# Advanced Control Flow (2.1 legacy document)

## Table of contents

# Table of contents

**Warning:**
This document was copied as is from the Cocoon 2.1 documentation, but has not yet been fully reviewed or moved to its new home.

# 1. Calling Java

You can easily call Java code from your Flowscripts, for example:

```
var map = new java.util.HashMap(); map.put("foo", "bar");
```

## 1.1. Imports

Classes in packages under java are accessible directly in your scripts.

Note that classes under java.lang are not automatically imported, however:

```
var n = new java.lang.Integer(3);
```

All other java packages and classes are accessible under the property Packages:

```
var tree = new Packages.javax.swing.JTree();
```

You can get the effect of Java imports using the importPackage() and importClass() functions:

| In Java: | In JavaScript: |
|---|---|
| import foo.*; | importPackage(Packages.foo); |
| import foo.Bar; | importClass(Packages.foo.Bar); |

Example:

```
importPackage(java.util); var set = new TreeSet();
```

## 1.2. Bean Properties

If your Java classes have getters and setters you can access them as properties in JavaScript:

```
var d = new java.util.Date(); d.year = 2003; // same effect as d.setYear(2003);
```

## 1.3. Dynamic Compilation

Cocoon includes an embedded Java compiler that can dynamically compile Java source files and load and execute the resulting classes at runtime. During development you can take advantage of this capability to rapidly develop, test, and debug your applications. The Cocoon source resolver is used to locate source files.

Example:

```
// Cause com.xyz.MyClass to be compiled and loaded:
importClass(Packages.com.xyz.MyClass); var obj = new MyClass("foo", 123); // call a constructor obj.someMethod(); // call a method
```

### 1.3.1. Configuration

You control this behavior by specifying configuration properties in the cocoon.xconf file located in the WEB-INF/ directory of your application. These properties are located in the component-instance element under flow-interpreters whose name attribute has the value "javascript". The following properties may be set:

| Property: | Description: |
|---|---|
| reload-scripts | Determines whether Cocoon should attempt to detect changes to source files and reload them. This applies to both JavaScript and Java source files |
| check-time | Specifies an interval in milliseconds after which Cocoon will check for changes to source files (ignored if reload-scripts is false or unspecified) |
| classpath | A semicolon separated list of URL's that will be searched for Java source files |

Example:

<flow-interpreters default="javascript" logger="flow"> <!-- FOM (Flow Object Model) --> <component-instance class="org.apache.cocoon.components.flow.javascript.fom.FOM_JavaScriptInterpreter" name="javascript"> <load-on-startup>resource://org/apache/cocoon/components/flow/javascript/fom/fom_system.js</load-on-startup> <reload-scripts>true</reload-scripts> <check-time>4000</check-time> <classpath>file:C:/dev/myPackages;src/java</classpath> <debugger>enabled</debugger> </component-instance> </flow-interpreters>

## 1. Comments

add your comments