правый ведущий глаз говорит о твердом, настойчивом, более агрессивном характере, левый - о мягком и уступчивом.

Вопрос N 3. Если при переплетении рук на груди наверху оказывается левая рука, то вы способны к кокетству, правая - склонны к простоте и простодушию.

Вопрос N 4. Если удобнее хлопать правой рукой, можно говорить о решительном характере, левой - вы часто колеблетесь, прежде чем принять решение.

ПППП - для вас характерен консерватизм, ориентация на общепринятое мнение (на стереотип). Вы не любите конфликтовать, спорить и ссориться.

ПППЛ - определяющая черта вашего характера - нерешительность.

ППЛП - для вас характерны кокетство, решительность, чувство юмора, артистизм. При общении с вами необходимы юмор и решительность. Очень контактный тип характера. Этот тип у женщин встречается наиболее часто.

ППЛЛ - редкий тип характера. Мягкий. Наблюдается некоторое противоречие между нерешительностью (левое аплодирование) и твердостью характера (правый ведущий глаз).

ПЛПП - тип характера, сочетающий аналитический склад ума и мягкость. Чаще встречается у женщин - тип "деловой женщины". Медленное привыкание, осторожность, в отношениях терпимость и некоторая холодность.

ПЛПЛ - слабый и самый редкий тип характера. Обладатели такого харак- тера беззащитны, подвержены различному влиянию. Встречается, как правило, у женщин.

ЛППП - такое сочетание встречается очень часто. Основная черта -эмоциональность, сочетающаяся с недостаточной настойчивостью.

ЛППЛ - для вас характерны мягкость, наивность. Требуете особого, внимательного отношения к себе - тип "маленькой королевы".

ЛЛПП - вам присущи дружелюбие и простота, некоторая разбросанность интересов, склонность к самоанализу.

ЛЛПЛ - в вашем характере преобладают простодушие, мягкость, доверчивость. Очень редкий тип характера, практически не встречается у мужчин.

ЛЛЛП - вы эмоциональный, энергичный и решительный человек, но часто наспех принимаете решения, которые приносят серьезные осложнения. Очень важен дополнительный тормозной механизм. Мужчины с таким характером менее эмоциональны.

ЛЛЛЛ - у вас антиконсервативный тип характера. Способны на старые вещи взглянуть по-новому. Характерны эмоциональность, эгоизм, упрямство, иногда переходящее в замкнутость.

ЛПЛП - самый сильный тип характера. Вас трудно в чем-либо убедить. Вы с трудом меняете свою точку зрения. Но в то же время вы энергичны, упорно добиваетесь поставленных целей.

ЛПЛЛ - вы настойчивы в достижении поставленных целей. Обладатели такого характера - люди неуступчивые, переубедить их порой оказывается невозможно. Они склонны к самоанализу, с трудом находят себе новых друзей.

ПЛЛП - у вас легкий характер. Вы счастливо умеете избегать конфликтов, любите путешествовать. Легко находите друзей. Однако вы часто меняете свои увлечения.

ПЛЛЛ - вам присущи непостоянство и независимость, желание все сделать самому. Способность анализировать помогает вам успешно решать сложные задачи. Обычно вы кажетесь мягким, но становитесь требовательным и даже жестоким, когда доходит до дела.

Задание 3. Реализуйте тест:

Я И АЛКОГОЛЬ. За каждый положительный ответ начисляется 1 балл. Постарайтесь честно ответить на следующие вопросы:

1. После ссоры в семье, после выговора начальника ищете ли Вы успокоение в спиртных напитках ?

2. Замечаете ли Вы, что стали в последнее время пить больше ?

3. Случалось ли Вам, проснувшись утром после выпивки, не помнить, что было вчера ?

4. Когда пьете в компании, не стараетесь ли Вы незаметно выпить побольше ?

5. Случались ли в вашей жизни ситуации, когда без алкоголя Вы чувствовали себя неуверенно ?

