Тема 5.2
«Экранные формы и их обработка»

Вступление

В теме, посвящённой HTML, мы уже затронули вопрос тегов для создания веб-форм.

Сейчас мы поговорим о том, какие бывают формы, и как их следует обрабатывать.

Итак…

Виды форм

Формы в общем случае делятся на два основных вида:

- однооконные формы – полностью расположены на одной странице;
- пошаговые формы – новые поля появляются (после загрузки новой страницы или в рамках старой с использованием JavaScript/AJAX) по мере заполнения уже показанных.

Проще всего реализовывается обработка однооконных форм и пошаговых форм, в которых отображение новых полей связано с загрузкой новой страницы.

Однооконные формы

Общий алгоритм обработки однооконных форм таков:

1. При заходе пользователя на страницу, содержащую форму, мы отслеживаем этот факт через проверки того, что массивы $_POST и/или $_GET пусты. Поскольку данные из формы чаще всего отправляются методом POST, чаще нам придётся работать с массивом $_POST.
2. После того, как форма сгенерирована, нам остаётся только ждать действий пользователя.
3. В процессе заполнения пользователем полей мы можем (и, в последнее время это используется всё чаще) проверять правильность ввода данных непосредственно в момент их ввода с использованием AJAX.
4. Когда пользователь всё заполнил и хочет отправить форму, мы должны для его удобства проверить с помощью JavaScript правильность заполнения полей и, в случае обнаружения ошибок, не позволять отправлять форму.

Одноооконные формы

5. Получив данные из формы на стороне сервера, мы должны проверить:
   - наличие в списке данных всех необходимых полей (и отсутствие лишних, причём наличие лишних полей может говорить о попытке взлома);
   - корректность данных во всех полях, причём с двух точек зрения: с точки зрения человека (так, например, не бывает фамилий «12345») и с точки зрения компьютера (так, например, ФИО вида «; drop table admins; ---» должно нас, мягко скажем, насторожить.

6. Если какие-то поля заполнены неверно, мы должны повторно сгенерировать и отправить пользователю форму, подставив во все поля (как правило, кроме полей с паролями) введённые пользователем значения и указав, какие именно значения введены неверно (и в чём ошибка).

7. Только после того, как мы убедились в том, что все данные введены верно, мы можем продолжать выполнять связанные с этими данными операции (запись в БД и т.п.)

Пошаговые формы

В том случае, если пошаговые формы работают по принципу «новая часть на новой странице», алгоритм их обработки, фактически, сводится в алгоритму обработки однооконных форм за одним неприятным исключением: «что делать, если пользователь вернётся на одну-две-три… страницы назад?»

Ответ на этот вопрос не однозначен и зависит от того, насколько мы хотим упростить такому пользователю жизнь.

В некоторых случаях всё может быть просто, когда браузер запоминает данные формы.

Но иногда он их может и не запомнить, или, наоборот, нам не надо позволять пользователю «возвращаться назад» (например, если пользователь проходит какой-то тест).

Пошаговые формы

В любом случае самым универсальным алгоритмом является такой:

1. На каждом шаге формы при её генерации формировать некое случайное значение и помещать в скрытое поле. При получении данных с текущего шага генерировать значение для следующего и ожидать его.
2. В случае, если мы на каком-то шаге получили неожиданное (как правило – старое) значение этого случайного числа – значит, пользователь что-то «нахимичил».
3. Если мы решили не давать пользователю возвращаться назад, неожиданное случайное значение даёт нам основание прервать его работу (возможно, начать заполнение всей формы заново).
4. Если мы решили дать пользователю возможность возвращаться к предыдущим шагам, мы можем с использованием механизма сессий фиксировать все уже введённые пользователем значения и автоматически подставлять их в тот или иной шаг формы таким образом, что пользователю остаётся только нажимать «Далее», пока он не доберётся до того шага, с которого начал «двигаться назад».
5. Более того, мы можем определить, сколько шагов пользователь уже выполнил и либо сразу автоматически показывать ему следующий (невыполненный) шаг формы, либо выдавать подсказку вида «Данные за шаги 1-7 получены. Кликните здесь, чтобы перейти к шагу 8.»

Пошаговые формы

Если наша форма генерируется «в одной странице», значит у нас используется AJAX. Фактически, это означает, что данные точно так же приходят к нам на сервер и обрабатываются так, словно они пришли из обычной формы.

Наиболее универсальным способом является полное перенесение логики обработки и генерации шагов на сторону сервера с тем, чтобы скрипт на AJAX просто показывал пользователю данные, полученные от сервера, а затем отправлял нам то, что ввёл пользователь.

