SYBEX Sample Chapter

Mining eBay Web Services: Building Applications with the eBay API

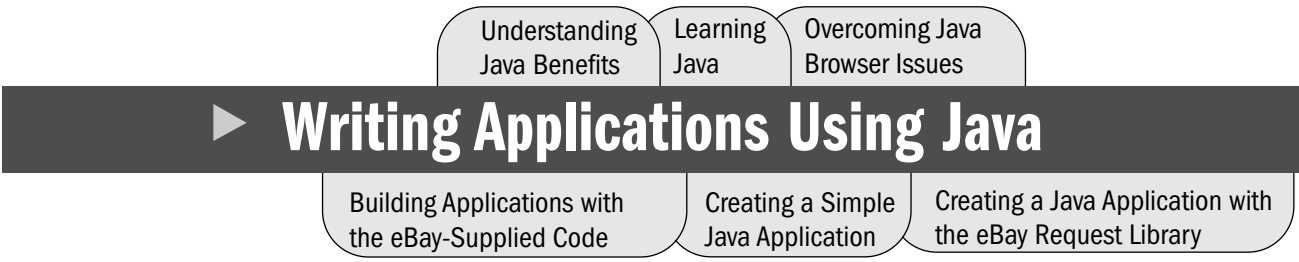
John Paul Mueller

Chapter 10: Writing Applications Using Java

ISBN: 0-7821-4339-3

Sybex Inc.
1151 Marina Village Parkway
Alameda, CA 94501
U.S.A.
Phone: 510-523-8233
www.sybex.com

Chapter 10



Most people have heard about Java and many people have worked with it. Java appears on Web sites with some regularity because it lets Web designers create solutions that work with a number of browsers. Developers create Java applets for many Web-enabled applications as well as desktop applications. In short, Java appears in numerous places, so it's no wonder that you can use Java with eBay Web Services too.

This chapter discusses techniques for using Java with eBay Web Services. The examples show various strategies you can use to improve the user experience, while keeping the cost of development low. In general, you'll find many resources for using Java online. In addition, because Sun essentially owns Java, you'll find that it enjoys a level of support that most solutions can only dream about and openness not normally found with fully proprietary solutions.

However, using Java can become problematic in some cases. Java isn't fully proprietary, nor is it fully open. Consequently, some contention surrounds Java, and you need to consider the issues using Java can cause. (Read about these issues at `http://www.infoworld.com/article/03/05/12/HNfowler_1.html`.) This chapter doesn't examine these issues in detail, but it does provide enough information that you can learn more about the issues yourself and make a decision about the suitability of Java for your next application. As part of the Java overview, you'll also learn where you can find more information about the language. As with all other languages in the book, I assume you've already learned Java and performed the required software installation before you begin this chapter.

▶ NOTE

The examples in this chapter rely on the Java 2 Platform, Standard Edition (J2SE) version 1.4.2 available at `http://java.sun.com/j2se/1.4.2/download.html`. Older versions of the product might work, but you may need to modify the example code to exclude new features or functionality. In addition, I used the Windows platform for writing many of the applications and associated explanations. While the source code will work on any platform that supports the latest version of Java, you might need to modify some of the usage instructions slightly for other platforms.

Understanding the Benefits of Using Java

Java is popular because it can do so many things well. You can use Java on either the client or the server, or even both at the same time. This feature makes Java different from other solutions such as PHP because it provides flexibility in determining where an application runs. Unlike many other solutions, Java enjoys wide platform support, so a solution you create for one platform has a good chance of working on other platforms, too. In fact, you'll find numerous Java applications available for download that run on multiple platforms. For example, a single byte-code file (compiled code) can run on Macintosh OS X, OS/2, Unix, VMS, and Windows.

Many developers use Java for both desktop and Web-based applications, although most people equate Java with Web development where it's a much stronger presence. The fact that you can use it for multiple application types makes Java a good solution for many multi-environment scenarios. Even though you'd need to make changes to an application (desktop) to run it as an applet (Web), the changes are minimal compared to other languages. However, make sure you understand the limits of Java compared to platform-specific solutions such as Visual Basic before you decide to use it in more than one place.

Using Java for Web applications has many significant benefits. The most important benefit is that so many platforms support Java natively. You can develop Web-based applications using products such as Shockwave or Flash, but this solution often forces the Web site visitor to download a browser plug-in. Given Microsoft's loss of a lawsuit allowing plug-ins, using Macromedia Shockwave or Macromedia Flash might not even be an option in the future (see the eWeek article at `http://www.eweek.com/article2/0,4149,1269693,00.asp` for details). Java applications generally work without any additional effort at all on the part of the user. In addition, Java is more capable than most other languages used for Web application presentation because it allows full interactivity between the client and server.

One feature that could be a benefit or a problem, depending on how you view it, is the fact that Java applications rely on a runtime engine that keeps them in a secure environment known as a sandbox. A Java application can only use the resources allotted to it by the runtime engine or Java Virtual Machine (JVM). This feature is beneficial because it improves security and makes it less likely that an errant Java application will cause other applications to fail (crash). The feature can cause problems by making it difficult to access resources the application needs to perform essential tasks such as writing data to the hard drive.