6. Стремитесь ли Вы опорожнить первую рюмку быстрее, чем делали это раньше ?

7. Приходите ли Вы в негодование, когда близкие осуждают ваши выпивки?

8. Замечаете ли Вы у себя провалы памяти ?

9. Всегда ли у Вас находятся причины, оправдывающие выпивку ?

10. Часто ли Вы жалеете о том, что сделали или сказали в пьяном виде ?

11. Возникает ли у Вас желание контролировать количество потребляемых спиртных напитков ?

12. Часто ли Вы нарушаете данное себе обещание - пить меньше или вообще бросить пить?

13. Пытались ли Вы бросить пить ?

14. Стремитесь ли Вы к тому, чтобы ваша семья и друзья не видели Вас пьяным ?

15. Замечали ли Вы, что из-за активного потребления спиртного у Вас появились финансовые затруднения и проблемы на работе ?

16. Увеличилось ли число людей, которые, как Вам кажется, несправедливо к Вам относятся ?

17. Дрожат ли у Вас руки даже после небольшой выпивки

18. Бывает ли, что Ваш запой длится несколько дне ?

19. Чувствуете ли Вы иногда депрессию и нежелание жить?

20. Бывают ли у Вас после пьянки слуховые и зрительные галлюцинации?

1-7: ранняя стадия, которая длится обычно 10-15 лет. **8-17**: средняя стадия, длится обычно 2-5 лет. **18-20**: последняя стадия алкоголизма.

ТЕОРИЯ

Форма запроса клиента

Клиент отсылает серверу запрос в одной из двух форм: в полной или сокращенной. Запрос в первой форме называется соответственно полным запросом, а во второй форме – простым запросом.

Простой запрос содержит метод доступа и адрес ресурса. Формально это можно записать так:

```
<Простой-Запрос> := <Метод> <символ пробел>
    <Запрашиваемый-URL> <символ новой строки>
```

В качестве метода могут быть указаны GET, POST, HEAD, PUT, DELETE и другие.

В качестве запрашиваемого URL чаще всего используется URL-адрес ресурса.

Пример простого запроса:

```
GET http://www.ru/
```

Здесь GET – это метод доступа, т.е. метод, который должен быть применен к запрашиваемому ресурсу, а http://www.ru/ – это URL-адрес запрашиваемого ресурса.

Полный запрос содержит строку состояния, несколько заголовков (заголовок запроса, общий заголовок или заголовок содержания) и, возможно, тело запроса. Формально общий вид полного запроса можно записать так:

```
<Полный запрос> := <Строка Состояния>
    (<Общий заголовок>|<Заголовок запроса>|
    <Заголовок содержания>)
    <символ новой строки>
    [<содержание запроса>]
```

Квадратные скобки здесь обозначают необязательные элементы заголовка, через вертикальную черту перечислены альтернативные варианты.

Элемент `<Строка состояния>` содержит метод запроса и URL ресурса (как и простой запрос) и, кроме того, используемую версию протокола HTTP. Например, для вызова внешней программы можно задействовать следующую строку состояния:

```
POST http://www.ru/cgi-bin/test HTTP/1.0
```

В данном случае используется метод POST и протокол HTTP версии 1.0.

В обеих формах запроса важное место занимает *URL* запрашиваемого ресурса. Чаще всего URL используется в виде URL-адреса ресурса. При обращении к серверу можно применять как полную форму URL, так и упрощенную.

Полная форма содержит тип протокола доступа, адрес сервера ресурса и адрес ресурса на сервере (рисунок 4.2).

В сокращенной форме опускают протокол и адрес сервера, указывая только местоположение ресурса от корня сервера. Полную форму используют, если возможна пересылка запроса другому серверу. Если же работа происходит только с одним сервером, то чаще применяют сокращенную форму.

http://www.ru/cgi-bin/test?var1=value1&var2=value2

Figure: Полная форма URL

Методы

Как уже говорилось, любой запрос клиента к серверу должен начинаться с указания метода. Метод сообщает о цели запроса клиента. Протокол HTTP поддерживает достаточно много методов, но реально используются только три: POST, GET и HEAD.