В таком случае логика управления шагами полностью идентична описанной в предыдущем алгоритме.

Иногда форму передают в AJAX-скрипт сразу целиком, но показывают пользователю по частям, а лишь в самом конце отправляют её целиком на сервер.

Тогда логика обработки формы на стороне сервера сводится к алгоритму обработки однооконных форм за исключением того факта, что в случае ошибок в полях пользователю можно смело показывать новую пустую форму, т.к. наличие таких ошибок говорит нам о том, что он «мухлюет»: раз он увидел все шаги – значит, AJAX у него работает, а раз AJAX работает – значит, ошибки в полях пользователь должен был увидеть и исправить до отправки формы.

Ради осторожности и на всякий случай можно всё же указать, в каких полях были ошибки, но эта мера относится к разряду перестраховки.

Рекомендации и частые ошибки

Сейчас мы рассмотрим некоторые рекомендации по работе с формами и часто допускаемые ошибки.

1. «Одно поле – одно значение!» Часто разработчики (особенно начинающие) предлагают пользователю заполнить поле в некотором формате, например: «Введите фамилию, имя отчество – каждое слово с новой строки». Это не удобно ни пользователю (который может допустить ошибку), ни программисту: надо анализировать поле, разбирать его формат; а как указать, например, что ошибка в имени если пришли только две строки (фамилия и отчество) – и как вообще узнать, что пропущено именно имя?

И как красиво, например, подсветить красным имя, в котором, как мы определили, есть ошибка? Если бы на каждое значение, т.е. на фамилию, имя и отчество было отдельное поле – таких проблем бы не было.

К подобным же ошибкам относится заведение одного текстового поля с указанием «Введите дату в формате год-месяц-день». Вам такого в это поле на вводят, что мало не покажется.

Рекомендации и частые ошибки

2. «Показывайте примеры и заполняйте поля значениями по умолчанию!» Если поле допускает ввод значения по умолчанию (например, текущую дату) – введите его туда. Это сэкономит время и нервы пользователя. Если значение по умолчанию недопустимо, покажите пример: «Имя должно было не длиннее 20-ти символов, может содержать латинские буквы в нижнем и верхнем регистре, цифры и знаки подчёркивания, например: vasya_pupkin_2010».

3. «Никаких примеров паролей!» Окрылённые идеей из предыдущего пункта, некоторые разработчики включают в код статический пример пароля. Поверьте, половина «недальновидных пользователей» использует как раз этот пароль из примера, чем сильно облегчит жизнь злоумышленникам. Если уж очень хочется помочь пользователю сделать себе хитрый пароль, генерируйте каждый пример заново – динамически.

Рекомендации и частые ошибки

4. «Поясняйте, в чём ошибка!» Если какое-то поле заполнено с ошибкой, укажите, в чём именно она заключается, например: «Поле ФАМИЛИЯ содержит цифры, что не является допустимым». В случае со «сложными данными» даже опытный пользователь может долго гадать, что же хочет ваш скрипт, если нет такой подсказки.

5. «Подсвечивайте поля с ошибками!» Некоторые «горе-разработчики» не удосуживаются даже указать, в каком поле была ошибка. Они просто пишут «Некоторые поля заполнены неверно». Это развивает профессионализм и интуицию пользователей, но сильно их нервирует. Поле должно быть как-то помечено цветом, суть ошибки чётко указана, а рядом, если возможно, приведён либо пример правильного заполнения, либо вообще приведено обработанное значение формы, в котором ошибка устранена.

6. «При повторной генерации формы восстанавливайте введённые пользователем значения!» Потратить ещё пять минут на то, чтобы заново заполнить 30 полей – плохое развлечение.

Рекомендации и частые ошибки

7. «Используйте captcha (картинку для защиты от автоматической обработки форм) разумной сложности!» Да, можно «наворотить» capthca так, что её не распознает не только ни один «робот», но и далеко не каждый человек. Но делать этого не нужно, т.к. после 3-5 попыток понять, означает ли у вас символ «О» – ноль, русскую или английскую заглавную (?!) букву «О», пользователь уйдёт на сайт конкурентов.

8. «Обеспечьте работоспособность формы БЕЗ использования cookies!» Очень часто можно встретить ситуацию, когда при отключённых cookies скрипт считает, что пользователь что-то ввёл неверно или не ввёл вообще (чаще всего – captcha). Это очень плохо, т.к. cookies часто блокируются файрволом или вообще отключены.