Another feature that you can view as a benefit or problem is the fact that Java tends to take a one size fits all approach to platform support. Yes, every platform requires a special JVM, but that JVM tends to have the same functionality as every other JVM. This means you face fewer problems porting Java applications from one platform to the next. In fact, except for text-based products such as PHP, Java is one of the easiest languages to port. However, the one size fits all approach also means that you'll experience problems using advanced features a specific platform has to offer. Microsoft tried to address the lack of platform-specific support in Java by (among other things) creating its own version of the JVM (see the "Understanding Java Browser Issues" section for details).

Resources for Learning Java

Learning any high-end language is difficult. However, some developers have complained that Java is one of the harder languages to learn because it lacks tools that other high-end languages provide. Sun is apparently aware of the complaints because it's promised to provide better tools (see the article at `http://www.infoworld.com/article/03/06/09/23NNjavaone_1.html` for details). Ease of use issues aside, Java is still a very powerful and flexible language, so you should honestly consider this solution for your next eBay Web Services application. A single chapter can't show you everything about Java, so this section provides resources where you can learn more.

▶ TIP

The court of public opinion on whether .NET or Java is the best solution to use for Web services is split. According to a survey late in 2002 (see `http://www.infoworld.com/article/02/10/09/021009hndevsurvey_1.html?1010thap` for details), developers are spending equal time on both technologies. In short, both technologies are popular—you need to decide which one meets your needs best.

Choosing a Java Editor

The Sun tutorial suggests using Windows Notepad or a similar text editor, such as Notepad+ (`http://www.mypeecee.org/rogsoft/`), for creating your Java applet code. However, you should consider using a good Java editor to make the development experience a lot better. The Sun ONE Studio 4, Community Edition IDE mentioned in the tutorial isn't available for download any longer—Sun has replaced it with a 60-day demonstration version of Sun ONE Studio 5, Standard Edition. You can get the Sun ONE Studio 5, Standard Edition IDE at `http://wwws.sun.com/software/sundev/`. The advantage of using the official IDE is that Sun designed it for Java and the IDE provides Java-specific help.

In some cases, a third party product such as SlickEdit (`http://www.slickedit.com/`) is actually a better deal. The SlickEdit solution provides support for multiple languages, which means you only have to learn one editor. Although it is a shrink-wrapped application, you can obtain a limited use trial version from the company Web site.

You might also consider using a product such as jEdit (`http://www.jedit.org/`) because the same executable runs on Macintosh OS X, OS/2, Unix, VMS, and Windows. The author wrote this editor in Java and it points out the platform independence this language provides in a real world application. You'll find that jEdit has great community support, so you can download any of a number of add-on products for it. I used the jEdit editor to write the eBaySearch and SimpleApplication examples in this chapter and found the color coding it provides extremely helpful. You can download this open source product free.

Another great IDE is JCreator (`http://www.jcreator.com/`). You can get the freeware version of the product and use it as long as you like. The professional version of the product is shareware, so you can download and use it free for 30 days. I also tried this editor while working on the ERLSample and SimpleSearch examples for this chapter. It's a great choice for developers who have worked with VBA or Visual Studio and are familiar with the IDE for those products. It is also one of the better editors for large projects because it helps you organize your project better and includes features such as a debugger.

One of the best places to learn about Java is the Sun Web site at `http://java.sun.com/docs/books/tutorial/`. This site provides a good overview of Java and some great introductory material you can use to learn the language. For example, this tutorial explains the difference between a stand-alone application and an applet. Note that this Web site does provide separate instructions for Windows, Linux/Unix, and Macintosh developers. You might find it helpful to read the instructions for your platform of interest, and then quickly glance through the other two sections to pick up platform-specific issues.

If you need another basic tutorial, then look at Brewing Java: A Tutorial at `http://www.ibiblio.org/javafaq/javatutorial.html`. The author of this tutorial actually expanded it into a book that he also mentions on the site. If you find the whole concept of object-oriented

programming with Java difficult to understand, you'll want to view the Don't Fear the OOP tutorial at `http://sepwww.stanford.edu/sep/josman/oop/oop1.htm`. You can get interactive Java coding lessons at the Coder School (`http://www.coderschool.com/`). Many of the offerings on this site are free, but you'll also find some excellent paid offerings as well.

An excellent beginner tutorial is Phil Heller's *Ground-Up Java* (Sybex, 2004). *Ground-Up Java* assumes no programming experience, but gets the reader up and running as a Java programmer quickly. The unique aspect of this book is the collection of powerful animated illustrations on the accompanying CD-ROM. They provide a crash-free environment to experiment with Java programming. The animated illustrations combined with the graded exercises that conclude every chapter and Phil's clear explanations of concepts and techniques make *Ground-Up Java* a programming course and computer lab rolled into one.

Once you learn a little about Java, try some of the more advanced developer tutorials offered by Sun at `http://developer.java.sun.com/developer/onlineTraining/`. The tutorials on this site are diverse and some are complex, so make sure you understand the requirements for using the tutorial before you get too involved (the requirements normally appear as part of the tutorial's introduction). The feature I like most about this site is that all of the tutorials have dates, so you know how old the information is before you get started.

A number of third parties also provide advanced tutorials and this section doesn't even begin to list them all. One of the more interesting offerings is the Advanced Java/J2EE Tutorial at `http://my.execpc.com/~gopalan/java/java_tutorial.html`. This tutorial begins with a comparison of the various communication technologies (including Java/RMI, DCOM, and CORBA). You might also want to look at Java Coffee Break at `http://www.javacoffeebreak.com/` because it includes a wide range of tutorials (some advanced) as well as other resources.

