COFFEESCRIPT

CoffeeScript Plugin - Reference Documentation

Authors: Jeff Brown

Version: 1.0-SNAPSHOT

Table of Contents

1. Introduction To The CoffeeScript Plugin

The CoffeeScript plugin provides support for taking advantage of CoffeeScript in a Grails application. This document describes the CoffeeScript integration options in a Grails application. For details about the CoffeeScript language and general usage see jashkenas.github.com/coffee-script/.

2. Defining CoffeeScript Functions

CoffeeScript functions may be defined in .coffee files under `src/coffee/`. A simple example:

```
// src/coffee/math.coffee
window.multiplyNumbers = (x, y) -> result = x * y;
msg = x + " * " + y + " is " + result;
return msg
```

All CoffeeScript source files should be defined under `src/coffee/` and should have the `.coffee` extension. At build time all of the `.coffee` files will be compiled to JavaScript using the `coffee` compiler which must be available at build time. The Grails application will have no runtime dependency on the compiler.

2.1 Compiler Configuration Options

The plugin compiles the `.coffee` files which are defined under `src/coffee/` to JavaScript using the `coffee` compiler. By default the plugin will use the `coffee` compiler which is on the PATH at build time. You may override this behavior by assigning a value to the `grails.coffeescript.compiler.location` system property at build time.

```
grails -Dgrails.coffeescript.compiler.location=/opt/node/bin/coffee run-app
```

Another option is to assign a value to the `grails.coffeescript.compiler.location` property in `grails-app/conf/BuildConfig.groovy`.

```
// grails-app/conf/BuildConfig.groovy
grails.coffeescript.compiler.location='/opt/node/bin/coffee'
```

The order of preference is the system property, followed by the `BuildConfig` property and finally the compiler found on the PATH.

The compiler is only needed when compiling `.coffee` files under `src/coffee/`. If the application does not define any of those and is only using inline CoffeeScript, the compiler is not needed.

3. Calling CaffeeScript Functions From A GSP

The plugin provides a custom GSP tag for pulling a compiled CoffeeScript library into a page. Note that what is being pulled in to the page is the compiled JavaScript, not the CoffeeScript.

```
<html>
<head>
    <coffee:js name="math"/>
</head>
<body>
    <form name="multiplicationForm">
        X = <g:textField name="x" value="" />,
        Y = <g:textField name="y" value="" />
        <input type="button" value="Multiply" onclick="document.all.result.innerText=multipl
    </form>
    <div id="result"></div>
</body>
</html>
```

The `js` tag in the `coffee` namespace requires that a value be assigned to the `name` attribute which should correspond to the name of a `.coffee` file which is defined under `src/coffee/`. The name should not include the `.coffee` or `.js` extension.

4. Defining Inline CoffeeScript

The plugin provides support for embedding CoffeeScript code directly in a GSP page using the `inline` GSP tag which is defined in the `coffee` namespace.

```
<html>
<body>
    <coffee:inline>
        window.addNumbers = (x, y) -> result = Number(x) + Number(y);
        msg = x + " + " + y + " is " + result;
        document.all.result.innerText = msg
    </coffee:inline>
    <form name="additionForm">
        X = <g:textField name="x" value="" />,
        Y = <g:textField name="y" value="" />
        <input type="button" value="Add" onclick="addNumbers(x.value, y.value)" />
    </form>
    <div id="result"></div>
</body>
</html>
```