

Тема 4

«Тестирование форм»

Вступление

Формы – неотъемлемый атрибут любого веб-ориентированного приложения, подразумевающего взаимодействие с человеком.

От качества реализации механизма работы с формами зависит множество аспектов качества веб-ориентированного приложения, от юзабилити до производительности и безопасности.

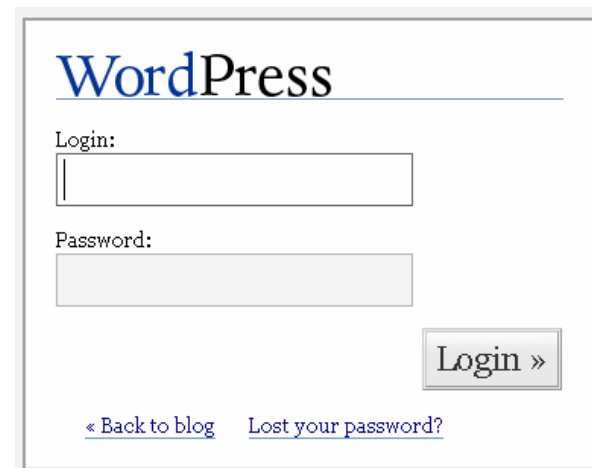
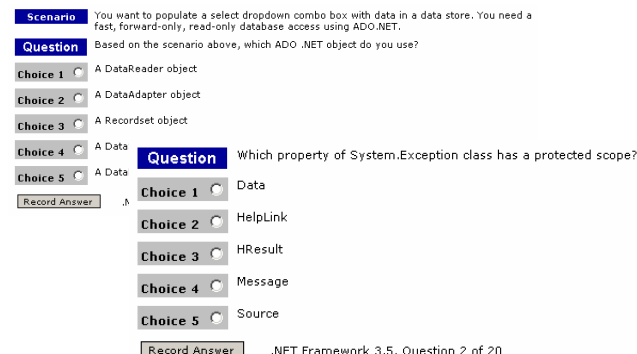


Виды форм

Определения

Формы в общем случае делятся на два основных вида:

- однооконные формы — полностью (со всеми своими полями) 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.

A screenshot of a WordPress login page. At the top, the word "WordPress" is displayed in a blue serif font, underlined. Below it, there are two input fields: "Login:" and "Password:". To the right of the password field is a "Login »" button. At the bottom, there are two links: "« Back to blog" and "Lost your password?".A screenshot of a .NET Framework 3.5 question interface. It shows a "Scenario" section with text about populating a dropdown menu. Below it is a "Question" section asking which ADO.NET object to use. There are five radio button choices: "Choice 1" (A DataReader object), "Choice 2" (A DataAdapter object), "Choice 3" (A Recordset object), "Choice 4" (A Data), and "Choice 5" (A Data). A "Record Answer" button is at the bottom. To the right, another "Question" section asks which property of System.Exception class has a protected scope. It has five radio button choices: "Choice 1" (Data), "Choice 2" (HelpLink), "Choice 3" (HResult), "Choice 4" (Message), and "Choice 5" (Source). A "Record Answer" button is at the bottom. The footer indicates ".NET Framework 3.5, Question 2 of 20".

Начнём с однооконных форм, т.к. здесь многие моменты будут более очевидны.

Путь (URL)

BAD: Вместо страницы корзины появляется надпись «Спасибо, операция выполнена успешно».

В лучшем случае после этого происходит редирект на страницу корзины.

В худшем – пользователю приходится нажимать в браузере **Back** и... видеть подхваченную из кэша страницу, на которой остались старые данные.

Вывод: работа с веб-приложением должна быть похожа на работу с «обычным настольным приложением» в плане своей последовательности и плавности.

Следует избегать лишних сообщений, действий и т.п. Это следует продумывать и при программировании, и при тестировании.

Месторасположение

2. Месторасположение

GOOD: Если форма расположена достаточно низко, нужно **сделать автоматическую прокрутку к форме**.