9. «По возможности проверяйте поля на недопустимые значения ещё в процессе заполнения!» Так, например, выбранный пользователем логин можно проверить на уникальность ещё до отправки формы, используя AJAX. Точно так же можно проверить пароль на сложность подбора. Можно проверить на корректность введённую дату и т.п.

Рекомендации и частые ошибки

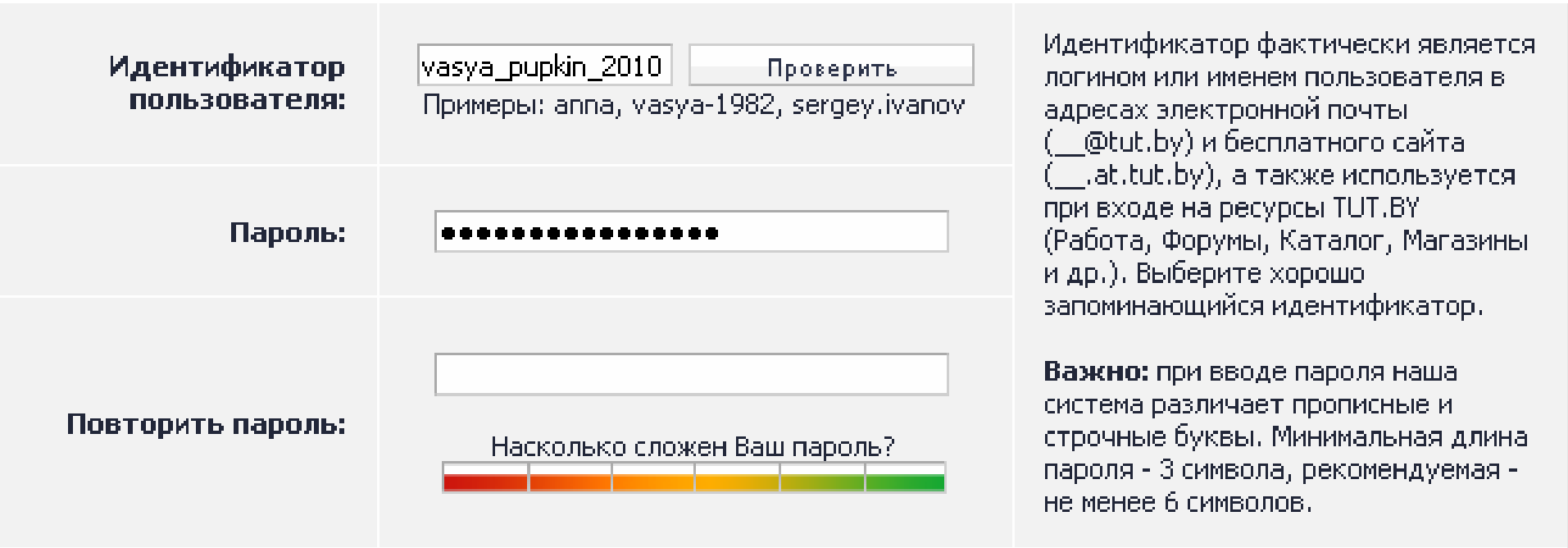
10.«Проверяйте значения полей не только на корректность формата, но и на смысловую корректность!» Так, например, дата рождения 10-03-1648 – корректна с точки зрения формата даты, но некорректна с точки зрения здравого смысла, если только к вам на сайт не заглянул кто-то из клана Маклаудов. Также бывает полезно проверить на существование введённый URL и домен в имени почтового ящика.

11.«Безопасность – хорошее дело!» Всё, что можно безболезненно зашифровать (пароли, номера кредитных карт, номер паспорта и т.п.) – сразу шифруйте (хэшируйте) и храните уже в зашифрованном виде. Хранение таких данных в открытом виде чревато судебными исками в случае взлома сайта.

А сейчас, как в старом анекдоте, – слайды…

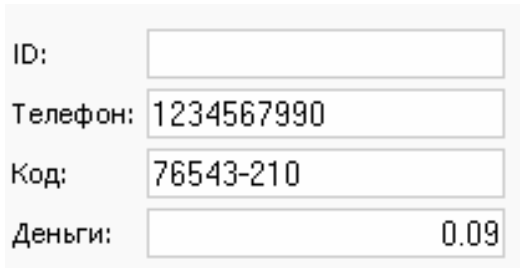
Пример хорошей формы

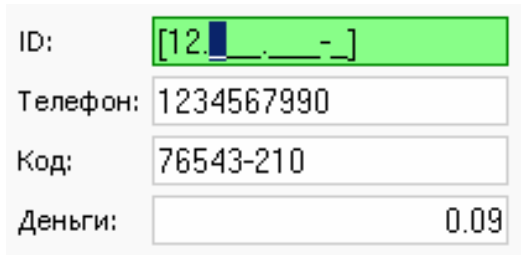
В этой форме есть возможность проверить, свободно ли для регистрации такое имя пользователя; приведены примеры имён пользователя; даны рекомендации и комментарии по поводу пароля:

