CoffeeScript

CoffeeScript (http://coffeescript.org) is a language that compiles down to JavaScript.

Starting with version 5.4, Tapestry can automatically compile your CoffeeScript code into JavaScript on the fly. This is done with the optional tapestry-webresources module. It is highly recommended for anyone who wants to use CoffeeScript in their application ... just let Tapestry do the compilation at runtime (with access to Tapestry's full exception reporting capabilities).

Related Articles

- TypeScript
- JavaScript Modules
- Client-Side JavaScript
- CoffeeScript
- JavaScript FAQ
- Legacy JavaScript
- Component Cheat Sheet
- Ajax and Zones
- Assets

To use the tapestry-webresources module, just add the JAR to your project. For example, if you're using Maven:

pom.xml (partial)

```
<dependency>
    <groupId>org.apache.tapestry</groupId>
    <artifactId>tapestry-webresources</artifactId>
    <version>${tapestry-release-version}</version>
</dependency>
```

All of Tapestry's own client-side code is written in CoffeeScript and compiled, at *build* time, to JavaScript.

> You are completely free to use either JavaScript or CoffeeScript in your Tapestry application.

CoffeeScript versus JavaScript

CoffeeScript can be quite controversial, but also quite effective.

On the controversial side, it uses significant whitespace to denote blocks of code (like Python or Haskell) rather than curl braces (like Java, JavaScript, or Ruby). In fact, it draws influences from a wide rage of other languages, incorporating many functional programming features, a rudimentary class system, string interpolation, destructuring assignment, and more.

However, for all that, it is very close to JavaScript; it simply rounds out many of the rough edges of JavaScript (and adds a few of its own). As wide ranging as CoffeeScript syntax can be, there's generally a very close mapping from CoffeeScript to JavaScript.

CoffeeScript code often approaches Python's goal of being a "executable psuedo-code"; code that looks like a placeholder that will actually run. Some people find it more pleasant to read than JavaScript ... more *essence* and less *ceremony*.

You can still write bad code in CoffeeScript. You can write bad code in any language.