Table of Contents

Data Forms **1**
- Overview 1
- Defining a form 1
- Enabling forms by Web 2
- Adding a form to a topic 2
- Changing a form 3
- Structure of a form definition 3
  - Values in other topics 5
- Extending the range of form data types 6
- Hints and tips 6
  - Build an HTML form to create new form-based topics 6
  - Searching in form data 6
  - Gotcha! 6

Data Forms

Data forms allow you to add structured data to topics. The data stored in the form fields can be used to search and filter topics. The combination of structured data and search queries are the base for building database applications.

Overview

By adding form-based input to freeform content, you can structure topics with unlimited, easily searchable categories. A form is enabled for a web and can be added to a topic. The form data is shown in tabular format when the topic is viewed, and can be changed in edit mode using edit fields, radio buttons, check boxes and list boxes. Many different form types can be defined in a web, though a topic can only have one form attached to it at a time.

Typical steps to build an application based on Foswiki forms:

1. Define a form definition
2. Enable the form for a web
3. Build an HTML form to create new topics based on that template topic
4. Build a FormattedSearch to list topics that share the same form

For a step by step tutorial, see AnApplicationWithWikiForm.

Defining a form

A form definition specifies the fields in a form. A form definition is simply a page containing a Foswiki table, where each row of the table specifies one form field.

1. Create a new topic with your form name: **`YourForm`**, **`ExpenseReportForm`**, **`InfoCategoryForm`**, **`RecordReviewForm`**, whatever you need.
2. Create a TML table, with each column representing one element of an entry field: **`Name`**, **`Type`**, **`Size`**, **`Values`**, **`Tooltip message`**, and **`Attributes`** *(see sample below)*.
3. For each field, fill in a new line; for the type of field, select from the list.
4. Save the topic *(you can later choose to enable/disable individual forms)*.

Example:

```
| *Name* | *Type* | *Size* | *Values* | *Tooltip message* |
*Attributes* |
| TopicClassification | select | 1 | NoDisclosure,
PublicSupported, PublicFAQ | blah blah... |   |
| OperatingSystem | checkbox | 3 | OsHPUX, OsLinux, OsSolaris,
OsWin | blah blah... |   |
| OsVersion | text | 16 | | blah blah... |   |
```

Name	Type	Size	Values	Tooltip message	Attributes
TopicClassification	select	1	NoDisclosure, PublicSupported, PublicFAQ	blah blah...	
OperatingSystem	checkbox	3	OsHPUX, OsLinux, OsSolaris, OsWin	blah blah...	
OsVersion	text	16		blah blah...	

See structure of a form for full details of what types are available and what all the columns mean.

You can also retrieve possible values for `select`, `checkbox` or `radio` types from other topics:

Example:

- In the WebForm topic, define the form:

Name	Type	Size	Values	Tooltip message	Attributes
TopicClassification	select	1		blah blah...	
OperatingSystem	checkbox	3		blah blah...	
OsVersion	text	16		blah blah...	

  Leave the **`Values`** field **blank**.

- Then in the TopicClassification topic, define the possible values:

```
| *Name*             |
| NoDisclosure       |
| Public Supported   |
| Public FAQ         |
```

Name
NoDisclosure
Public Supported
Public FAQ

Field values can also be set using the result of expanding other macros. For example,

```
%SEARCH{"Office$" scope="topic" web="%USERSWEB%" nonoise="on"
type="regex" format="$web.$topic" separator=", " }%
```

When used in the value field of the form definition, this will find all topic names in the Main web which end in "Office" and use them as the legal field values.

Enabling forms by Web

Forms have to be enabled for each individual web. The **`WEBFORMS`** setting in WebPreferences is optional and defines a list of possible form definitions.

Example:

- Set WEBFORMS = BugForm, FeatureForm, Books.BookLoanForm

- With **`WEBFORMS`** enabled, an extra button is added to the edit view. If the topic doesn't have a Form, an **Add Form** button appears at the end of the topic. If a Form is present, a **Change** button appears in the top row of the Form. The buttons open a screen that enables selection of a form specified in `WEBFORMS`, or the **No form** option.
- You have to list the available form topics explicitly. You cannot use a `SEARCH` to define `WEBFORMS`.

Adding a form to a topic

- Edit the topic and follow the "Add form" button to add a Form. This is typically done to a template topic, either to the `WebTopicEditTemplate` topic in a web, or a new topic that serves as an application specific template topic. Initial Form values can be set there.