Newsgroups can also provide essential information to the Java developer. One of the best newsgroups to try is `comp.lang.java`. Note that this newsgroup has numerous subfolders you'll also want to visit. For example, you can keep track of Java bugs on the `comp.lang.java.bugs` newsgroup. The `comp.lang.java` newsgroup enjoys broad support and some people even support it on their Web sites. For example, check out the comp.lang.java FAQ List at `http://www.ibiblio.org/javafaq/javafaq.html`. You can also try newsgroups such as `alt.comp.lang.java`. Make sure you check any vendor-specific Java newsgroups groups such as `borland.public.jbuilder.java` when you use a particular product.

Understanding Java Browser Issues

As previously mentioned, you can use Java in a client, server, or mixed solution. The problem with developing either a client or mixed solution is that you have to consider the user's browser. The media has documented the combat between Sun and Microsoft over the JVM. (See the

story at http://archive.infoworld.com/articles/hn/xml/02/12/05/021205hnmsblames.xml?1205tham as just one example.) Microsoft, as usual, decided to produce its own version of the JVM, which is incompatible with Sun's version. Some users might have this incompatible version installed, even though Sun won a lawsuit over the issue and Microsoft no longer produces it.

The latest twist in the battle is that some versions of Windows no longer come with the JVM installed, which means that the client can't run your Java application at all. In some cases, Microsoft is withdrawing products from the market earlier than anticipated to ensure they meet the court-ordered deadline for restricting distribution of their custom JVM solution (see the eWeek article at http://eletters.eweek.com/zd1/cts?d=79-353-2-3-67152-42164-1 for details). Microsoft originally shipped Windows XP without a JVM, downloading the JVM on demand, rather than supplying it as a default (see the InfoWorld story at http://archive.infoworld.com/articles/hn/xml/02/06/18/020618hnjavasupport.xml?0620thap for details).

Sun is also active in the JVM battle. For example, the company is trying to get around the Windows JVM problem by signing individual companies to distribute the Sun version of the JVM (see the InfoWorld article at http://www.infoworld.com/article/03/06/11/HNjavadell_1.html for details). The two companies are still in court over this issue (see the story at http://www.infoworld.com/article/03/04/03/HNmsorder_1.html). Because of this contention, you can't be sure which version of the JVM a client has or even if the client has the JVM installed.

Even if the client has the JVM installed, crackers have made Java one of their tools of choice. Consequently, many people turn off support for the JVM in their browsers. This task is amazingly easy with products such as Internet Explorer. Because you don't know whether the client has the JVM installed, telling them to turn the JVM on in an error message is unlikely to produce the desired results in many cases.

Finally, it might seem like everyone would have a JVM installed and all the proper browser support, but that's not true. Many browsers simply don't have the required support. You can view the Webmonkey charts at the following locations for specific platform support of Java.

- Windows
  http://hotwired.lycos.com/webmonkey/reference/browser_chart/index.html
- Macintosh
  http://hotwired.lycos.com/webmonkey/reference/browser_chart/index_mac.html
- Linux
  http://hotwired.lycos.com/webmonkey/reference/browser_chart/index_nix.html
- Other
  http://hotwired.lycos.com/webmonkey/reference/browser_chart/index_other.html

The bottom line is that you have to know the client capabilities of the users of your application or develop a server-side solution. Java is a great solution because it's so flexible, but it also carries a number of problems that you might not run into with less flexible or less capable solutions. You need to decide whether the potential browser problems with Java are going to interfere with your eBay Web Services application.

Viewing the eBay-Supplied Code

eBay provides sample code that shows how to create an item listing. This example requires some special setup because it relies on a special library. In general, the example won't run until you perform the required setup. The following sections describe the setup, use, and functionality of the eBay-supplied example. You'll find this example in the `\Program Files\eBay\SDK\Samples\Java\SimpleListJava` folder of the eBay Web Services Kit.

▶ WARNING

After working with this example for a considerable time, it's apparent that this particular example isn't very stable, nor is the SDK access technique it uses. It is the only example in the eBay Web Services Kit that didn't run out of the box on any of my test systems. The number of messages on the eBay forums indicates that I'm not the only one with this problem. Unless you can find another COM bridge to replace the `Jacob.DLL` file described in the sections that follow, you'll probably get better results using the API technique. All of the other examples in this chapter use the API technique.

Performing the Required Setup

Java requires special support for accessing eBay Web Services. The example assumes you're using a Windows machine. You'll need to copy `Jacob.DLL` from the `\Program Files\eBay\SDK\Java` folder to the `Windows \System32` folder to ensure Java can find it when needed. As an alternative, you can add the `\Program Files\eBay\SDK\Java` folder to your system path.

JACOB is short for Java COM Bridge—the function that the DLL serves. Make sure that you don't have any other version of `Jacob.DLL` available on the test system. Older versions can cause the test application to fail. Fortunately, just installing the Java SDK won't install `Jacob.DLL` for you, so you shouldn't need to do anything special if you create a clean installation or haven't specifically installed `Jacob.DLL` for another purpose. You can obtain the current `Jacob.DLL` documentation, including the source code, at `http://danadler.com/jacob/`.

