IoC References

On This Page

Overview

The Infusion IoC Framework uses a basic syntax for referencing objects in the current context.

References always take the syntactic form `{context-name}.some.path.segments` - the meaning and form of the context name can vary and have a different meaning in different contexts:

Syntax	Description

`"{<componentRef>}.<path to member>"`	• `<componentRef>` is a reference to a component via one of its context names. It may be
	  ◦ "that" to reference the current component e.g. `"{that}"`
	  ◦ the fully qualified name of the component's type or one of its grade names e.g. `"{fluid.pagedTable}"`
	  ◦ the 'short name,' of the component's type or one of its gradeNames, i.e. the last segment of the fully namespaced name, e.g. `"{pagedTable}"`
	  ◦ the component's member name, i.e. the name used when defining a subcomponent in a components block
	• `<path to member>` is an EL path into the referenced component's members (this path may be empty)

`"{arguments}.<index>"`	• `{arguments}` refers to the array of arguments passed to a function. This form is used in the definition of Invokers • `<index>` is the 0-based numeric index of the desired argument Note that the `arguments` context name can only be used in contexts where arguments are in scope - for example, as part of the arguments to an event listener or invoker
"{source}.<path to member>"	• {source} refers to the particular instance of the `sources` array which was used to instantiate this particular dynamic component. This context name is not valid outside a dynamic component definition.
`"{<iocss expression>}.<path to member>"`	• `<iocss expression>` is an IoCSS expression referencing a component. • `<path to member>` is an EL path into the referenced component's members. **Note that full IoCSS expressions are not valid in all contexts. They can primarily be used in the `target` field of the `distributeOptions` record.**

Where To Use IoC References

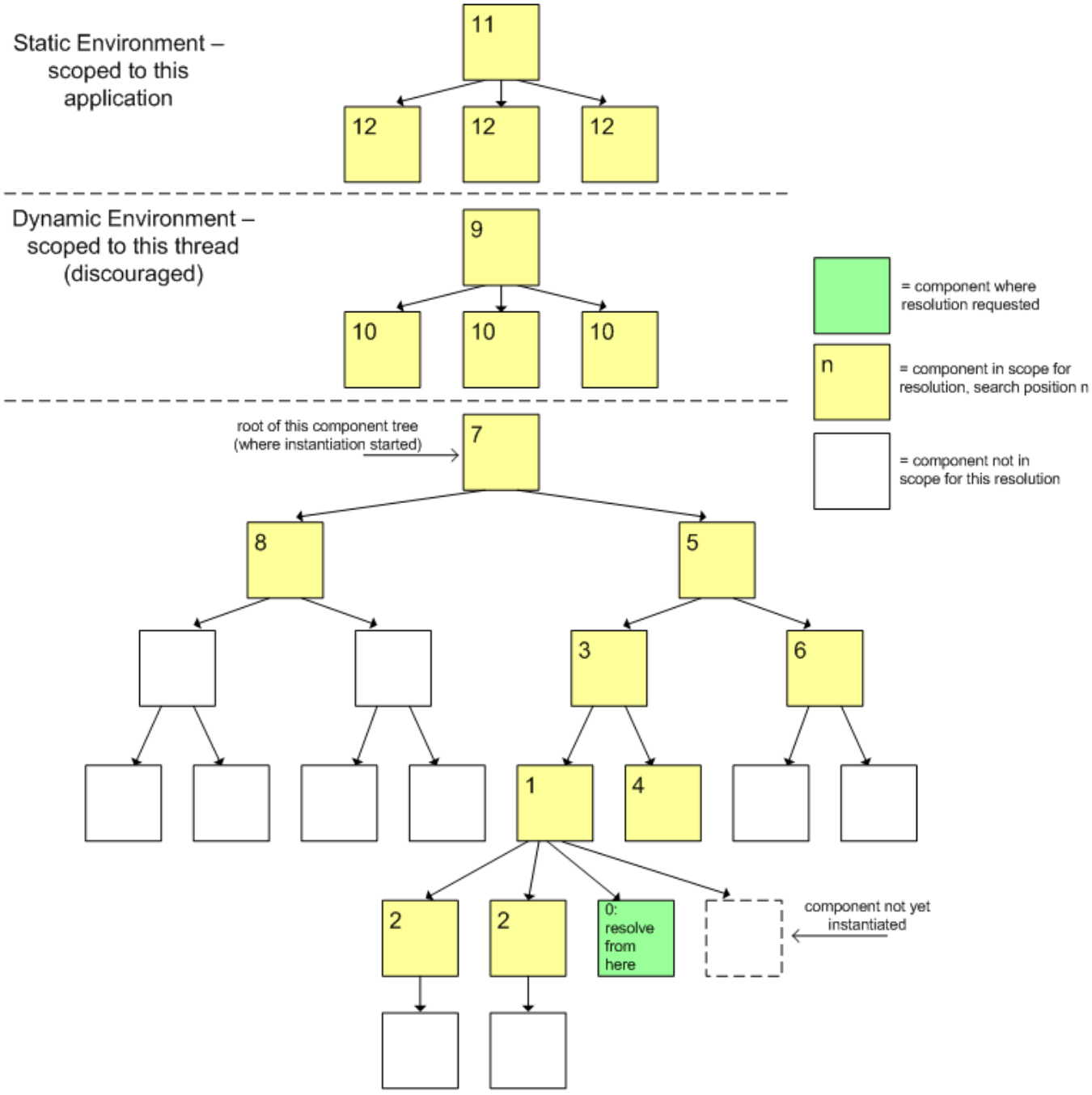
IoC references may be used almost anywhere within a component's options, for example:

- in the subcomponent definitions,
- in invoker specifications,

- in options distributions,
- in listeners specifications, including as the left hand side key specifying an event in a listeners block.
- in events specifications, including as the left hand side key specifying an event in an events block

How IoC References are resolved

For a conventional IoC reference (of the style `<componentRef>` rather than the style `<iocss expression>`), a search is begun upwards from the site of the reference in the component tree to find the first component which matches the context name. The following diagram shows a possible such reference site in green:



The set of components which are in scope for resolution from this site are shown in yellow in this diagram. These are components which are either i) an ancestor of the component holding the reference site, or ii) a sibling of such a component. The context reference matches a component if it matches via one of the 3 rules in the above table - ***either*** it agrees with a fully-qualified grade or type name of a component, ***or*** it agrees with the last path segment of such a name. If no context name matches anywhere in the tree, the reference expression resolves to `undefined`. In this case, if the path segments following the context name in the reference expression are not empty, the framework will throw an error.

Examples of "{<componentRef>}"

In the example below, the IoC reference `"{that}"` on line 6 refers to the component in which it is being used.

```
fluid.defaults("fluid.prefs.separatedPanel", {
    gradeNames: ["fluid.prefs.prefsEditorLoader", "autoInit"],
    listeners: {
        onCreate: {
            listener: "fluid.prefs.prefsEditorLoader.hideReset",
            args: ["{that}"]
        }
    }
});
```

This could equally be written using the short name of the `fluid.prefs.separatedPanel` component, as shown below:

```
fluid.defaults("fluid.prefs.separatedPanel", {
    gradeNames: ["fluid.prefs.prefsEditorLoader", "autoInit"],
    listeners: {
        onCreate: {
            listener: "fluid.prefs.prefsEditorLoader.hideReset",
            args: ["{separatedPanel}"]
        }
    }
});
```

The above two examples are equivalent.

In the example below, the IoC expression `"{fluid.prefs.enactors.tableOfContents}"` on line 6 refers to the component being defined by the defaults block. The short name `tableOfContents` cannot be used here, because it would not be unique: It would be unclear whether the nickname was referring to `fluid.prefs.enactors.tableOfContents` or `fluid.tableOfContents`.

```
fluid.defaults("fluid.prefs.enactors.tableOfContents", {
    gradeNames: ["fluid.viewComponent", "fluid.prefs.enactors", "autoInit"],
    components: {
        tableOfContents: {
            type: "fluid.tableOfContents",
            container: "{fluid.prefs.enactors.tableOfContents}.container",
            options: {...}
        }
    }
});
```

Another way to avoid the ambiguity mentioned above would be to use the member name, which is the name used when defining the subcomponent in the components block. In the example below `"{toc}"` on line 10 refers to the name used to define the subcomponent on line 4.

```
fluid.defaults("fluid.prefs.enactors.tableOfContents", {
    gradeNames: ["fluid.viewComponent", "fluid.prefs.enactors", "autoInit"],
    components: {
        toc: {
            type: "fluid.tableOfContents",
            container: "{fluid.prefs.enactors.tableOfContents}.container",
            options: {
                components: {
                    type: "fluid.tableOfContents.levels",
                    container: "{toc}.dom.tocContainer"
                }
                }
        }
    }
});
```

Examples of `"{<componentRef>}.<path to member>"`

The example below includes several IoC references. All of them are inside a subcomponent declaration and all include `"{controllers}"`, which in this case is a reference to the parent component. Specifically:

- on lines 16 and 17, the references are to the model and applier that are members of the parent component;
- on lines 19-21, the references are to events defined on the parent component, on lines 7-10;
- line 14 uses a reference to one of the selectors defined on the parent component, on line 4.

```
fluid.defaults("fluid.videoPlayer.controllers", {
    gradeNames: ["fluid.viewComponent", "autoInit"],
    selectors: {
        scrubberContainer: ".flc-videoPlayer-scrubberContainer",
    },
    events: {
        onScrub: null,
        onStartScrub: null,
        afterScrub: null,
    },
    components: {
        scrubber: {
            type: "fluid.videoPlayer.controllers.scrubber",
            container: "{controllers}.dom.scrubberContainer",
            options: {
                model: "{controllers}.model",
                applier: "{controllers}.applier",
                events: {
                    onScrub: "{controllers}.events.onScrub",
                    afterScrub: "{controllers}.events.afterScrub",
                    onStartScrub: "{controllers}.events.onStartScrub"
                }
            }
        }
    }
});
```

