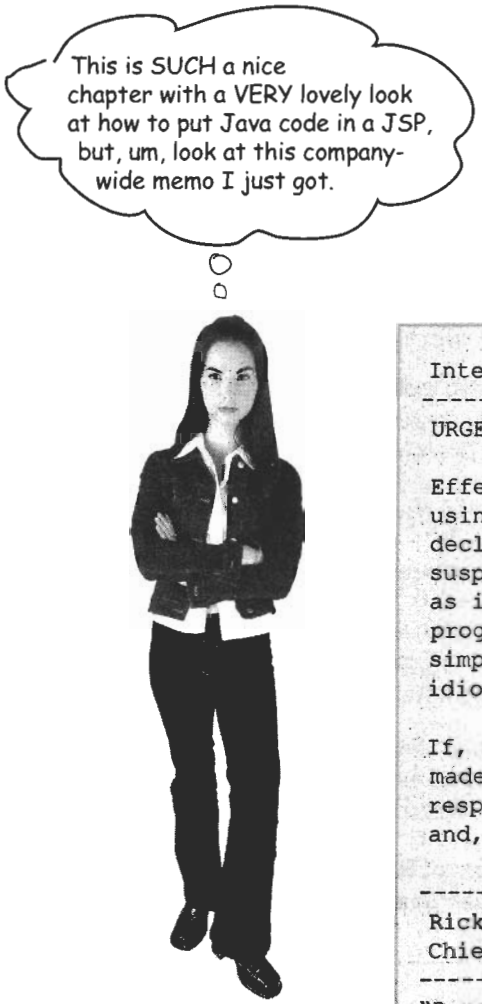


Interoffice Memo from the CTO

URGENT

Effective immediately, anyone caught using scriptlets, expressions, or declarations in their JSP code will be suspended without pay until such time as it can be determined whether the programmer was fully responsible or simply trying to maintain some OTHER idiot's code.

If, in fact, the determination is made that the programmer is, in fact, responsible, the company will go ahead and, in fact, terminate the employee.

Rick Forester
Chief Technology Officer

"Remember: there is no "I" in TEAM."

"Write your code as if the next guy* to maintain it is a homicidal maniac who knows where you live."

[*Note to HR: we use "guy" in its non-gender specific form.]

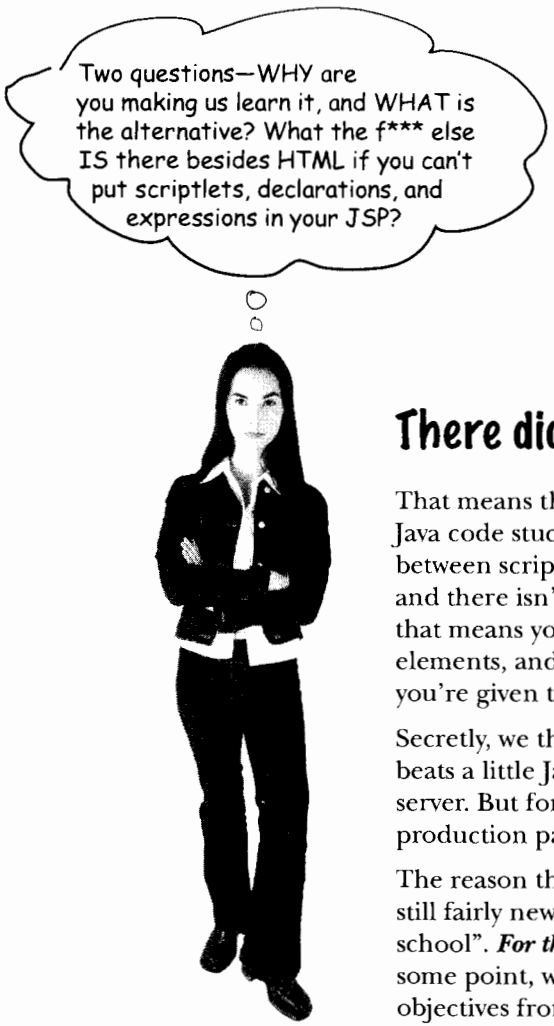
Scriptlets considered harmful?

