14.1 Why Use Beans?

OK, so you already understand the benefit of using separate Java classes instead of embedding large amounts of code directly in JSP pages. As discussed in Section 11.3 (Limiting the Amount of Java Code in JSP Pages), separate classes are easier to write, compile, test, debug, and reuse. But what do beans provide that other classes do not? After all, beans are merely regular Java classes that follow some simple conventions defined by the JavaBeans specification; beans extend no particular class, are in no particular package, and use no particular interface.

Although it is true that beans are merely Java classes that are written in a standard format, there are several advantages to their use. With beans in general, visual manipulation tools and other programs can automatically discover information about classes that follow this format and can create and manipulate the classes without the user having to explicitly write any code. In JSP in particular, use of JavaBeans components provides three advantages over scriptlets and JSP expressions that refer to normal Java classes.

1. **No Java syntax.** By using beans, page authors can manipulate Java objects using only XML-compatible syntax: no parentheses, semicolons, or curly braces. This promotes a stronger separation between the content and the presentation and is especially useful in large development teams that have separate Web and Java developers.

2. **Simpler object sharing.** When you use the JSP bean constructs, you can much more easily share objects among multiple pages or between requests than if you use the equivalent explicit Java code.

3. **Convenient correspondence between request parameters and object properties.** The JSP bean constructs greatly simplify the process of reading request parameters, converting from strings, and putting the results inside objects.