

Introduction

This tutorial shows you how to call the eBay Finding service using the findingKit provided by eBay. It also shows you how to update the findingKit to a newer version of Finding service when available.

FindingKit provides a simplified client interface that wraps around the standard JAX-WS proxy for the eBay Finding service. It makes it easier for a developer to interact with the eBay Finding service by hiding common tasks like xml serialization and http header setting, and by providing common facilities like message logging.

Below is a tutorial that shows you how to access the eBay Finding service using findingKit. In this tutorial, you'll build a simple console application that displays the finding result, using the findItemsByKeywords call.

Complete Source Code

The complete code is provided in the findingKit package samples\consolefinditem subdirectory.

Before You Begin

There are a few prerequisites for completing this tutorial:

1. You must have installed Sun JDK 1.6.x or above.
2. You must have installed Eclipse IDE 3.2 or above.
3. You have joined the eBay Developers Program and obtained application keys for the eBay production or sandbox environment.
4. You must have installed Apache Ant 1.7.0 or above if you want to update findingKit to a newer version of eBay Finding service.

Step 1: Create the Project in Eclipse

Create a Java project called consoleFindItem in Eclipse (see Fig 1).

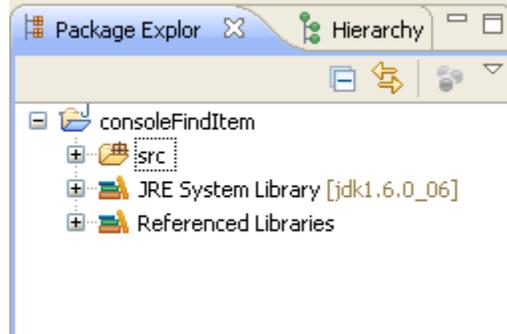


Fig 1. Create a consoleFindItem Java Project

Step 2: Add Reference

Add finding.jar and log4j-1.2.16.jar to the java build path of the consoleFindItem project(see Fig 2.). You can find these jars in the findingKit\lib folder. The finding.jar contains a JAX-WS proxy for the eBay Finding service and a few helper classes to simplify interactions with eBay Finding service. Log4j is internally used by findingKit for message logging.