- Additionally a new topic can be given a Form using the `formtemplate` parameter in the (edit or save) URL. Initial values can then be provided in the URLs or as form values:
  - other than checkboxes: **`name`**, ex: **`?BugPriority=1`**
  - checkbox: **`namevalue=1`**, ex: **`?ColorRed=1`**.
    Boxes with a tick must be specified.
  - Example: This will add a textfield for the new topic name and a "Create"-Button to your topic. When the button is pressed, the topic editor will open with the form "MyForm" already attached to the new topic.

```
<form name="newtopic" action="%SCRIPTURLPATH{"edit"}%/%WEB%/">
   <input type="hidden" name="formtemplate" value="MyForm" />
   New topic name <input type="text" name="topic" size="40" />
   <input type="submit" class="foswikiSubmit" value="Create" />
</form>
```

- **Note:** You can create a topic in one step, without going through the edit screen. To do that, specify the `save` script instead of the `edit` script in the form action. When you specify the `save` script you must to use the "post" method. Example:

```
<form name="newtopic" action="%SCRIPTURLPATH{"save"}%/%WEB%/" method="post">
   .....
</form>
```

- The `edit` and `save` scripts understand many more parameters, see CommandAndCGIScripts#edit and CommandAndCGIScripts#save for details.

- **Tip:** For Wiki Applications you can automatically generate unique topicnames.

- **Note:** Initial values will **not** be set in the form of a new topic if you *only* use the formtemplate parameter.

Changing a form

- You can change a form definition, and Foswiki will try to make sure you don't lose any data from the topics that use that form.

- If you change the form definition, the changes will not take affect in a topic that uses that form until you edit and save it.

- If you add a new field to the form, then it will appear next time you edit a topic that uses the form.

- If you delete a field from the form, or change a field name, then the data will not be visible when you edit the topic (the changed form definition will be used). **If you save the topic, the old data will be lost** (though thanks to revision control, you can always see it in older versions of the topic)

- If two people edit the same topic containing a form at exactly the same time, and both change fields in the form, Foswiki will try to merge the changes so that no data is lost.

Structure of a form definition

A form definition specifies the fields in a form. A form definition is simply a TML table contained in a topic, where each row of the table specifies one form field.

Each **column** of the table is one element of an entry field: **`Name`**, **`Type`**, **`Size`**, **`Values`**, **`Tooltip message`**, and **`Attributes`**.

The `Name`, `Type` and `Size` columns are required. Other columns are optional. The form **must** have a header row (e.g. `| *Name* | *Type* | *Size* |`).

`Name` is the name of the form field.

The **`Type`**, **`Size`** and **`Value`** fields describe the legal values for this field, and how to display them.

- **`Type`** `checkbox` specifies one or more checkboxes. The `Size` field specifies how many checkboxes will be displayed on each line. The `Value` field should be a comma-separated list of item labels.
    - **`Type`** `checkbox+buttons` will add **Set** and **Clear** buttons to the basic `checkbox` type.
- **`Type`** `radio` is like `checkbox` except that radio buttons are mutually exclusive; only one can be selected.
- **`Type`** `label` specifies read-only label text. The `Value` field should contain the text of the label.
- **`Type`** `select` specifies a select box. The `Value` field should contain a comma-separated list of options for the box. The `Size` field can specify a fixed size for the box (e.g. `1`, or a range e.g. `3..10`. If you specify a range, then the box will never be smaller than 3 items, never larger than 10, and will be 5 high if there are only 5 options.
    - There are two modifiers that can be applied to the `select` type:
        - `select+multi` turns multiselect on for the select, to allow Shift+Click and Ctrl+Click to select (or deselect) multiple items.
        - `select+values` allows the definition of values that are different to the displayed text. For example:

          ```
          | Field 9 | select+values | 5 | One, Two=2, Three=III, Four | Various
          ```

          shows but the values or options `Two` and `Three` are `2` and `III` respectively. You can combine these modifiers e.g. `select+multi+values`
- **`Type`** `text` specifies a one-line text field. `Size` specifies the text box width in number of characters. `Value` is the initial (default) content when a new topic is created with this form definition.
- **`Type`** `textarea` specifies a multi-line text box. The `Size` field should specify columns x rows, e.g. `80x6`; default size is 40x5. As for `text`, the `Value` field specifies the initial text
- **`Type`** `date` specifies a single-line text box and a button next to it; clicking on the button will bring up a calendar from which the user can select a date. The date can also be typed into the text box. `Size` specifies the text box width in characters. As for `text`, the `Value` field specifies the initial text

`Tooltip message` is a message that will be displayed when the cursor is hovered over the field in `edit` view.

`Attributes` specifies special attributes for the field. Multiple attributes can be entered, separated by spaces.