Examples of "`{arguments}.x`"

The example below uses the "`{arguments}.x`" syntax on line 9 to deliver the first and second arguments passed to listeners to the `onMove` event to the `fluid.moduleLayout.onMoveListener` function.

```
fluid.defaults("fluid.moduleLayoutHandler", {
    gradeNames: ["fluid.layoutHandler", "autoInit"],
    events: {
        onMove: "{reorderer}.events.onMove"
    },
    listeners: {
        onMove: {
            listener: "fluid.moduleLayout.onMoveListener",
            args: ["{arguments}.0", "{arguments}.1", "{that}.layout"]
        }
    }
});
```

Examples of "`{<iocss expression>}`"

The example below uses an IoCSS expression on line 17. The expression refers to the `images` selector defined (on line 4) in the `moreText` subcomponent (lines 10-12) that is a direct descendent of the current component.

```
 fluid.defaults("gpii.explorationTool.enactors.showMoreText", {
    gradeNames: ["fluid.viewComponent", "fluid.uiOptions.enactors", "autoInit"],
    selectors: {
        images: "img, [role~='img']"
    },
});
fluid.defaults("gpii.explorationTool.enactorSet", {
    gradeNames: ["fluid.uiEnhancer.starterEnactors", "autoInit"],
    components: {
        moreText: {
            type: "gpii.explorationTool.enactors.showMoreText"
        }
    },
    distributeOptions: {
        source: "{that}.options.moreTextSelector",
        removeSource: true,
        target: "{that > moreText}.options.selectors.images"
    }
});
```

More Examples

Example 1

```
// Range Annotator
fluid.defaults("fluid.pagedTable.rangeAnnotator", {
    gradeNames : ["fluid.eventedComponent", "autoInit"],
    listeners : {
        "{pagedTable}.events.onRenderPageLinks" : {
            funcName : "fluid.pagedTable.rangeAnnotator.onRenderPageLinks",
            args : ["{pagedTable}", "{arguments}.0", "{arguments}.1"]
        }
    }
});
// Paged Table
fluid.defaults("fluid.pagedTable", {
    gradeNames : ["fluid.pager", "fluid.table", "autoInit"],
    components : {
        rangeAnnotator : {
            type : "fluid.pagedTable.rangeAnnotator"
        }
    },
    ...
});
```

Example 1 above defines a `rangeAnnotator`, which is used as a subcomponent of a pagedTable. This definition uses several IoC references:

- on line 5, the expression `"{pagedTable}.events.onRenderPageLinks"` is used to refer to the `onRenderPageLinks` event of the `pagedTable` component
- on line 7, three IoC references are used:
    - `"{pagedTable}.events.onRenderPageLinks"` refers to the `pagedTable` component
    - `"{arguments}.0"` and `"{arguments}.1"` refer to the first and second arguments supplied when the source event is fired (`onRenderPageLinks`)

Example 2

```
 fluid.defaults("fluid.videoPlayer.languageControls.eventBinder", {
    gradeNames: ["fluid.eventedComponent", "autoInit"],
    listeners: {
        "{button}.events.onPress": "{menu}.toggleView"
    }
});
```

Example 2 above uses two IoC references on line 4:

- `"{button}.events.onPress"` refers to the `onPress` even of the `button` component
- `"{menu}.toggleView"` refers to the `toggleView` method of the `menu` component

Example 3

```
fluid.defaults("fluid.uploader", {
    gradeNames: ["fluid.viewComponent", "autoInit"],
    components: {
        uploaderImpl: {
            type: "fluid.uploaderImpl"
        }
    },
    distributeOptions: {
        source: "{that}.options",
        removeSource: true,
        exclusions: ["components.uploaderContext", "components.uploaderImpl"],
        target: "{that > uploaderImpl}.options"
    }
});
```

Example 3 above uses IoC references in the `distributeOptions` block:

- line 9 identifies the `options` block of the current `that` (i.e. `fluid.uploader`)
- line 12 identifies the `uploaderImpl` subcomponent of the current `that` (defined on line 4) (see IoCSS for more information about this notation)

Reserved IoC Names

The following names are reserved within the IoC system:

- that
- arguments
- options
- container
- source
  sourcePath
- change

As a result, you should typically avoid defining types that use these names as the final segment (e.g "todoList.source" or "todoList.panel.container"), since it will be impossible to resolve references to these components in many contexts.