Is it true? *Could* there be a downside to putting all this Java into your JSP? After all, isn't that the whole frickin' POINT to a JSP? So that you write your Java in what is essentially an HTML page as opposed to writing HTML in a Java class?

Some people believe (OK, technically a *lot* of people including the JSP and Servlet spec teams) that it's *bad practice* to put all this Java into your JSP.

Why? Imagine you've been hired to build a big web site. Your team includes a small handful of back-end Java programmers, and a huge group of "web designers"—graphic artists and page creators who use Dreamweaver and Photoshop to build fabulous-looking web pages. These are not *programmers* (well, except for the ones who still think HTML is "coding").

Aspiring actors working as web designers while waiting for their big showbiz break.



There didn't used to BE an alternative.

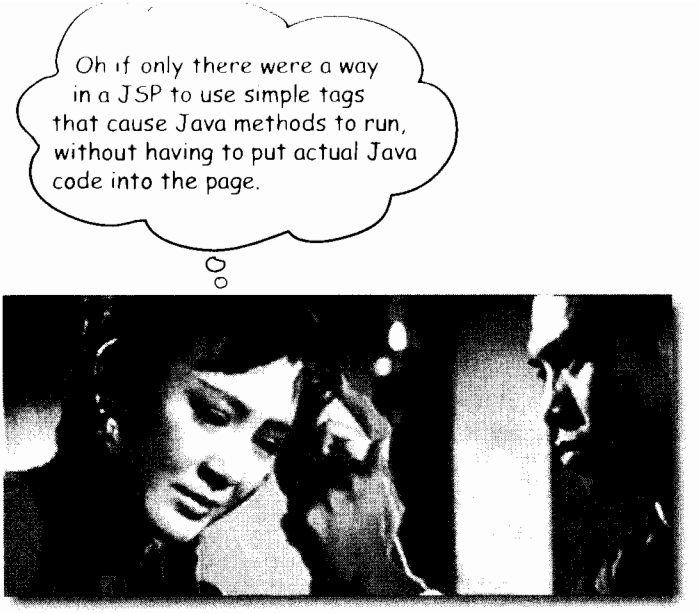
That means there are already *mountains* of JSP files brimming with Java code stuck in every conceivable spot in the page, nestled between scriptlet, expression, and declaration tags. It's out there and there isn't anything anyone can do to change the past. So that means you've got to know how to *read* and *understand* these elements, and how to *maintain* pages written with them (unless you're given the chance to massively refactor the app's JSPs).

Secretly, we think there's still a place for some of this–nothing beats a little Java in a JSP for quickly testing something out on your server. But for the most part, you don't want to use this for real, production pages.

The reason this is all on the exam is because the *alternatives* are still fairly new, so most of the pages out there today are still "old-school". ***For the time being, you still have to be able to work with it!*** At some point, when the new Java-free techniques hit critical mass, the objectives from this chapter will probably drop off the exam, and we'll all breathe a collective sigh at the death of Java-in-JSPs.

But today is not that day.

(Note to parents and teachers: the four-letter word implied in this thought bubble, that starts with "f", followed by three asterisks, is NOT what you think. It was just a word that we found too funny to include without distracting the reader, so we bleeped it out. Because it's funny. Not *bad*.)



EL: the answer to, well, *everything.*

Or *almost* everything. But certainly an answer to two big complaints about putting actual Java into a JSP:

1) Web page designers shouldn't have to know Java.

2) Java code in a JSP is hard to change and maintain.

EL stands for "Expression Language", and it became officially part of the spec beginning with JSP 2.0 spec. EL is nearly always a much simpler way to do some of the things you'd normally do with scriptlets and expressions.