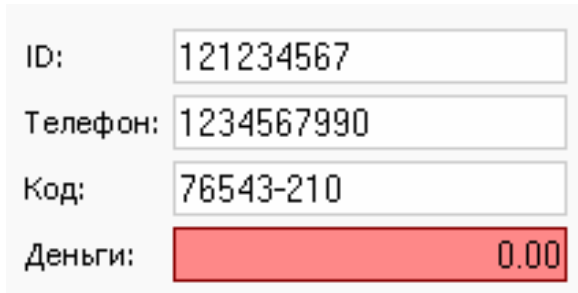


Пример хорошей формы

Здесь для проверки ввода значений используются налагаемые на поля маски, что позволяет пользователю видеть, что он вводит и насколько правильно:

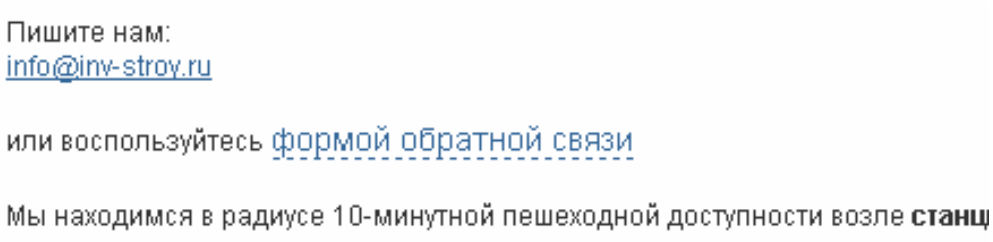




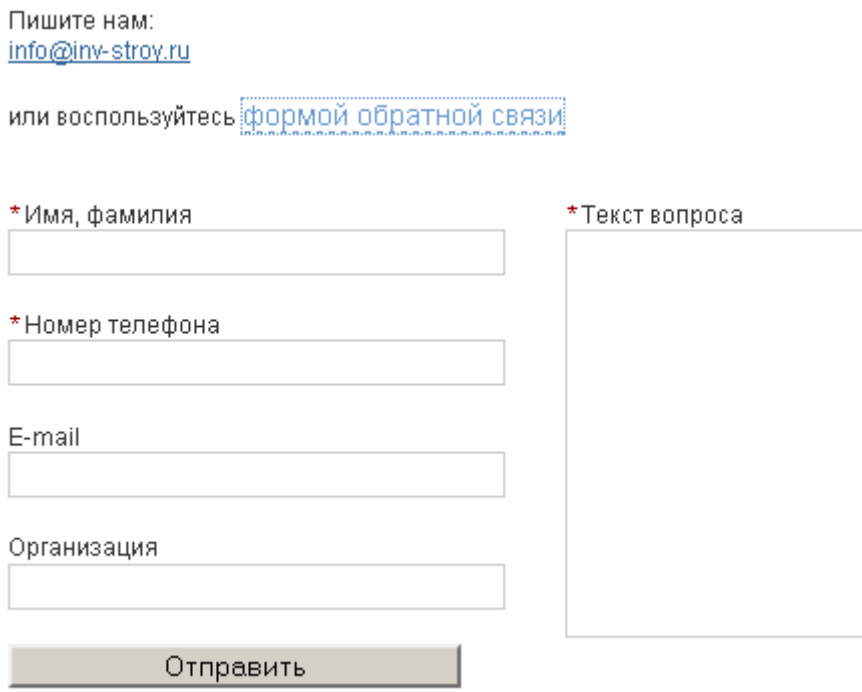


Пример плохой формы

Собственно, формы здесь и нет. Она появится, если кликнуть по ссылке «формой обратной связи». Если JavaScript у посетителя отключён, форма вообще никогда не появится…



Форма появится, если всё сработает:



Пример плохой формы

В этой форме – классика: пользователь должен угадать, как писать дату.

Запись для физических лиц:

Тренинг/семинар: Управление отделом продаж

Дата проведения:

ФИО:

Контактные данные:

Пример плохой формы

И снова классика – сначала надо угадывать, что и как…

А после нажатия «Отправить» форма исчезает и появляется…