▶ WARNING

The `Jacob.DLL` doesn't run well with some versions of Windows unless you have specific features installed. Make sure you install the .NET Framework version 1.0, not version 1.1 for the best results on both Windows 2000 and Windows XP. (You can download the .NET Framework 1.0 at `http://www.microsoft.com/downloads/details.aspx?FamilyId=D7158DEE-A83F-4E21-B05A-009D06457787`.) eBay provides this information as part of the `ReadMe` file in the example folder. Tests showed that a machine with the .NET Framework 1.1 installed did indeed experience problems. A second machine with a side-by-side installation didn't experience as many problems. Any test machine you set up should have all the current service packs and patches installed. eBay recommends that Windows XP developers have Service Pack 1 installed as a minimum (there's a SP1a available and Microsoft plans to release SP2 soon).

Running the Example

Most developers run example programs to see what's possible and then dissect the code to make their own development experience better than it could be learning from scratch. One of the most important benefits of using the eBay-supplied Java example is to ensure you have a viable setup before you begin developing an application. It doesn't pay to progress any further in the chapter unless you can get the sample application to work. If this example doesn't work, it means that `Jacob.DLL` is probably incompatible with your system and you need to use the API technique to write your code.

To use the example, open a command window in the `\Program Files\eBay\SDK\Samples\Java\SimpleListJava` folder of the eBay Web Services Kit. Type **`Make`** at the command prompt and press Enter. Java will build the example. If Java doesn't build the example, make sure you have paths set up so the Java compiler can locate all of the files it needs. Run the example by typing **`SimpleListJava`** and pressing Enter.

▶ NOTE

The example times out after a relatively short time. Consequently, you'll want to enter the information relatively quickly to ensure you get to see a result.

The example is a simple console application. It begins by requesting your developer keys and the user information. As with most of the eBay samples, this sample lacks any error trapping, so you need to enter the information carefully. Unlike most eBay examples, this one

doesn't assume that you want to use the sandbox to make your request, so you have to supply the URL of the environment you want to use. The example does suggest the sandbox URL of `https://api.sandbox.ebay.com/ws/api.dll`.

▶ TIP

As part of the Java example input, you can provide the name and location of a log file. It's tempting to simply press Enter as suggested, but the log file can provide some interesting information as you create your own application. Generally, it's a good idea to create the log file so you can follow the sequence of communication between the example application and eBay.

Once you enter the preliminary information, the example lets you make a series of calls to eBay, so this example isn't simply a matter of listing a new product online. If you're creating a communication log, it pays to make every call offered so you can see how they work with the example and Java. The first prompt asks whether you want to use the `GetCategoriesCall()`. Type **y** and press Enter. You should see a list of categories. This is the point where the application normally fails if it's going to fail. If you see an error that describes an access error at the current Java thread for `com.jacob.com`, you know the SDK setup won't work (at least, not well). Errors that don't mention `com.jacob.com` could have other causes such as an inaccurate entry.

After the example shows a list of categories, it asks you to enter information for listing a new item on eBay. This information includes the title, description, location, category, price, and BuyItNow price of the item. The example assumes certain pieces of information for you, such as setting the title bold and setting the region information. The application builds the new item, at this point, and sends it to eBay. You should see a success message that includes the item identifier.

Examining the Code

The example code is relatively straightforward. It begins by creating a new session by calling `IApiSession apiSession = new ApiSession()`. Notice that you must use the interface and not the actual object when creating the session variable. The code fills in the required `apiSession` variables using inputs from the command line. It also fills in one optional entry, the log file, when you provide a filename. Because the code creates an XML log, it must define a special object to hold the data. Here's the general procedure that you should follow when creating a log.

```
// Create log file object.
ILogFile log = new LogFile();
log.open(input);
```

```
// Assign the log to the session.
apiSession.setLogCallXml(true);
apiSession.setLog(log);
```

After the example completes creating the session, it creates the first session call object, `apiGetCats`, using the `IGetCategoriesCall` interface if you choose to get the category list. It appears that Java can't assign the session to the `GetCategoriesCall` object directly, so the code assigns this information indirectly. Normally, this would mean assigning the session to the `apiGetCats.APICall.ApiCallSession` property, but the example uses a two-step process to perform the task:

```
parentApi = apiGetCats.getAPICall();
parentApi.setApiCallSession(apiSession);
```

The first step retrieves the `APICall` property. The second step associates the session to the `APICall.ApiCallSession` property. The main reason for this convoluted approach is that `Jacob.DLL` doesn't appear to support multiple levels of indirection. Notice the use of get and set methods for the property values. The code completes the categories call by displaying the categories as a single list using a simple loop.

The code begins building an `IItem` object next. This object contains the item information for the new listing send to eBay by the `AddItemCall()` method. Creating the item is a matter of filling out the properties based on inputs provided by the user. After the code creates the `IItem` object, it creates the `IAddItemCall` object to make the call. This call uses the same convoluted approach to assigning the session to the object. When the call is successful, it displays the `item.getItemId()` value on screen.

At one point in the listing, you're going to run into a commented out call to `BuildAttributes()`. It isn't clear what this extra code is supposed to do in the example. However, the function does exist and you might want to work with it to build items, such as cars, that do rely on attributes.

Developing a Simple Java Application

This section shows how to create a simple Java application using the API technique to access eBay. As previously mentioned, the current SDK technique setup for Java is unstable. The API technique appears to work flawlessly. Despite the need to write more code when using the API technique, the stability is worth the additional effort.

Rather than use the command line approach of the eBay-supplied example, this example relies on a third party editor, JCreator. The code will work with any editor you choose to use, including a simple text editor. However, you'll find debugging and running the application goes a lot faster if you have a good editor.