Например, так:

`action="/order/#form"`

(при обработке данных)

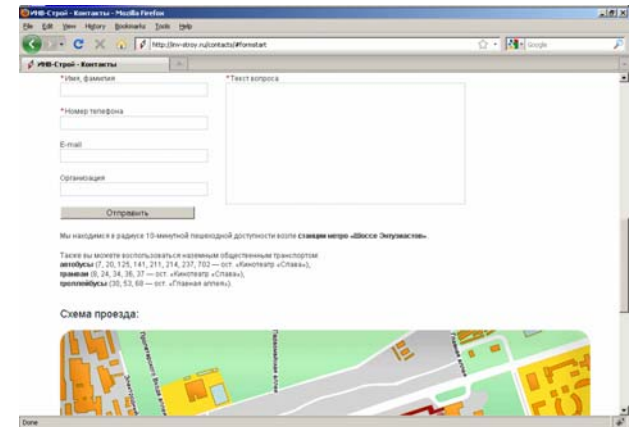
или:

`href="/order/#form"`

(при изначальном переходе к форме).

Перед самой формой пишем:

``



Месторасположение

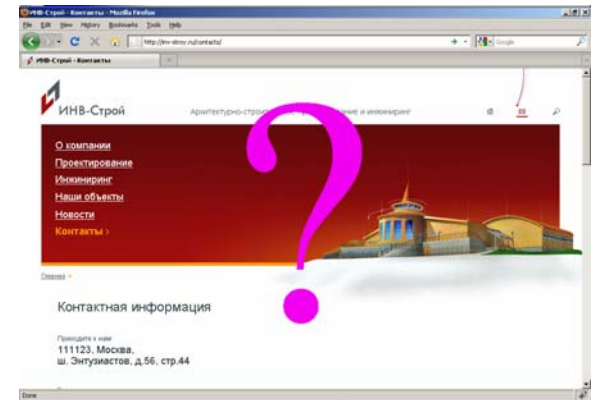
BAD: Страница с формой или результатами выполнения операции открывается в состоянии «вверху окна браузера – верх страницы».

Форма или некоторая информация о выполненной операции видна частично или не видна вообще.

Даже если пользователь догадается проскроллить страницу, это всё равно неудобно.

А пользователь может и не догадаться.

Вывод: для перехода к некоторой операции или следующему шагу некоторой операции должно быть достаточно выполнения **ОДНОГО** действия. Лишние клики, скроллинг и т.п. - всё это вредит как минимум юзабилити.



Ошибочные ситуации

3. Ошибочные ситуации.

GOOD: В случае, если пользователь ввёл некоторые данные некорректно (или не ввёл вообще), форма должна быть показана заново, и при этом:

а) все введённые данные (за исключением паролей и полей CAPTCHA) должны сохранять свои значения (ВСЕ ПОЛЯ! Т.е. и чек-боксы, и радиобаттоны, и списки и т.д.);

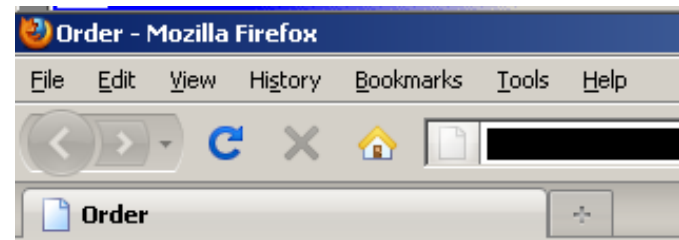
б) в удобном для восприятия месте (чаще всего – сразу над формой) следует чётко указать причину неудачи отправки данных; все неверно заполненные поля следует визуально выделить, а рядом с полем (при возможности) указать суть ошибки заполнения и подсказку по правильному заполнению.

Ошибочные ситуации

BAD: Иногда можно увидеть такое:

а) **Значения полей «обнуляются»** (поля или очищаются вообще или принимают некоторые значения по умолчанию. Особенно круто, когда так себя ведёт лишь часть полей.

б) **Причина неудачи отправки данных не объяснена** вообще или объяснена в стиле «Вы неверно заполнили некоторые поля». Развиваем интуицию пользователя?



Sorry, incorrect input...

Ошибочные ситуации

BAD (продолжение):

в) Ошибочно заполненные поля не указаны, суть ошибки не подчёркнута или подчёркнута способом, допускающим недопонимание.