- An attribute `H` indicates that this field should not be shown in view mode. However, the field is available for editing and storing information.
- An attribute `M` indicates that this field is mandatory. The topic cannot be saved unless a value is provided for this field. If the field is found empty during topic save, an error is raised and the user is redirected to an `oops` page. Mandatory fields are indicated by an asterisks next to the field name.

For example, a simple form just supporting entry of a name and a date would look as follows:

```
| *Name* | *Type* | *Size* |
```

```
| Name    | text    | 80      |
| Date    | date    | 30      |
```

Field Name Notes:

- Field names have to be unique.
- A very few field names are reserved. If you try to use one of these names, Foswiki will automatically append an underscore to the name when the form is used.
- You can space out the title of the field, and it will still find the topic e.g. `Aeroplane Manufacturers` is equivalent to `AeroplaneManufacturers`.
- If a `label` field has no name, it will **not** be shown when the form is **viewed**, only when it is **edited**.
- Field names can in theory include any text, but you should stick to alphanumeric characters. If you want to use a non-wikiname for a `select`, `checkbox` or `radio` field, and want to get the values from another topic, you can use **`[[...]]`** links. This notation can also be used when referencing another topic to obtain field values, but a name other than the topic name is required as the name of the field.
- Leading and trailing spaces are *not* significant.

Field Value Notes:

- The field value will be used to initialize a field when a form is created, unless specific values are given by the topic template or query parameters. The first item in the list for a select or radio type is the default item. For `label`, `text`, and `textarea` fields the value may also contain commas. `checkbox` fields cannot be initialized through the form definition.
- Leading and trailing spaces are *not* significant.
- Field values can also be generated through a FormattedSearch, which must yield a suitable table as the result.
- Macros in the initial values of a form definition get expanded when the form definition is loaded.
    - If you want to use a `|` character in the initial values field, you have to precede it with a backslash, thus: `\|`.
    - You can use `<nop>` to prevent macros from being expanded.
    - The FormatTokens can be used to prevent expansion of other characters.

General Notes:

- The topic definition is not read when a topic is viewed.
- Form definition topics can be protected in the usual manner, using AccessControl, to limit who can change the form definition and/or individual value lists. Note that view access is required to be able to edit topics that use the form definition, though view access to the form definition is *not* required to view a topic where the form has been used.

Values in other topics

As described above, you can also retrieve possible values for select, checkbox or radio types from other topics. For example, if you have a rows defined like this:

```
| *Name*                 | *Type* | *Size* |
| AeroplaneManufacturers | select |        |
```

the Foswiki will look for the topic AeroplaneManufacturers to get the possible values for the `select`.

The AeroplaneManufacturers topic must contain a table, where each row of the table describes a possible value. The table only requires one column, **`Name`**. Other columns may be present, but are ignored.

For example:

```
| *Name* |
| Routan |
| Focke-Wulf |
| De Havilland |
```

Notes:

- The **`Values`** column **must be empty** in the referring form definition.

Extending the range of form data types

You can extend the range of data types accepted by forms by using Plugins. All such extended data types are single-valued (can only have one value) with the following exceptions:

- any type name starting with `checkbox`
- any type name with `+multi` anywhere in the name

Types with names like this can both take multiple values.

Hints and tips

Build an HTML form to create new form-based topics

- New topics with a form are created by simple HTML forms asking for a topic name. For example, you can have a `SubmitExpenseReport` topic where you can create new expense reports, a `SubmitVacationRequest` topic, and so on. These can specify the required template topic with its associated form. Template topics has more.

A form definition specifies the fields in a form. A form definition is simply a page containing a Foswiki table, where each row of the table specifies one form field.

Searching in form data

The best way to search in form data is using the structured query language in the SEARCH macro. See QuerySearch for more information.

Gotcha!

- Some browsers may strip linefeeds from `text` fields when a topic is saved. If you need linefeeds in a field, make sure it is a `textarea`.

Related Topics: UserDocumentationCategory, TemplateTopics, AnApplicationWithWikiForm

- Index
- Search
- Changes
- Notifications
- RSS Feed
- Statistics
- Preferences

- **User Reference**
- BeginnersStartHere
- TextFormattingRules
- Macros
- FormattedSearch
- QuerySearch
- DocumentGraphics
- SkinBrowser
- InstalledPlugins

- **Admin Maintenance**
- Reference Manual
- AdminToolsCategory
- InterWikis
- ManagingWebs
- SiteTools
- DefaultPreferences
- WebPreferences

- **Categories**
- Admin Documentation
- Admin Tools
- Developer Doc
- User Documentation
- User Tools

- **Webs**
- Public
- System