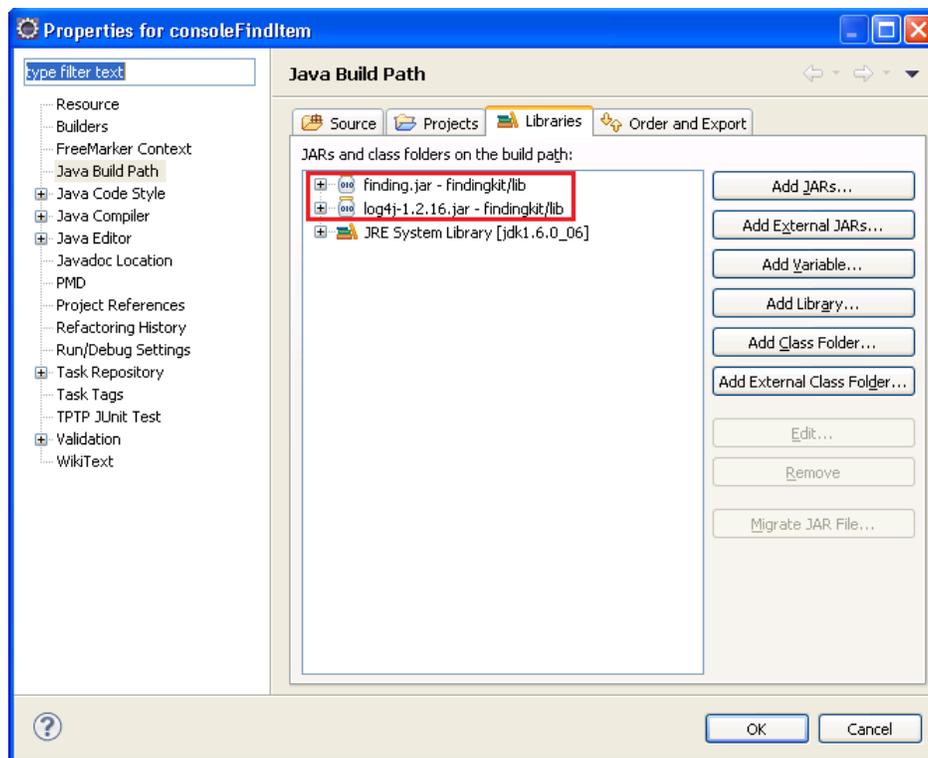


Fig 2. consoleFindItem Libraries Reference

Step 3 Create the Main Program

Create a FindItem.java file in the project and add the following code in this file (see Listing 1):

```

package com.ebay.sample;

import java.util.List;

import com.ebay.services.client.ClientConfig;
import com.ebay.services.client.FindingServiceClientFactory;
import com.ebay.services.finding.FindItemsByKeywordsRequest;
import com.ebay.services.finding.FindItemsByKeywordsResponse;
import com.ebay.services.finding.FindingServicePortType;
import com.ebay.services.finding.PaginationInput;
import com.ebay.services.finding.SearchItem;

/**
 * A sample to show eBay Finding service call using the simplified interface
 * provided by the findingKit.
 *
 * @author boyang
 */
public class FindItem {

    public static void main(String[] args) {
        try {
            // initialize service end-point configuration
            ClientConfig config = new ClientConfig();
            config.setApplicationId("Your AppID here");

            //create a service client
            FindingServicePortType serviceClient =
                FindingServiceClientFactory.getServiceClient(config);

            //create request object
            FindItemsByKeywordsRequest request = new FindItemsByKeywordsRequest();
            //set request parameters
            request.setKeywords("harry potter phoenix");
            PaginationInput pi = new PaginationInput();
            pi.setEntriesPerPage(2);
            request.setPaginationInput(pi);

            //call service
            FindItemsByKeywordsResponse result =
                serviceClient.findItemsByKeywords(request);

            //output result
            System.out.println("Ack = "+result.getAck());
            System.out.println("Find " + result.getSearchResult().getCount() + "
                items. ");
            List<SearchItem> items = result.getSearchResult().getItem();
            for(SearchItem item : items) {
                System.out.println(item.getTitle());
            }

        } catch (Exception ex) {
            // handle exception if any
            ex.printStackTrace();
        }
    }
}

```

Listing 1. FindItem.java

The program starts by importing Java and findingKit classes. Here we import ClientConfig, FindingServiceClientFactory and some Finding service types such as FindItemsByKeywordsRequest, FindItemsByKeywordsResponse.

The main function shows a typical Finding service call flow which is analyzed as following:

1. Setup Client Configuration

Client side configuration, such as application id, must be set up before the client can communicate with an eBay service. In the sample, we instantiate a ClientConfig instance and set configurations accordingly. For all supported configurations, please refer to the source of the ClientConfig class. Note, some configurations are mandatory, such as application id, while others are optional, such as enabling logging. Regarding target service endpoint address, by default the Finding service production server address will be used. If you want to point to the Finding service sandbox server address, just set the address on the ClientConfig instance to override the default one.

2. Create Service Client

An application uses the service client to communicate with a service. In the sample, we can easily get a Finding service client instance by invoking the static factory method on the FindingServiceClientFactory class.

3. Create Outbound Request and Setup Request Parameters

To call an operation on a service, you need to create a request and populate request parameters first. In the sample, we create a FindItemsByKeywordsRequest instance, and populate keyword and pagination information accordingly. For all supported input parameters of a request type, please refer to [eBay Finding Service Call Reference](#).