Например: «Неверно указана дата». Какая и почему?

Может быть, пользователь допустил ошибку в самой дате (42 декабря) в поле «Дата рождения», а может – указал просто заведомо абсурдную дату (например: дату своего рождения как 01.01.2150).

Вывод: пользователь должен быть избавлен от необходимости быть телепатом, чтобы пользоваться вашим сайтом. Он также должен быть избавлен от необходимости повторно выполнять те действия, без повторного выполнения которых можно обойтись.

Поля, их значения и иже с ними

4. Поля, их значения и иже с ними.

Только что (в пункте 3) мы говорили об обработке ситуации «юзер что-то ввёл не так». Но, допустим, юзер ещё ничего не ввёл. Итак.

GOOD: Одно поле – одно значение.
Согласитесь, что заполнять, обрабатывать и тестировать такое (сейчас будет рисунок) – просто (пока опустим вопросы выравнивания, цветового и графического оформления и т.д.).

Поля, их значения и иже с ними

GOOD (продолжение):

Хорошая форма:

Обязательные для заполнения поля помечены (*)

ФИО

Фамилия (*): (до 30 символов, только русские буквы)

Имя (*): (до 30 символов, только русские буквы)

Отчество (*): (до 30 символов, только русские буквы)

Дата рождения

День (*): Месяц (*): Год (*): (1900-2009)

Поля, их значения и иже с ними

BAD: А вот такое и заполнять, и обрабатывать (парсить значения) и тестировать сложно.

Плохая форма:

Все поля обязательны:

ФИО:

Дата рождения:

Поля, их значения и иже с ними

GOOD: Если это возможно (и, особенно, если желательно!) – **помещайте рядом с полем пример его заполнения** (диапазон дат, пример имени пользователя и т.п.)

Это облегчает жизнь:

- посетителям сайта (**проще заполнять форму**);
- программистам (**проще понять, какие будут значения, какие на них можно наложить ограничения и т.д.**);
- тестировщикам (в случае с диапазонами пример заполнения **даёт готовые граничные условия для классов эквивалентности**, в случае «просто примера» сразу **есть входные данные для простого позитивного теста**).

Поля, их значения и иже с ними

BAD: Иногда из названия поля невозможно догадаться, что и в каком формате следует ввести.

И если у программиста и тестировщика есть спецификации (да в них лазить, как известно, часто – лень), то у пользователя остаётся только телепатия.

Пример: поле «Идентификатор товара». Ага. И какой он?

Может, «T001-24-н-87-Б»?

Или «123-6»?

Или «Мониторчик синенький»?

Product ID:

Поля, их значения и иже с ними

TOO BAD! Но есть одно исключение. Категорически не рекомендуется приводить примеры паролей (по той простой причине, что львиная доля «пользователей невдумчивых» именно такой пароль и использует).

Даже сгенерированный уникальный пароль лучше не приводить (он будет заэкширован как часть текста страницы).

Password:

~~(fe37jd32ug)~~

Вывод: уделяйте внимание тому, как организованы поля формы. Это, в конечном итоге, экономит время всем разработчикам и пользователям проекта.

Значения полей и спецсимволы

5. Значения полей и спецсимволы

GOOD: Следует чётко отдавать себе отчёт в том, что через поле, теоретически, могут быть введены любые символы. ЛЮ-БЫ-Е.

Из этого, конечно, следует необходимость тщательной фильтрации данных перед передачей их в БД или иной приёмник.

Но также следует фильтровать данные и правильно оформлять формы, имея в виду, что в случае неверного заполнения части полей данные будут снова показаны в форме (см. пункт 3).

Name: You can not use digits and special symbols in this field

Значения полей и спецсимволы

GOOD (продолжение):

Базовые идеи:

1. Помним и используем простое правило:
значения атрибутов тегов берутся в кавычки.
Без исключений.

2. Используем функции, позволяющие
отобразить данные как часть HTML-кода
(например, банальную функцию
`htmlspecialchars()` в PHP).

Значения полей и спецсимволы

BAD: Плохого по этому пункту можно написать много.

Но самые «весёлые» ситуации возникают, когда нарушены вышеназванные «базовые идеи».