Метод GET

позволяет получить любые данные, идентифицированные с помощью URL в запросе ресурса. Если URL указывает на программу, то возвращается результат работы программы, а не ее текст (если, конечно, текст не есть результат ее работы). Дополнительная информация, необходимая для обработки запроса, встраивается в сам запрос (в строку статуса). При использовании метода GET в поле тела ресурса возвращается собственно затребованная информация (текст HTML-документа, например).

Существует разновидность метода GET – условный GET. Этот метод сообщает серверу о том, что на запрос нужно ответить, только если выполнено условие, содержащееся в поле if-Modified-Since заголовка запроса. Если говорить более точно, то тело ресурса передается в ответ на запрос, если этот ресурс изменялся после даты, указанной в if-Modified-Since.

Метод HEAD

аналогичен методу GET, только не возвращает тело ресурса и не имеет условного аналога. Метод HEAD используют для получения информации о ресурсе. Это может пригодиться, например, при решении задачи тестирования гипертекстовых ссылок.

Метод POST

разработан для передачи на сервер такой информации, как аннотации ресурсов, новостные и почтовые сообщения, данные для добавления в базу данных, т.е. для передачи информации большого объема и достаточно важной. В отличие от методов GET и HEAD, в POST передается тело ресурса, которое и является информацией, получаемой из полей форм или других источников ввода.

Использование HTML-форм для передачи данных на сервер

Для метода GET

При отправке данных формы с помощью метода GET содержимое формы добавляется к URL после знака вопроса в виде пар имя=значения, объединенных с помощью амперсанта &:

```
action?name1=value1&name2=value2&name3=value3
```

Здесь `action` – это URL-адрес программы, которая должна обрабатывать форму (это либо программа, заданная в атрибуте action тега form, либо сама текущая программа, если этот атрибут опущен). Имена `name1, name2, name3` соответствуют именам элементов формы, а `value1, value2, value3` – значениям этих элементов. Все специальные символы, включая `=` и `&`, в именах или значениях этих параметров будут опущены. Поэтому не стоит использовать в названиях или значениях элементов формы эти символы и символы кириллицы в идентификаторах.

Если в поле для ввода ввести какой-нибудь служебный символ, то он будет передан в его шестнадцатеричном коде, например, символ $ заменится на %24. Так же передаются и русские буквы.

Для полей ввода текста и пароля (это элементы input с атрибутом type=text и type=password), значением будет то, что введет пользователь. Если пользователь ничего не вводит в такое поле, то в строке запроса будет присутствовать элемент name=, где name соответствует имени этого элемента формы.

Для кнопок типа checkbox и radio button значение value определяется атрибутом VALUE в том случае, когда кнопка отмечена. Не отмеченные кнопки при составлении строки запроса игнорируются целиком. Несколько кнопок типа checkbox могут иметь один атрибут NAME (и различные VALUE), если это необходимо. Кнопки типа radio button предназначены для одного из всех предложенных вариантов и поэтому должны иметь одинаковый атрибут NAME и различные атрибуты VALUE.

В принципе создавать HTML-форму для передачи данных методом GET не обязательно. Можно просто добавить в строку URL нужные переменные и их значения.

```
http://www.ru/test.php?id=10&user=pit
```

В связи с этим у передачи данных методом GET есть один существенный **недостаток** – любой может подделать значения параметров. Поэтому не советуем использовать этот метод для доступа к защищенным паролем страницам, для передачи информации, влияющей на безопасность работы программы или сервера. Кроме того, не стоит применять метод GET для передачи информации, которую не разрешено изменять пользователю.

Несмотря на все эти недостатки, использовать метод GET достаточно удобно при отладке скриптов (тогда можно видеть значения и имена передаваемых переменных) и для передачи параметров, не влияющих на безопасность.

