Renderer Data Binding

Overview

The Renderer is capable of binding your data model to the rendered markup, so that any changes made by the user through the interface are reflected in the model.

The association between the data model and the rendered markup is bi-directional:

- at render time, the value of a component will be extracted from the model, and
- when the value of a bound DOM node changes, the change is applied to the model, either manually (for example, by calling `fluid.applyChange()`) or automatically (i.e. the Renderer updates the model for you, and fires a `modelChanged` event). (See the `autoBind` option below for more information.)

Using Data Binding

To take advantage of the Renderer data binding, you must:

- use a data model
- construct the component tree to reference the data model, through the `valuebinding` field

On This Page

See Also

- Renderer
- Renderer Decorators
- Renderer Component Trees
- Renderer API
- Fluid Renderer - Background

Still need help?

Join the infusion-users mailing list and ask your questions there.

Model

Your data model can take any form you like, so long as it is pure data. The data model must be passed as a parameter to the renderer through the `model` option:

```
var myOptions = {
    model: myModel
};
fluid.selfRender(myRootNode, myComponentTree, myOptions);
```

valuebinding

To use the `valuebinding` field, an EL path into the data model is provided, instead of a direct value for the `value` field.

```
var componentTree = {
    children: [
        {ID: "rsfID", optionList: {valuebinding: "valuesELPath"},
                      optionnames: {valuebinding: "namesELPath"},
                      selection: {valuebinding: "selectionELPath"}
        }
    ]
}
```

Auto-binding

The Renderer includes an additional option to cause updates to the DOM to *automatically* be reflected in the model:

Option name	Description	Default
`autoBind`	If set, any user modification of fields with `valuebindings` set will immediately be reflected in the current state of the supplied `model`	`false`

If `autoBind` is not set to `true`, the model will only be updated upon calls to `fluid.applyChange()`. For example, when using the Inline Edit control, an update event handler can simply be written as follows:

```
function commitChange(node) {
    var newValue = $(node).val();
    fluid.applyChange(node, newValue);
}
```

Fossils and the Fossil Bolus

It is not necessary to understand exactly how the data binding is accomplished to benefit from its use, but for the curious, here is a bit of an explanation.

The association between the data model and the rendered markup is managed through the use of *fossils*:

A **fossil** is a JavaScript object that contains

- an index of the `submittingname` field of each bound component,
- its old value (that is, the value at time of rendering), and
- the EL path into the model

The **fossil bolus** is the list of all fossils for the rendered markup. It is attached to the root node of the markup using `jQuery.data()`, under the name `fluid-binding-root`.