Например, пользователь заполняет поле «Место проживания» так:

Отель "Минск", 2-й этаж

Форма сделана бездарно, фильтрации нет, а в коде прописано так:

```
<input type=text name=place value={value} />
```

Значения полей и спецсимволы

BAD (продолжение):

Когда плейсхолдер {value} будет заменён на реальное значение, код примет вид:

```
< ... value=Отель "Минск", 2-й этаж />
```

В переводе на русский:
value="Отель"

Всё. Данные о названии отеля и этаже утеряны.

Значения полей и спецсимволы

Вывод: переоценить правильное оформление HTML-кода и вдумчивую фильтрацию полученных от пользователя данных **едва ли возможно**.

JavaScript – Y/N ?

6. JavaScript в формах – «за» и «против»

GOOD: JavaScript, AJAX и иже с ними – прекрасные инструменты, которые позволяют значительно «оживить» приложение в плане эстетического восприятия, а также ускорить работу с формами.

Так, например, можно проверять значения полей на корректность ещё в процессе заполнения и/или перед отправкой данных из формы.

Можно выводить красивые подсказки, элегантно сообщать об ошибочных ситуациях и способах их решения.

A screenshot of a web form titled "User informations". It contains three text input fields: "Desired username :", "First name :", and "Last name :". Below the "Last name :" field is a dropdown menu with the text "* Please select an option". Several dark gray tooltip boxes with white text provide validation feedback: "* This field is required" and "* No special characters allowed" for the username field; "* This field is required" and "* Letters only" for the first name field; and "* This field is required" and "* Letters only" for the last name field. At the bottom, there are radio buttons and a "* This field is required" message.

JavaScript – Y/N ?

BAD: Но! **JavaScript может быть отключён**. Он также может «сглючить» в силу множества сложнопредсказуемых причин.

И что тогда? А тогда форма, рассчитанная на активное использование JavaScript и лишённая альтернативных способов «общения с пользователем» становится как минимум неинформативной (принимает "bad вид" из пункта 4), а как максимум – **неработоспособной**.

Вывод: НЕЛЬЗЯ полагаться ТОЛЬКО на JavaScript при работе с формами. Формы и механизмы их обработки должны оставаться полностью функциональными (и терять минимум юзабилити) при отключённом JavaScript.

Company Name:

Company URL:

First Name:

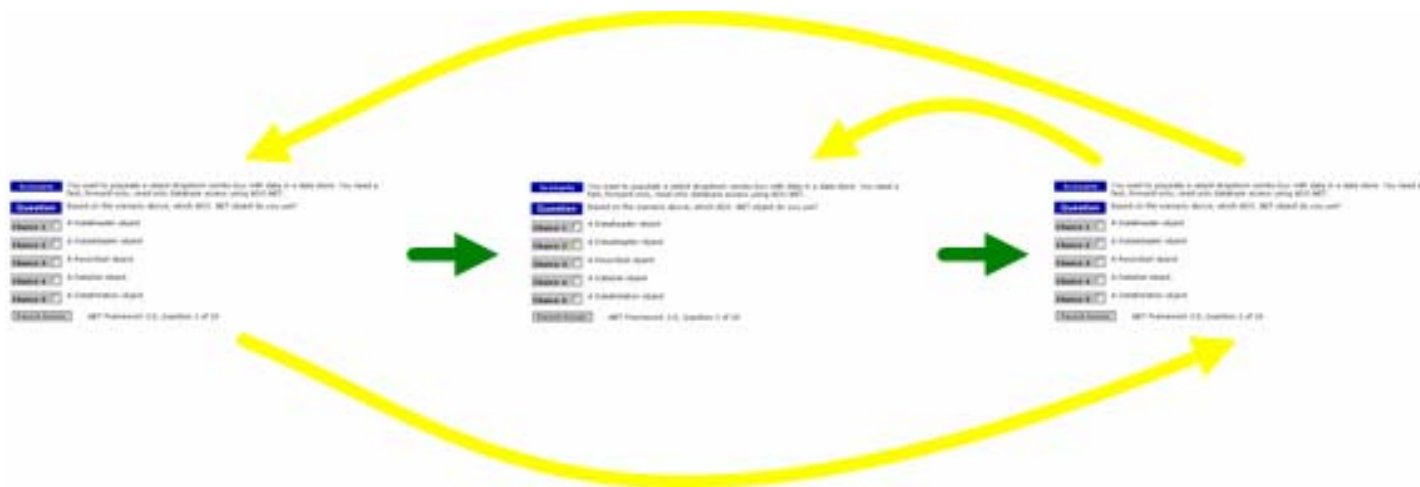
Last Name:

Пошаговые формы

7. Пошаговые формы

Основная беда пошаговых форм – неочевидность их функционирования при возврате на предыдущие шаги или восстановлении работы после обрыва связи.

Пункты 1-6 в полной мере относятся к каждому отдельному шагу пошаговой формы, а специфику сейчас рассмотрим.



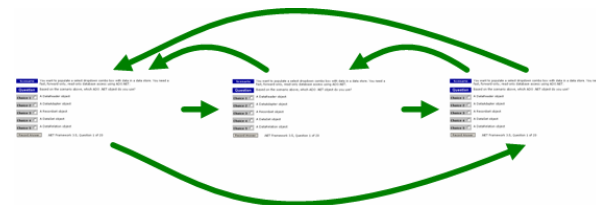
Пошаговые формы

GOOD: Следует однозначно давать понять пользователю, разрешено ли ему возвращаться на предыдущие шаги.

Если **разрешено** — этот механизм должен быть тщательно продуман с целью исключения возможности искажения и потери данных.

Если **запрещено** — должен быть механизм, явно запрещающий возврат, а в случае попыток совершить таковой должно появляться сообщение об ошибке.

Также, возможно, следует предпринять дальнейшие шаги (например, перенаправить пользователя на первый шаг формы для повторного заполнения).



Пошаговые формы

BAD: Механизм управления шагами работает «как поведёт».

В зависимости от браузера и иных факторов **возврат то не работает** («page has expired»), **то работает нестабильно** (данные искажаются и теряются), то вытворяет иные трюки и фокусы.



Вывод: пошаговые формы сложнее однооконных, а потому их разработке и тестированию следует уделять **повышенное внимание**.

Переходим к тестированию

Перед вами – краткий, «максимально универсализированный» чек-лист:

1. Расположение формы на экране перед заполнением, после неверного заполнения, после верного заполнения.
2. Сохранение/изменение URL при отправке данных из формы.
3. Отправка формы с незаполненными обязательными полями.
4. Отправка формы с неверно заполненными полями (числа вне диапазонов, строки превышают допустимую длину и т.п.)

Чек-лист

5. Отправка формы с «**нелогичными данными**» (например, дата рождения – в будущем).
6. Отправка формы с полями, содержащими **спецсимволы**.
7. **Восстановление значений полей** после отправки формы с неверно заполненными полями.
8. Информативность **сообщений об ошибках**.
9. **Функциональная сгруппированность полей** формы. **Информативность надписей, подсказок**.
10. **Работоспособность и юзабилити** формы с отключённым JavaScript.
11. **Использовать чек-листы по каждому стандартному полю** (текстовое, числовое, дата и т.п.)

Тест для проверки изученного

1. Что (в идеале) должен видеть пользователь сайта после успешного заполнения формы? Какие рекомендации относительно поведения приложения в этой ситуации вы можете дать?
2. Форма может располагаться не сразу вверху страницы, а ниже. Что в таком случае должно сделать приложение, чтобы обеспечить пользователю необходимый уровень удобства работы?
3. Если не заполнены какие-то обязательные поля формы или просто часть полей заполнена неверно – каким должно быть поведение приложения?
4. Какие вы можете дать рекомендации относительно организации полей формы?
5. Примеры значений каких полей не следует приводить?
6. Чем опасно отсутствие фильтрации значений полей?
7. О чём следует помнить, используя в формах JavaScript?
8. В чём особенность пошаговых форм? Какие проверки следует выполнить, чтобы избежать наиболее вероятных специфических для данного вида форм проблем?
9. Назовите несколько серьёзных проблем с формами (наличие которых позволяет сделать вывод о крайне низком качестве приложения).
10. Приведите несколько пунктов чек-листа формы.