4. Call the Operation on the Service Client and Receive Inbound Response

Real interaction between an application and an eBay service takes place here. You call an operation on the service client and pass in the request instance as a parameter. If the call is successful, you will get a corresponding response instance. Behind the scenes, JAX-WS proxy will do the low-level message communication tasks. In the sample, we call findItemsByKeywords operation on the client, and pass in the FindItemsByKeywordsRequest instance we already created above. On a successful call, we will get a FindItemsByKeywordsResponse instance.

5. Handle Response

Once you get the response, it's up to you to decide how to further handle or present the response. In the sample, we simply log the found items number and titles to the console window. For all supported output parameters of a response type, please refer to [eBay Finding Service Call Reference](#).

6. Handle Exception

If any of the above steps goes wrong (for example, if the service call fails and throws an exception), it's the application's responsibility to capture and handle exception accordingly. In the sample, we simply output the exception to the console window.

Step 4 Configure Logging

FindingKit internally uses log4j for message logging. If you want to enable payload logging, you need to configure log4j properly. In the sample, we add a log4j.properties file in the src folder of the project to configure the log4j to send the log to the console (see Fig 3.). For log4j details and its settings, please refer to its [official site](#).

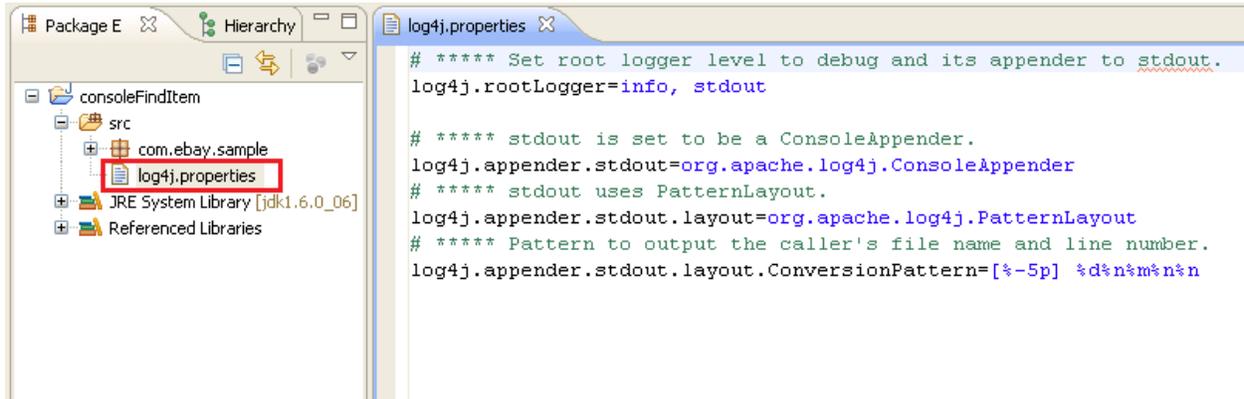
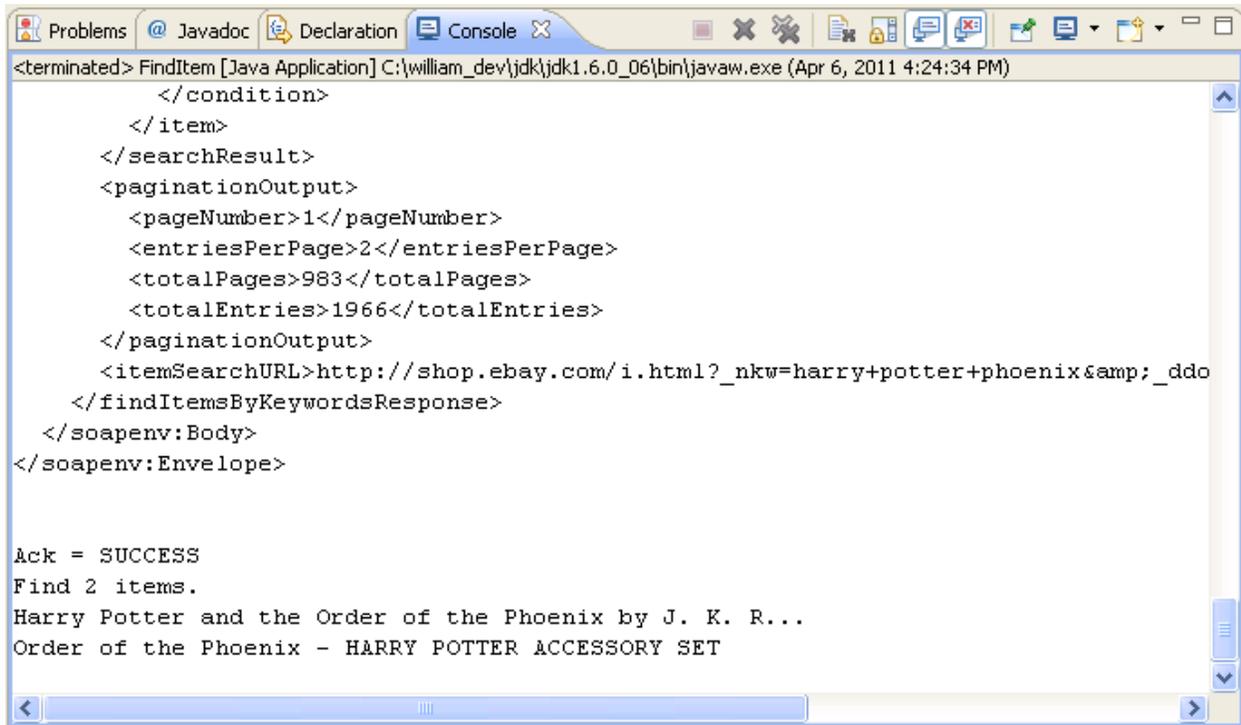