Для метода POST

Содержимое формы кодируется точно так же, как для метода GET, но вместо добавления строки к URL содержимое запроса посылается блоком данных как часть операции POST. Если присутствует атрибут ACTION, то значение URL, которое там находится, определяет, куда посылать этот блок данных. Этот метод, как уже отмечалось, рекомендуется для передачи больших по объему блоков данных.

Информация, введенная пользователем и отправленная серверу с помощью метода POST, подается на стандартный ввод программе, указанной в атрибуте action, или текущему скрипту, если этот атрибут опущен. Длина посылаемого файла передается в переменной окружения `CONTENT_LENGTH`, а тип данных – в переменной `CONTENT_TYPE`.

Передать данные методом POST можно только с помощью HTML-формы, поскольку данные передаются в теле запроса, а не в заголовке, как в GET. Соответственно и изменить значение параметров можно, только изменив значение, введенное в форму. При использовании POST пользователь не видит передаваемые серверу данные.

Основное преимущество POST запросов – это их большая безопасность и функциональность по сравнению с GET-запросами. Поэтому метод POST чаще используют для передачи важной информации, а также информации большого объема. Тем не менее не стоит целиком полагаться на безопасность этого механизма, поскольку данные POST запроса также можно подделать, например создав html-файл на своей машине и заполнив его нужными данными. Кроме того,

не все клиенты могут применять метод POST, что ограничивает варианты его использования.

При отправке данных на сервер любым методом передаются не только сами данные, введенные пользователем, но и ряд переменных, называемых переменными окружения, характеризующих клиента, историю его работы, пути к файлам и т.п. Вот некоторые из переменных окружения:

- `REMOTE_ADDR` – IP-адрес хоста (компьютера), отправляющего запрос;
- `REMOTE_HOST` – имя хоста, с которого отправлен запрос;
- `HTTP_REFERER` – адрес страницы, ссылающейся на текущий скрипт;
- `REQUEST_METHOD` – метод, который был использован при отправке запроса;
- `QUERY_STRING` – информация, находящаяся в URL после знака вопроса;
- `SCRIPT_NAME` – виртуальный путь к программе, которая должна выполняться;
- `HTTP_USER_AGENT` – информация о браузере, который использует клиент.

Обработка запросов с помощью PHP

Внутри PHP-скрипта существует несколько способов получения доступа к данным, переданным клиентом по протоколу HTTP. До версии PHP 4.1.0 доступ к таким данным осуществлялся по именам переданных переменных. Таким образом, если, например, было передано `first_name=Nina`, то внутри скрипта появлялась переменная `$first_name` со значением Nina. Если требовалось различать, каким методом были переданы данные, то использовались ассоциативные массивы `\$HTTP_POST_VARS` и `\$HTTP_GET_VARS`, ключами которых являлись имена переданных переменных, а значениями – соответственно значения этих переменных. Таким образом, если пара `first_name=Nina` передана методом GET, то

```
$HTTP_GET_VARS["first_name"]="Nina"".
```

Использовать в программе имена переданных переменных напрямую небезопасно. Поэтому было решено начиная с PHP 4.1.0 задействовать для обращения к переменным, переданным с помощью HTTP-запросов, специальный массив – `$_REQUEST`. Этот массив содержит данные, переданные методами POST и GET, а также с помощью HTTP cookies. Это суперглобальный ассоциативный массив, т.е. его значения можно получить в любом месте программы, используя в качестве ключа имя соответствующей переменной (элемента формы).

Пример

Figure: Пример внешнего вида формы.