Поля форм, стандартные тестовые случаи

Передача данных

Сейчас мы рассмотрим стандартные тестовые случаи для типичных полей форм.

Но прежде всего – проверим, **верным ли методом отправляются данные** из формы (атрибут «**method**» тега «**form**»).

Белый ящик:

`<form... method="post" ...>`

или

`<form... method="get" ...>`

Чёрный ящик:

отправьте данные и посмотрите на адресную строку браузера.

Post – никаких «неожиданных данных»:



Get – вы видите имена и значения полей:



Текстовые поля

Чаще всего используются три вида текстовых полей:


1) Однострочное поле:

`<input type="text" ...>`

A single-line text input field with a thin border, containing the text "Text Text Text".

2) Пароль:

`<input type="password" ...>`

A single-line password input field with a thin border, containing ten black dots to mask the text.

3) Многострочное поле:

`<textarea ...></textarea>`

A multi-line text area with a thin border, containing three lines of text: "Text1", "Text2", and "Text3". It has a vertical scrollbar on the right side.

Текстовые поля

Для всех текстовых полей стоит проверить:

1. Значение по умолчанию.
2. Заполнение поля корректными данными.
3. Оставить поле незаполненным (пустым).
4. Заполнить поле максимально разрешённым количеством СИМВОЛОВ.
5. Заполнить поле максимально разрешённым количеством СИМВОЛОВ + 1.
6. Заполнить поле такими спецсимволами как “ ‘ < > \ ; @ и т.п.
7. Заполнить поле только пробелами.
8. Вставить пробелы перед / в середине / в конце текста.
9. Ввести цифры.
10. Ввести данные на разных языках (в т.ч. одновременно).
11. Использовать разные кодировки (cp1251, utf8 и т.п.).
12. Проверить, корректно ли восстановлено значение поля после повторной загрузки страницы (кроме полей с паролями и CAPTCHA).

Что ещё можно добавить? Ваши предложения.

Текстовые поля: пароли

В дополнение к только что рассмотренному для полей с паролями стоит проверить:

1. Очищается ли значение поля после перезагрузки страницы.
2. Есть ли предупреждение о том, что использованный пароль слишком прост.
3. Есть ли предупреждение о том, что использованный пароль совпадает с логином или некоторой персональной информацией (телефон и т.п.).

Что ещё можно добавить? Ваши предложения.

Текстовые поля: многострочные поля

В дополнение к чек-листу по «текстовым полям вообще» для многострочных полей стоит проверить:

1. Сколько данных мы можем вставить в поле из буфера. Если поле «хитрое» и умеет принимать не только текст – проверим, **данные какого типа** оно принимает.
2. Применяется ли **конвертация концов строк** (т.е. `\n` → `
`). Корректно ли она работает. Нужна ли она здесь.

Что ещё можно добавить? Ваши предложения.

Чек-боксы

Для чек-боксов следует проверить:

1. Состояние по умолчанию.
2. Остаётся ли чек-брокс (не)выбранным после повторной загрузки формы.
3. Изменяется ли состояние чек-брокса при клике по ассоциированной с ним надписи.
4. «Выбрать все», «Снять отметку со всех», «Инвертировать выбор» (если присутствует).
5. Работоспособность взаимосвязи между чек-броксами (при её наличии)

Что ещё можно добавить? Ваши предложения.

I agree to receive spam: ☐

Радио-кнопки («радио-баттоны»)

Для **радио-кнопок** следует выполнить следующие проверки:

1. Состояние **по умолчанию**.
2. Сохранение состояния **после повторной загрузки формы**.
3. Изменяется ли состояние группы радио-кнопок **при клике по ассоциированной с некоторой опцией надписи**.
4. Если на странице присутствует **несколько групп радио-кнопок** — действительно ли все они независимы.
5. Можно ли **отправить форму**, не выбирая ни одну из опций, представленных радио-кнопками?

Что ещё можно добавить? Ваши предложения.