Configuring the JCreator Editor

The professional version of the JCreator editor can make working with eBay Web Services a lot easier because it provides help with the various packages and libraries. You don't have to spend time working with batch files because the application compiles within the IDE. In addition, the debugger makes it easy to see how requests to and responses from eBay Web Services work.

Because there are a number of ways to work with eBay Web Services, I decided to create a setup that could work with them all. This section shows how to install support for the SDK technique, the eBay Request Library technique (depends on the API technique), and the future Simple Object Access Protocol (SOAP) technique. You only have to perform steps 1 through 3 if you want to use the API technique without any fancy add-ons.

When using the SOAP technique, you'll need the Axis 1.1 or above library you can download from `http://ws.apache.org/axis/` and a current copy of Xerces-J you can download from `http://xml.apache.org/dist/xerces-j/`. You'll likely need a copy of the eBay support for SOAP, but the production SOAP environment information isn't available as of this writing. When using the eBay Request Library, you actually need two libraries, the one supplied by eBay and the JDOM library. Download a copy of the eBay Request Library from `http://developer.ebay.com/DevZone/docs/Samplecode.asp` and the JDOM library from `http://www.jdom.org/downloads/index.html` (I used Beta 9 of this particular library for the examples).

▶ NOTE

The JCreator Web site currently includes two versions of the product: 2.5 and 3.0. In addition, you can download a freeware or professional product of each of these major versions. The examples in this chapter assume you have JCreator 3.0 Professional. The steps will also work with the freeware product, but won't work with either 2.5 product.

1. Create a new workspace using the File ➢ New ➢ Project command. You'll see a Project Wizard dialog box shown in Figure 10.1.
2. Select the Basic Java Application option and click Next. The wizard will ask you to provide a project path.
3. Type a name for the project (the example uses SimpleSearch) in the Name field and click Next. (Click Finish if you intend to use the straight API technique.) You'll see the Project ClassPath dialog box shown in Figure 10.2. Notice that this dialog box has two tabs JDK Profiles and Required Libraries.

FIGURE 10.1:

Select a project type from the list of options.

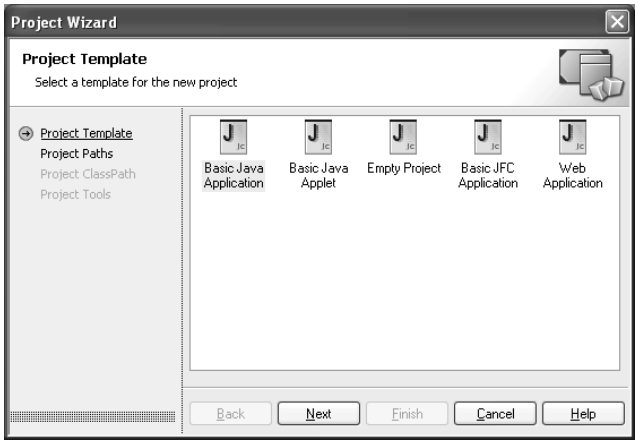
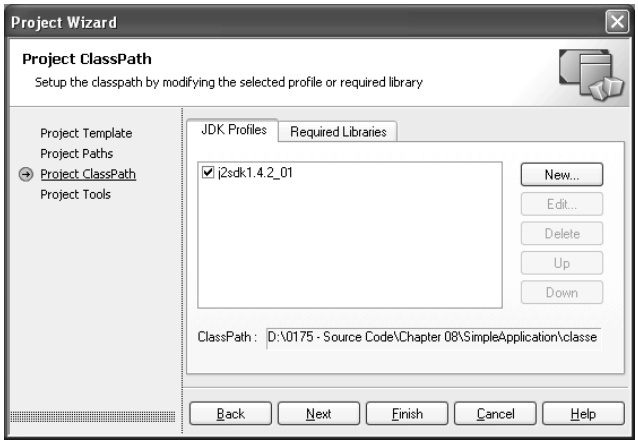


FIGURE 10.2:

Define one or more libraries as needed using the Project Wizard dialog box.



4. Select the Required Libraries tab so you can add the Axis, Xerces, JDOM, and custom eBay libraries.
5. Click New. You'll see the Set Library dialog box shown in Figure 10.3. This dialog box can add either archives, such as those used for Axis, Xerces, and JDOM, or library paths, such as the one used for the custom eBay Request Library.
6. Select Add ➢ Add Path or Add ➢ Add Archive as appropriate. In both cases, you'll see a dialog where you can select the required path or archive file. When selecting the custom eBay SDK library path, choose the `\Program Files\eBay\SDK\Java` folder because it's the top-level folder for the library. After you download the eBay Request Library, add its path

as a library. The default path is `\eBayRequestLibrary\src`. The default JDOM paths include `\jdom-b9\build` and `\jdom-b9\lib`. When working with Axis and Xerces, select all of the appropriate archive files in the `\axis-1_1\lib or \xerces-2_5_0` folder.

▶ NOTE

You can also add source code and documentation files when available using the Sources and Documentation tabs of the Set Library dialog box. The Axis documentation files appear in the `\axis-1_1\docs` folder, while the Xerces documentation files appear in the `\xerces-2_5_0\docs` folder. The eBay Request Library documentation appears in `\eBayRequestLibrary\doc`. The JDOM documentation appears at `\jdom-b9\build\apidocs`. You must use the external viewer when working with the SDK.

