Subcomponents

On This Page

See Also

If you've done any object-oriented programming, you're probably familiar with the idea of packaging up functionality into small, reusable, modular bundles. The Infusion Framework provides many supports for designing your application as a collection of components that work together. Components can be combined with parent-child relationships, creating a *component tree*.

Consider an example: the Infusion Pager. This component allows users to break up long lists of items into separate pages. You've probably seen pagers used for search results, etc. The Infusion Pager is actually implemented as a number of smaller components working together. The Pager Subcomponents page explains the different subcomponents.

Declaring subcomponents

Using the Inversion of Control system, subcomponents are declared in a special property of the parent's defaults called `components`. The `components` object is collection of key/value pairs:

- the key is a name for the subcomponent
- the value is an object that describes the subcomponent.

The subcomponent object can contain many properties, but the main one is the `type`, which is the string name of the subcomponent function.

```
// Define one subcomponent
fluid.defaults("tutorials.subcomponent1", {
    gradeNames: ["fluid.littleComponent", "autoInit"]
});

// Define another subcomponent
fluid.defaults("tutorials.subcomponent2", {
    gradeNames: ["fluid.littleComponent", "autoInit"]
});

// Define the parent component, to use the subcomponents
fluid.defaults("tutorials.parentComponent", {
    gradeNames: ["fluid.littleComponent", "autoInit"],
    components: {
        child1: {
            type: "tutorials.subcomponent1"
        },
        child2: {
            type: "tutorials.subcomponent2"
        }
    }
});
```

Configuring Subcomponents

In addition to the `type` of subcomponent, other configuration information can be given. These include:

- the container to use (for view components)
- a list of subcomponent `options`, for overriding the defaults of the subcomponent
- properties for controlling the timing of subcomponents, such as `priority` and `createOnEvent`

Example: User Interface Options

The Infusion [(Floe) UI Options (2008-2009)](#) (UI Options) component presents a collection of controls that allow a user to specify their preferences for customizing the presentation of the user interface and content resources. It works with the User Interface Enhancer (UI Enhancer), which carries out the transformations based on the recorded preferences.

The UI Options component has a component tree as shown to the right. Essentially:

- UI Options comes in three different versions, each of which have a UI Options subcomponent:
    - Fat Panel,
    - Full, without Preview, and
    - Full page, with Preview.
- The 'fat panel' version is presented as a sliding panel at the top of a page, and so has a Sliding Panel subcomponent.
- The 'fat panel' version applies changes directly to the page as they are being adjusted, so it has a Live Preview subcomponent.
- The 'full, with preview' version has a Preview subcomponent
- The 'full page, without preview' version has no subcomponents other than UI Options.
- The UI Options component has three subcomponents for the three different types of controls presented.

You can see from this tree that the UIOptions component is being re-used by the three different versions.
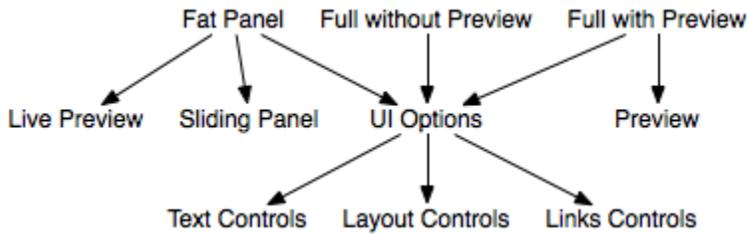
The code for this looks as shown below:

```
fluid.defaults("fluid.fatPanelUIOptions", {
    gradeNames: ["fluid.viewComponent", "autoInit"],
    components: {
        uiOptions: {
            type: "fluid.uiOptions",
            container: ".flc-slidingPanel-panel"
        },
        slidingPanel: {
            type: "fluid.slidingPanel",
            priority: "last",
            container: "{fatPanelUIOptions}.container"
        },
        preview: {
            type: "fluid.uiOptions.livePreview"
        }
    }
});

fluid.defaults("fluid.fullNoPreviewUIOptions", {
    gradeNames: ["fluid.viewComponent", "autoInit"],
    components: {
        uiOptions: {
            type: "fluid.uiOptions",
            container: "{fullNoPreviewUIOptions}.container"
        }
    }
});

fluid.defaults("fluid.fullPreviewUIOptions", {
    gradeNames: ["fluid.viewComponent", "autoInit"],
    components: {
        uiOptions: {
            type: "fluid.uiOptions",
            container: "{fullPreviewUIOptions}.container"
        },
        preview: {
            type: "fluid.uiOptions.preview",
            createOnEvent: "onReady"
        }
    }
```

```
    }
});

fluid.defaults("fluid.uiOptions", {
    gradeNames: ["fluid.viewComponent", "autoInit"],
    components: {
        textControls: {
            type: "fluid.uiOptions.textControls",
            container: "{uiOptions}.dom.textControls",
            createOnEvent: "onUIOptionsTemplateReady",
            options: {
                textSize: "{uiOptions}.options.textSize",
                lineSpacing: "{uiOptions}.options.lineSpacing",
                model: "{uiOptions}.model",
                applier: "{uiOptions}.applier"
            }
        },
        layoutControls: {
            type: "fluid.uiOptions.layoutControls",
            container: "{uiOptions}.dom.layoutControls",
            createOnEvent: "onUIOptionsTemplateReady",
            options: {
                model: "{uiOptions}.model",
                applier: "{uiOptions}.applier"
            }
        },
        linksControls: {
            type: "fluid.uiOptions.linksControls",
            container: "{uiOptions}.dom.linksControls",
            createOnEvent: "onUIOptionsTemplateReady",
            options: {
                model: "{uiOptions}.model",
                applier: "{uiOptions}.applier"
            }
        }
    }
});
```