Fig 3. log4j.properties file

Step 5 Run the Application

Before running the program, you must substitute your own eBay Developer Account Application ID in the code. Then, right-click the FindItem.java file, and in the popup menu click Run As -> Java Application to run the sample.

If everything works fine, you will see results similar to Fig 4.



```
<terminated> FindItem [Java Application] C:\william_dev\jdk\jdk1.6.0_06\bin\javaw.exe (Apr 6, 2011 4:24:34 PM)
  </condition>
</item>
</searchResult>
<paginationOutput>
  <pageNumber>1</pageNumber>
  <entriesPerPage>2</entriesPerPage>
  <totalPages>983</totalPages>
  <totalEntries>1966</totalEntries>
</paginationOutput>
  <itemSearchURL>http://shop.ebay.com/i.html?_nkw=harry+potter+phoenix&_ddo
</findItemsByKeywordsResponse>
</soapenv:Body>
</soapenv:Envelope>

Ack = SUCCESS
Find 2 items.
Harry Potter and the Order of the Phoenix by J. K. R...
Order of the Phoenix - HARRY POTTER ACCESSORY SET
```

Fig 4. Console Output

Now you have a working sample that can call eBay Finding service via findingKit. Congratulations!

How to Update the FindingKit to a New WSDL

The current findingKit is built with Finding service wsdl version 1.9.0. We provide ant build script in the package to let user update (or sync) to the latest WSDL.

If a new Finding service WSDL is published by eBay, you can simply update (or sync) to the latest version as follows:

1. Backup and remove the old finding.jar file in the lib of the findingKit package.
2. Download the latest eBay Finding service wsdl and put it in the wsdl folder of the findingKit package.
3. Update the wsdl.file property in the build.properties file if your wsdl file name is not the default (FindingService.wsdl), then change the target jar name or package name if necessary.
4. In command line, change directory to the root of the findingKit and run 'ant build'.
5. Verify that the finding.jar (or other name if you override the jar name in the build.properties file) in the lib folder has been successfully built.

Now your findingKit is updated to the latest eBay Finding WSDL.