7. Type a name for the library in the Name field of the Set Library dialog box and click OK. JCreator will add the library to the list of available libraries for the project.
8. Repeat steps 5 through 7 for the custom eBay, JDOM, Axis, and Xerces libraries. When you complete the steps, check only the library entries in the Project ClassPath dialog box that you actually need. Figure 10.4 shows a typical library setup for eBay Web Services.
9. Click Finish. JCreator will create a new project and workspace for you.

▶ TIP

Once you configure the libraries you need for eBay Web Services, they're available for every other project you create. All you need to do is check the required libraries. Use the eBaySDK option for SDK technique projects, the Axis and Xerces options for SOAP projects. Select the eBayRequestLibrary and JDOM options for API technique projects that use this library.

FIGURE 10.3:
Add library paths or archives using the Set Library dialog box.

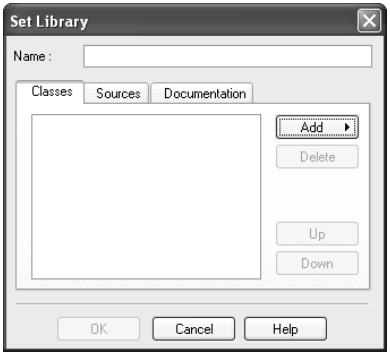
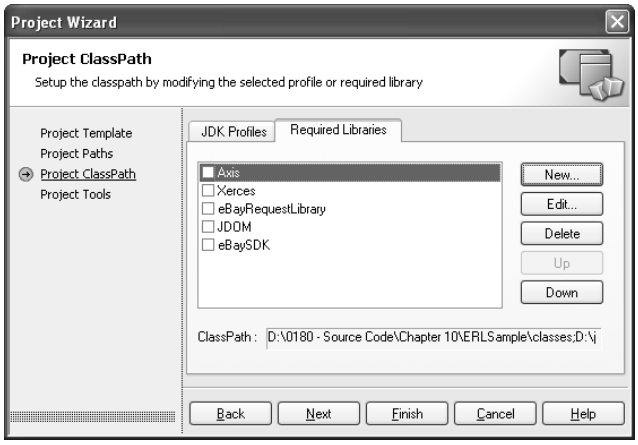


FIGURE 10.4:

Make sure you always check the libraries that you want to use with your project.



You've configured JCreator for use with eBay Web Services at this point. These steps probably seem like a lot more work than simply typing what you need into a batch file, but it's also less error prone and somewhat easier to configure. Generally, you'll find modification and updates are much easier to perform using this editor.

Writing the Application

This first example demonstrates the API technique without adding anything extra. You can call eBay Web Services and work with the data without anything more than the 1.4.2 (or above) version of the Java Development Kit (JDK). The example performs a simple `GetSearchResults()` call. Listing 10.1 shows the simple API technique for making the call. You'll find the complete source for this example in the `\Chapter 10\SimpleSearch` folder of the source code located on the Sybex Web site.

Listing 10.1 Using the Simple API Technique

```
public void actionPerformed(ActionEvent AE)
{
   ... Variable Declarations ...

   // End the program.
   if (AE.getSource() == btnQuit)
      System.exit(0);

   // Issue a request and receive a response.
   if (AE.getSource() == btnTest)
   {
      try
      {
```

```
// Define the development environment URL.
eBayUrl = new URL("https://api.sandbox.ebay.com/ws/api.dll");

// Create a connection to eBay.
Conn = (HttpsURLConnection) eBayUrl.openConnection();

// Specify the connection is for input/output.
Conn.setDoInput(true);
Conn.setDoOutput(true);

// Set the data transmission method.
Conn.setRequestMethod("POST");

// Define the header information for the request. These
// headers appear in the same order recommended by the
// documentation at
// http://developer.ebay.com/DevZone/docs/API_Doc/index.asp.
Conn.addRequestProperty("X-EBAY-API-SESSION-CERTIFICATE",
                        txtDevID.getText() + ";" +
                        txtAppID.getText() + ";" +
                        txtCertID.getText());
... Other Request Headers ...

// Build a string that contains the request.
Req =
   "<?xml version='1.0' encoding='utf-8'?>\r\n"+
   "<request>\r\n" +
   "<RequestToken>" +
      txtAuthToken.getText() +
   "</RequestToken>" +
   "<DetailLevel>0</DetailLevel>" +
   "<ErrorLevel>1</ErrorLevel>" +
   "<Query>" + txtSearch.getText() + "</Query>" +
   "<SiteId>0</SiteId>" +
   "<Verb>GetSearchResults</Verb>" +
   "</request>";

// Create the output stream.
DataOut = Conn.getOutputStream();

// Send the data.
SendIt = new PrintStream(DataOut);
SendIt.print(Req);
SendIt.close();

// Recieve the response.
DataIn = Conn.getInputStream();
DataBuf = new BufferedInputStream(DataIn);

// Build a DOM document.
BuildIt = DocumentBuilderFactory.newInstance();
```

```
   RespDoc = BuildIt.newDocumentBuilder();
   eBayDat = RespDoc.parse(DataBuf);

   // Drill down to the Items node.
   eBayLst =
    eBayDat.getElementsByTagName("eBay").item(0).getChildNodes();
   Search = eBayLst.item(3).getChildNodes();
   Items = Search.item(1).getChildNodes();

   // Set the number of rows in the table.
   RTM.setNumRows(
      Integer.parseInt
        (Search.item(3).getChildNodes().item(0).getNodeValue()));

   // Set the table row count.
   TblCnt = 0;

   // Clear the response table.
   for (int RCount = 0;
        RCount < 10;
        RCount++)
      for (int CCount = 0;
           CCount < 4;
           CCount++)
         tblResp.setValueAt("", RCount, CCount);

   // Process all the items.
   for (int Count = 0; Count < Items.getLength(); Count++)
      if (Items.item(Count).getNodeName() == "Item")
      {

         // Process an individual item.
         Item = Items.item(Count).getChildNodes();
         for (int ICount = 0;
              ICount < Item.getLength(); ICount++)
         {
            TheItem = Item.item(ICount);

            // Display the data.
            if (TheItem.getNodeName() == "Title")
               tblResp.setValueAt(
                  TheItem.getChildNodes().item(0).getNodeValue(),
                  TblCnt, 0);
            ... Other Display Statements ...
         }

         // Update the table row count.
         TblCnt++;
      }
}
catch(java.net.MalformedURLException e)
```

```
        {
           ... Lots of Error Trapping ...
        }
     }
  }
```