Создана форма для регистрации участников заочной школы программирования. Тогда в файле action.php, обрабатывающем эту форму, можно написать следующее:

```
<?php $str = "Здравствуйте,
    ".$_REQUEST["first_name"]. "
    ".$_REQUEST["last_name"]."! <br>";
$str .="Вы выбрали для изучения курс по
    ".$_REQUEST["kurs"];
echo $str; ?>
```

Тогда, если в форму мы ввели имя «Вася», фамилию «Петров» и выбрали среди всех курсов курс по PHP, на экране браузера получим такое сообщение:

```
Здравствуйте, Вася Петров!
Вы выбрали для изучения курс по PHP
```

После введения массива `$_REQUEST` массивы `$HTTP_POST_VARS` и `$HTTP_GET_VARS` для однородности были переименованы в `$_POST` и `$_GET` соответственно, но сами они из обихода не исчезли из соображений совместимости с предыдущими версиями PHP. В отличие от своих предшественников, массивы `$_POST` и `$_GET` стали суперглобальными, т.е. доступными напрямую и внутри функций и методов.

Приведем пример использования этих массивов. Допустим, нам нужно обработать форму, содержащую элементы ввода с именами `first_name`, `last_name`, `kurs` (например, форму, приведенную выше). Данные были переданы методом POST, и данные, переданные другими методами, мы обрабатывать не хотим. Это можно сделать следующим образом:

```
<?php $str = "Здравствуйте,
    ".$_POST ["first_name"]."
    ".$_POST ["last_name"] ."! <br>";
$str .= "Вы выбрали для изучения курс по ".
        $_POST["kurs"];
echo $str; ?>
```

Тогда на экране браузера, если мы ввели имя «Вася», фамилию «Петров» и выбрали среди всех курсов курс по PHP, увидим сообщение, как в предыдущем примере:

```
Здравствуйте, Вася Петров!
Вы выбрали для изучения курс по PHP
```

Для того чтобы сохранить возможность обработки скриптов более ранних версий, чем PHP 4.1.0, была введена директива `register_globals`, разрешающая или запрещающая доступ к переменным непосредственно по их именам. Если в файле настроек PHP параметр `register_globals=On`, то к переменным, переданным серверу методами GET и POST, можно обращаться просто по их именам (т.е. можно писать `$first_name`). Если же `register_globals=Off`, то нужно писать

```
$_REQUEST["first_name"]
```
или
```
$_POST["first_name"],
$_GET["first_name"],
$HTTP_POST_VARS["first_name"],
$HTTP_GET_VARS["first_name"].
```
С точки зрения безопасности эту директиву лучше отключать (т.е. `register_globals=Off`). При включенной директиве `register_globals` перечисленные выше массивы также будут содержать данные, переданные клиентом.

Иногда возникает необходимость узнать значение какой-либо переменной окружения, например метод, использовавшийся при передаче запроса или IP-адрес компьютера, отправившего запрос. Получить такую информацию можно с помощью функции `getenv()`. Она возвращает значение переменной окружения, имя которой передано ей в качестве параметра.

```
<? getenv("REQUEST_METHOD");
    // возвратит использованный метод
echo getenv ("REMOTE_ADDR");
    // выведет IP-адрес пользователя,
    // пославшего запрос
?>
```

Как мы уже говорили, если используется метод GET, то данные передаются добавлением строки запроса в виде пар «`имя_переменной=значение к URL-адресу ресурса`». Все, что записано в URL после знака вопроса, можно получить с помощью команды

```
getenv("QUERY_STRING");
```

Благодаря этому можно по методу GET передавать данные в каком-нибудь другом виде. Например, указывать только значения нескольких параметров через знак плюс, а в скрипте разбирать строку запроса на части или можно передавать значение всего одного параметра. В этом случае в массиве `$_GET` появится пустой элемент с ключом, равным этому значению (всей строке запроса), причем символ «+», встретившийся в строке запроса, будет заменен на подчеркивание «_».