Red: ☐

Green: ☐

Blue: ☐

Кнопки

Для любых **кнопок** на форме следует проверить:

1. Понятно ли, **что это за кнопка**.
2. Состояние **по умолчанию** (доступна / недоступна).
3. **Корректность текста (картинки)** на кнопке.
4. Можно ли **кликнуть по кнопке дважды**, и какова реакция приложения на такое действие.

Что ещё можно добавить? Ваши предложения.



Поле отправки файла

Для поля отправки файлов следует проверить:

1. Есть ли указание на то, **какие файлы (формат, размер) можно отправлять**.
2. Состояние **по умолчанию** (поле доступно/недоступно).
3. Наличие **фильтра по расширению**.
4. Какова **реакция приложения на отправку**: несуществующего файла, пустого файла, файла неверного формата, файла с неверным расширением, повреждённого файла, слишком большого файла.
5. Если при отправке файла появляется **полоса прогресса**, корректно ли она работает.

Что ещё можно добавить? Ваши предложения.

Скрытые поля

Для **скрытых полей** следует проверить:

1. Значение **по умолчанию**.
2. Как **значение поля** меняется в ответ на действия пользователя с формой.
3. Не содержит ли поле **информации, просмотр которой** пользователем нежелателен.
4. Как **можно навредить приложению**, используя специально созданное вредоносное значение в этом поле.

Что ещё можно добавить? Ваши предложения.

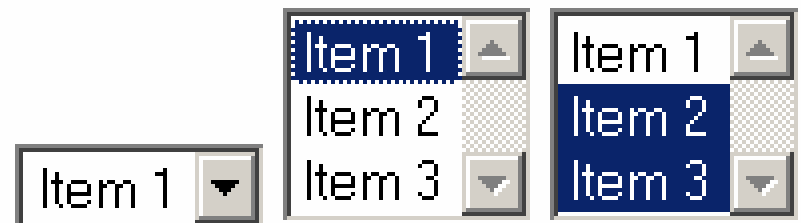
Списки

Существует **много разновидностей списков**: выпадающие, поле ввода + выпадающий список, список значений с полосой прокрутки, список с возможностью выбрать несколько значений и т.п. Поэтому без **хороших требований** тестировать списки очень сложно.

Для **списков** следует проверить:

1. Значение **по умолчанию**.
2. Сохраняет ли список своё **состояние после повторной загрузки формы**.
3. Как работает **возможность выбрать несколько вариантов**.
4. Как форма **реагирует на изменение значения списка** (если такая реакция предусмотрена).
5. Корректность **«заголовков группировки полей»** (если они должны быть).

Что ещё можно добавить? Ваши предложения.



Итак, тестирование форм – дело непростое. И оно тем более важно, что **на основе форм организуется взаимодействие с пользователями.**

Несколько советов:

1. Добейтесь наличия **ХОРОШИХ** требований. Тестировать сложную форму без хороших требований – крайне тяжело, а иногда и вовсе невозможно.
2. Помните о **юзабилити и внешнем виде** формы.
3. Проверьте работу формы **под разными браузерами.**
4. Загляните в **код формы.** Изучите его на предмет ошибок и потенциальных уязвимостей.
5. Представьте, что **формой будет пользоваться полный идиот** – что он может сделать не так.
6. Представьте, что **злоумышленник использует форму для атаки** на приложение – что он может сделать.
7. Используйте свой **опыт и здравый смысл.**

Тест для проверки изученного

1. Перечислите виды полей формы.
2. Назовите несколько тестов, которые можно применить к ЛЮБОМУ полю формы.
3. Какие проблемы может вызвать отсутствие фильтрации введённых в поле данных.
4. Если пользователь неверно заполнил несколько полей формы, как лучше сообщить об этом – за один раз или по мере исправления ошибок?
5. Через какие поля формы злоумышленнику проще всего передать вредоносные данные?
6. С заполнением каких полей формы, с вашей точки зрения, у большинства неквалифицированных пользователей возникает больше всего проблем?
7. Для каких полей не следует приводить пример заполнения?
8. Как можно избежать потери данных при заполнении очень большой (много полей) формы?
9. Назовите 2-3 теста, в случае не прохождения которых дальнейшее тестирование формы теряет смысл.
10. Как можно организовать проверку корректности введённых данных до отправки содержимого формы на сервер?