The most important piece of API technique code appears in the `actionPerformed()` method of the `ButtonHandler` class. The code begins by building a new URL. It then uses the URL to create a secure connection to eBay. Notice that you must coerce the output of the `eBayUrl.openConnection()` method. The code then sets the connection up for both sending and receiving data. It also sets the request method to POST using the `setRequestMethod("POST")` call and begins setting the request headers using the `addRequestProperty()` method.

Unlike many of the other API examples in the book, creating the XML request for Java is relatively easy. You create a string containing the required elements, rather than an XML document.

After the code creates the request, it obtains the output stream using `Conn.getOutputStream()` and sends the data using `SendIt.print(Req)`. The process is actually viewed as printing from the Java perspective. Make sure you close the output stream or you'll experience errors in the application later.

The next task is to retrieve the response by creating an input stream using `Conn.getInputStream()`. The code creates a buffered input stream next using `DataIn` (the input stream) as the data source. The code hasn't read the data yet; it has only prepared to read the data. The data reading process occurs as part of building a DOM document. The code uses a three-step process in this case: it creates a document builder factory, uses the factory to create a document builder, and finally uses the document builder to read the data and use the information to create a document. The `RespDoc.parse(DataBuf)` call actually reads the data in a string and converts the string to an XML document.

Reading through the document can be a frustrating experience unless you know how Java interprets the eBay input. If you look at the raw data you'll notice that every other element is a `#text` node. After working with the data for a while, it appears that Java interprets the carriage returns between the other elements as text nodes. Consequently, you can't assume that the items appear as a list of `<Item>` elements. Every `<item>` element has an associated `#text` node. The ramifications of this little oddity become apparent when you look at the code required to drill down into the various node levels including `<eBay>`, `<Search>`, and `<Items>`.

The results window has a default of 10 rows. In many cases, you can store the result set in 10 rows, but, in other cases, you'll need more. Because you can't know how many rows to create in advance, the code sets the number of rows based on the `<Count>` element using the `RTM.setNumRows()` method (`RTM` is the template used to create the result rows).

The code now needs to track which row of the table it's using with the `TblCnt` variable. You can't use the item element counter, `Count`, because the `<Items>` node includes text elements.

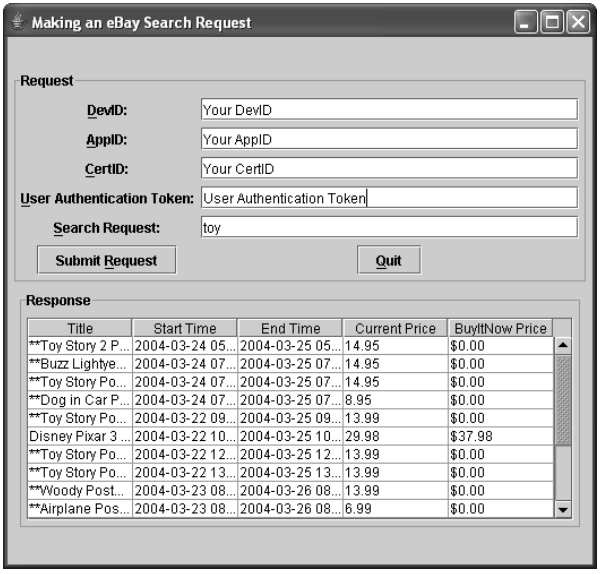
At this point, the code processes the `<Items>` node using a `for` loop. The code begins by determining whether the child node is an `<Item>` node using the `Items.item(Count).getNodeName()` method. If so, it begins processing the `<Item>` child nodes using another `for` loop. The code uses the `TheItem.getNodeName()` method to determine which data node it's processing (or whether it's another `#text` node). When the code does find a data node, it places the data in the table using the `tblResp.setValueAt()` method. Figure 10.5 shows typical output from this application.

Writing a Java Application Using the eBay Request Library

After working with the SDK technique and deciding I would rather commit hari-kari than use it with Java (it works great with many other languages) and wondering seriously if the API approach wasn't almost as bad, I stumbled on one of eBay's best kept developer secrets: the sample code at `http://developer.ebay.com/DevZone/docs/Samplecode.asp`. I would normally expect some fanfare or at least a well-placed pointer to the code in this area, but what you really need is a compass and a good map. You can't get there from anywhere—you must know it exists.

FIGURE 10.5:

The application creates typical output for the `GetSearchResults()` call.



