Creating a field

The field needs to be a *select_one* or a *select_multiple* field. Each choice in the choice list will contain one row of data (more on that later). For example, if the field is called "hh_mem", and the choice list containing information about household members is called "hh_members", then the field will start like this:

type	**name**
select_one hh_members	hh_mem

Defining your parameters

The field plug-in will take two parameters: the **delimiter** and the **headers**.

The **delimiter** will be used in choice list labels to separate the attributes or characteristics. For example, let's say you have this row of data:

Jane Doe	Female	27

If you wanted to use the pipe symbol | as the delimiter, you would enter that row of data like this:

Jane Doe|Female|27

See how the pipe symbol separates each attribute? We'll get into where that goes a bit later.

The **headers** parameter is exactly what it sounds like, containing the headers that will be used. These headers must be in a comma separated (') list. For example, let's say the header row of your table should look like this:

Name	**Gender**	**Age**

To get that header row, the **header** parameter should look like this:

header='Name, Gender, Age'

All put together, if you would like the **delimiter** to be a pipe, and for the table headers to be called 'Name', 'Gender', and 'Age', you would give the field this *appearance*:

custom-table-list(delimiter='|', header='Name, Gender, Age')

Creating data rows

You will create your table rows in the *label*s of the choices in the choice list assigned to the *select_one* or *select_multiple* field.

For example, let's say we want to use the pipe symbol | as the delimiter, and we want to display the following table in a SurveyCTO form field:

Name	**Gender**	**Age**
Jane Doe	Female	27
John Doe	Male	45
Mary Doe	Female	20

Since the attributes in the first row are "Jane Doe", "Female", and "27", the *label* of the first choice in the choice list will be:

Jane Doe|Female|27

Since the attributes in the second row are "John Doe", "Male", and "45", the label of the second choice in the choice list will be:

John Doe|Male|45

Here is what the completed choice list will look like:

list_name	**value**	**label**
hh_members	1	Jane Doe|Female|27
hh_members	2	John Doe|Male|45
hh_members	3	Mary Doe|Female|20

Tips

Creating *label*s for the rows

Because you want the form to be used with multiple respondents, you will rarely define the household information directly on the *choices* sheet. Instead, you will want to dynamically create each row, usually using the concat() function. For example, if the name of each household member will be stored in a repeated field called "name1" in the repeat group called "hh_members_rep", their gender will be stored in a repeated field called "gender", and their age will be stored in a repeated field called "age", you can create a *calculate* field called "row1" with this *calculation* expression:

concat(indexed-repeat(${name}, ${hh_members_rep}, 1), '|', indexed-repeat(${gender}, ${hh_members_rep}, 1), '|', indexed-repeat(${age}, ${hh_members_rep}, 1))

See how each attribute is separated by a pipe? (Of course, if you use a different **delimiter**, you would use a different character there.) You can use a similar expression in a field called "row2", but use 2 for the index instead of 1. When each row is complete, references to those row fields can be used as the *label*s in the choice list. In the end, the completed choice list may look something like this:

list_name	value	label	filter
hh_members	1	${row1}	1
hh_members	2	${row2}	2
hh_members	3	${row3}	3

Another option is to create a field for each property, and then separate them with the delimiter within the choice label. For example, a *calculate* field called "name1" can have this *calculation*:

indexed-repeat(${name}, ${hh_members_rep}, 1)

Keep going like that for each property and each row, and the choice list will end up looking something like this:

list_name	value	label	filter
hh_members	1	${name1}\|${gender1}|${age1}	1

hh_members	2	${name2}\|${gender2}|${age2}	2
hh_members	3	${name3}\|${gender3}|${age3}	3

See how each property is separated by the pipe delimiter? This method involves creating more fields, but it can be easier to have the expressions more broken-up like this. Use whichever method works best for you!

To learn more about making these kinds of choice lists, and how to filter out blank rows, check out part 3 of our guide to choice filters, example 2. Also check out our documentation on choosing among earlier entries.

For those "name" and other fields, there are two main ways to create them:

Ask

This method is obvious. The field "name" can be a *text* type field that asks for the name of the household member, "age" can be an *integer* type field that asks for the age, and so on.

Pull from pre-loaded data

Another method is to prepare a CSV file or server dataset, and pull the data from there. For example, let's say the attached CSV looks like this:

hh_member_key	**name**	**gender**	**age**
1	Jane Doe	Female	27
2	John Doe	Male	45
3	Mary Doe	Female	20

The repeated field "name" can be a *calculate* field with this *calculation*:

pulldata('hh_data', 'name', 'hh_mem_key', index())

That way, the first instance of the field "name" will be "Jane Doe", the second will be "John Doe", and so on.

You can take it a step further, and assign each household an ID that is used in 'hh_member_key'. For example, the "Doe" household can have the household ID of 1, and the "Li" household can have a household ID of 2. Those can be combined with the member number to create a unique identifier, like this:

hh_member_key	**name**	**gender**	**age**
101	Jane Doe	Female	27
102	John Doe	Male	45
103	Mary Doe	Female	20
201	Adnan Li	Male	14
202	Bhavna Li	Female	45
203	Charles Li	Male	47

Then, a field called "hh_num" can store the household number, and that can be combined with the index of the repeat group to create each 'hh_mem_key' value, like this:

pulldata('hh_data', 'name', 'hh_mem_key', concat(${hh_num}, '0', index()))

That will work with households that have fewer than 10 members.

To learn more, check out our documentation on pre-loading data into a form, as well as this webinar.