Of course right now you're thinking, "But if I want my JSP to use custom methods, how can I declare and write those methods if I can't use Java?"

Ahhhh... writing the actual functionality (method code) is *not* the purpose of EL. The purpose of EL is to offer a simpler way to *invoke* Java code—but the code itself belongs *somewhere else.* That means in a regular old Java class that's either a JavaBean, a class with static methods, or something called a Tag Handler. In other words, you don't write method code into your JSP when you're following today's Best Practices. You write the Java method *somewhere else*, and *call* it using EL.

JSP Element Magnets

Match the JSP element with its label by placing the JSP snippet in the box with the label representing that element type. Remember, you'll have Drag and Drop questions on the real exam similar to this exercise, so don't skip it!

JSP element type

- directive
- declaration
- EL expression
- scriptlet
- expression
- action

JSP snippet

Drag these over and drop them onto the matching label.

```
<% Float one = new Float(42.5); %>
```

```
<%! int y = 3; %>
```

```
<%@ page import="java.util.*" %>
```

```
<jsp:include file="foo.html" />
```

```
<%= pageContext.getAttribute("foo") %>
```

```
email: ${applicationScope.mail}
```

JSP Element Magnets: the Sequel

You know what they're called, but do you remember ***where they go in the generated servlet***? Of course you do. But this is just a little reinforcement/practice before we move on to a different chapter and topic.

(Put the element in the box corresponding to where that element's generated code will go in the servlet class file. Note that the magnet itself does not represent the ACTUAL code that will be generated.

```
[redacted]

public final class BasicCounter_jsp extends org.apache.jasper.runtime.HttpJspBase
                    implements org.apache.jasper.runtime.JspSourceDependent {

  [redacted]

  public void _jspService(HttpServletRequest request, HttpServletResponse response)
                                  throws java.io.IOException, ServletException {

    ...

    [redacted]

    [redacted]

    [redacted]

    ...
  }
}
```

The order of these three magnets does not matter.

<%= request.getAttribute("foo") %>

email: ${applicationScope.mail}

<%@ page import="java.util.*" %>

<% Float one = new Float(42.5); %>

<%! int y = 3; %>

Head First Servlets & JSP™

by Bryan Basham, Kathy Sierra, and Bert Bates

Copyright © 2004 O'Reilly Media, Inc. All rights reserved.

Printed in the United States of America.

Published by O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472.

O'Reilly Media books may be purchased for educational, business, or sales promotional use. Online editions are also available for most titles (safari.oreilly.com). For more information, contact our corporate/institutional sales department: (800) 998-9938 or corporate@oreilly.com.

Editor: Mike Loukides

Cover Designer: Edie Freedman

Interior Decorators: Kathy Sierra and Bert Bates

Anthropomorphizer: Kathy Sierra

Servlet Wrangler: Bryan Basham

Assistant to the Front Controller: Bert Bates

Printing History:

August 2004: First Edition.

The O'Reilly logo is a registered trademark of O'Reilly Media, Inc. Java and all Java-based trademarks and logos are trademarks or registered trademarks of Sun Microsystems, Inc., in the United States and other countries. O'Reilly Media, Inc. is independent of Sun Microsystems.

Many of the designations used by manufacturers and sellers to distinguish their products are claimed as trademarks.

Where those designations appear in this book, and O'Reilly Media, Inc. was aware of a trademark claim, the designations have been printed in caps or initial caps.

While every precaution has been taken in the preparation of this book, the publisher and the authors assume no responsibility for errors or omissions, or for damages resulting from the use of the information contained herein.

In other words, if you use anything in *Head First Servlets & JSP™* to, say, run a nuclear power plant or air traffic control system, you're on your own.

The authors hope you remember them, should you create a huge, successful dot com as a result of reading this book. We'll take stock options, beer, or dark chocolate.

ISBN: 0-596-00540-7

[M]