Методом POST данные передаются только с помощью форм, и пользователь (клиент) не видит, какие именно данные отправляются серверу. Чтобы их увидеть, хакер должен подменить нашу форму своей. Тогда сервер отправит результаты обработки неправильной формы не туда, куда нужно. Чтобы этого избежать, можно проверять адрес страницы, с которой были посланы данные. Это можно сделать опять же с помощью функции `getenv()`:

```
getenv("HTTP_REFERER");
```

Пример обработки запроса с помощью PHP

Нужно написать обработчки формы (см. выше) для регистрации участников заочной школы программирования и после регистрации отправить участнику сообщение.

```
<h2>Форма для регистрации студентов</h2>
<form action="1.php" method=POST>
Имя <br><input type=text name="first_name"
    value="Введите Ваше имя"><br>
Фамилия <br><input type=text name="last_name"><br>
E-mail <br><input type=text name="email"><br>
<p> Выберите курс, который вы бы хотели посещать:<br>
<input type=checkbox name='kurs[]' value='PHP'>PHP<br>
<input type=checkbox name='kurs[]' value='Lisp'>Lisp<br>
<input type=checkbox name='kurs[]' value='Perl'>Perl<br>
<input type=checkbox name='kurs[]' value='Unix'>Unix<br>
<P>Что вы хотите, чтобы мы знали о вас? <BR>
<textarea name="comment" cols=32 rows=5></textarea>
<input type=submit value="Отправить">
<input type=reset value="Отменить">
</form>
```

Figure: Пример кода формы.

Следует отметить, способ передачи значений элемента checkbox. Когда мы пишем в имени элемента kurs[], это значит, что первый отмеченный элемент checkbox будет записан в первый элемент массива kurs, второй отмеченный checkbox – во второй элемент массива и т.д. Можно, конечно, просто дать разные имена элементам checkbox, но это усложнит обработку данных, если курсов будет много.

Скрипт, который все это будет разбирать и обрабатывать, называется 1.php (форма ссылается именно на этот файл, что записано в ее атрибуте action). По умолчанию используется для передачи метод GET, но мы указали POST. По полученным сведениям от зарегистрировавшегося человека, скрипт генерирует соответствующее сообщение. Если человек выбрал какие-то курсы, то ему выводится сообщение о времени их проведения и о лекторах, которые их читают. Если человек ничего не выбрал, то выводится сообщение о следующем собрании заочной школы программистов (ЗШП).

```php
<?
// создадим массивы соответствий курс-время его
// проведения и курс-его лектор
$times = array("PHP"=>"14.30","Lisp"=>"12.00",
    "Perl"=>"15.00","Unix"=>"14.00");
$lectors = array("PHP"=>"Василий Васильевич",
"Lisp"=>"Иван Иванович", "Perl"=>"Петр Петрович", "Unix"=>"Семен Семенович"
define("SIGN","С уважением, администрация");
    // определяем подпись письма как константу
define("MEETING_TIME","18.00");
    // задаем время собрания студентов
$date = "12 мая"; // задаем дату проведения лекций
    //начинаем составлять текст сообщения
$str = "Здравствуйте, уважаемый " . $_POST["first_name"]
    . " " . $_POST["last_name"]."!<br>";
$str .= "<br>Сообщаем Вам, что ";
$kurses = $_POST["kurs"];    // сохраним в этой переменной
                                // список выбранных курсов
if (!isset($kurses)) { // если не выбран ни один курс
    $event = "следующее собрание студентов";
    $str .= "$event состоится $date ". MEETING_TIME . "<br>";
} else { // если хотя бы один курс выбран
    $event = "выбранные Вами лекции состоятся $date <ul>";
        //функция count вычисляет число элементов в массиве
    for ($i=0;$i<count($kurses);$i++){
                            // для каждого выбранного курса
        $k = $kurses[$i]; // запоминаем название курса
        $lect = $lect . "<li>лекция по $k в $times[$k]";
                            // составляем сообщение
        $lect .= " (Ваш лектор, $lectors[$k])";
    }
    $event = $event . $lect . "</ul>";
    $str .= "$event";
}
$str .= "<br>". SIGN; // добавляем подпись
echo $str; // выводим сообщение на экран
?>
```

Figure: Пример скрипта обработки формы.