The example in this section uses the eBay Request Library, which is perhaps the best tool in the Java developer's toolkit. I chose to replicate the `GetSearchResults()` call shown in Listing 10.1 for this example so you could compare the differences for yourself. The code you see in Listing 10.2 performs the same task, produces the same result, and took about a quarter of the time to write. Not only did I write fewer lines of code, but it was easier to write as well. You'll find the complete source for this example in the `\Chapter 10\ERLSample` folder of the source code located on the Sybex Web site.

Listing 10.2 Using the eBay Request Library

```
public void actionPerformed(ActionEvent AE)
{
   int                     TblCnt;     // Current table row.
   Properties              ReqData;    // Request properties.
   GetSearchResults        Req;        // The request.
   Result                  Response;   // Request response.
   SearchResult            AllItems;   // All of the items.
   ItemResult              Item;       // One item.

   // End the program.
   if (AE.getSource() == btnQuit)
      System.exit(0);

   // Issue a request and receive a response.
   if (AE.getSource() == btnTest)
   {
      try
      {
         // Create the request properties.
         ReqData = new Properties();

         // Set the request properties.
         ReqData.put(EBayRequest.HostURL,
                     "https://api.sandbox.ebay.com/ws/api.dll");
         ReqData.put(EBayRequest.DeveloperId, txtDevID.getText());
         ReqData.put(EBayRequest.ApplicationId, txtAppID.getText());
         ReqData.put(EBayRequest.CertificateId, txtCertID.getText());
         ReqData.put(EBayRequest.CompatibilityLevel, "305");
         ReqData.put(EBayRequest.DetailLevel, "1");
         ReqData.put(EBayRequest.ErrorLevel, "1");
         ReqData.put(EBayRequest.SiteId, "0");
         ReqData.put(EBayRequest.UserId, txtUserName.getText());
         ReqData.put(EBayRequest.UserPassword,
                     txtUserPassword.getText());

         // Create the request object.
         Req = new GetSearchResults(ReqData);
```

```
            // Add the query.
            Req.setQuery(txtSearch.getText());

            // Get the response.
            Response = Req.execute();

            // Get all of the items.
            AllItems = Response.getSearch();

            // Set the number of rows in the table.
            RTM.setNumRows(AllItems.getCount());

            // Set the table row count.
            TblCnt = 0;

            // Parse the individual items.
            for (Iterator Count = AllItems.getItems().iterator();
                 Count.hasNext(); )
            {
               // Obtain the single item.
               Item = (ItemResult)Count.next();

               // Place the values in the table.
               tblResp.setValueAt(Item.getTitle(), TblCnt, 0);
               ... Other Values ...

               // Update the row counter.
               TblCnt++;
            }

         }
         catch(java.net.MalformedURLException e)
         {
            ... Lots of Error Trapping ...
         }
      }
   }
```

The code begins by creating a series of properties to use to make the call. The `ReqData` object contains the same entries you'd normally need to add to the request header and parts of the request message. The difference is that you only need to supply the information once.

After the code creates the request data, it builds a request object using the `GetSearchResults()` constructor. The constructor accepts `ReqData` as input to configure `Req`. It's important to note that the current incarnation of the eBay Request Library doesn't support the full set of API calls. It can easily be made to support the full set, but the developer of the library hasn't created support for all the API calls yet. The call also has to add the one special option that the `GetSearchResults()` call requires, a query, using the `Req.setQuery()` method.

▶ NOTE

The current version of the eBay Request Library only supports the username and password access method. However, the code for adding a user authentication token is trivial. Since eBay makes the full source code available, you should modify it to ensure you can use the user authentication token access technique.

The code makes the request and gets the response using a single line of code, `Response = Req.execute()`. It begins to process the information immediately by recovering the `<Search>` node data. Notice you don't have to go through a bunch of intermediate steps to go directly to the `<Search>` node.

At this point, the code sets the number of rows in the response table using the `AllItems .getCount()` method. It then uses a simple `Iterator` to process the list of items. The code places each iterated item into `Item` by coercing the `Count.next()` output. It sets the various table values using the `tblResp.setValueAt()` method. The output from this application is very similar to the output shown in Figure 10.5.

Your Call to Action

This chapter helps you understand how to use Java to build an eBay Web Services application. In fact, the chapter discusses how to build several application types so you get a better idea of just how flexible Java is. The chapter has also pointed out hurdles you might encounter when building the application. Java is an outstanding language with amazing flexibility, but it also has significant problems that you can't overcome with ease. The important issue is to determine whether the benefits of using Java outweigh potential problems when you decide whether to use this language.

Begin your preparation for using Java by using the Web site URLs in the "Resources for Learning Java" section of the chapter. Once you know Java well enough, you're ready to look at the requirements for your application. Make sure you spend enough time considering the issue of whether the intelligence for your application will reside on the client or the server (or something in between). You also need to consider the server setup you want to use and decide what kind of functionality to build into your application. You might decide to start with something as simple as a search site so you can see how eBay Web Services performs, as well as how you need to configure your setup for a more advanced application.

Chapter 11 moves development from desktop, laptop, notebook, and other large devices to the small, mobile devices that many people use today. These devices are lightweight, easy to

carry, and generally allow the user to communicate everywhere. As great as these devices are for the general user, they're a problematic platform at best for the developer because they need to consider the limitations of such devices. Most mobile devices have small displays, lack full keyboard functionality, have limited memory, have reduced processing power, and have significant operating system limits. However, even with these problems, mobile devices can serve as an important platform for an eBay Web Services application and eBay certainly makes it easy to use these devices.