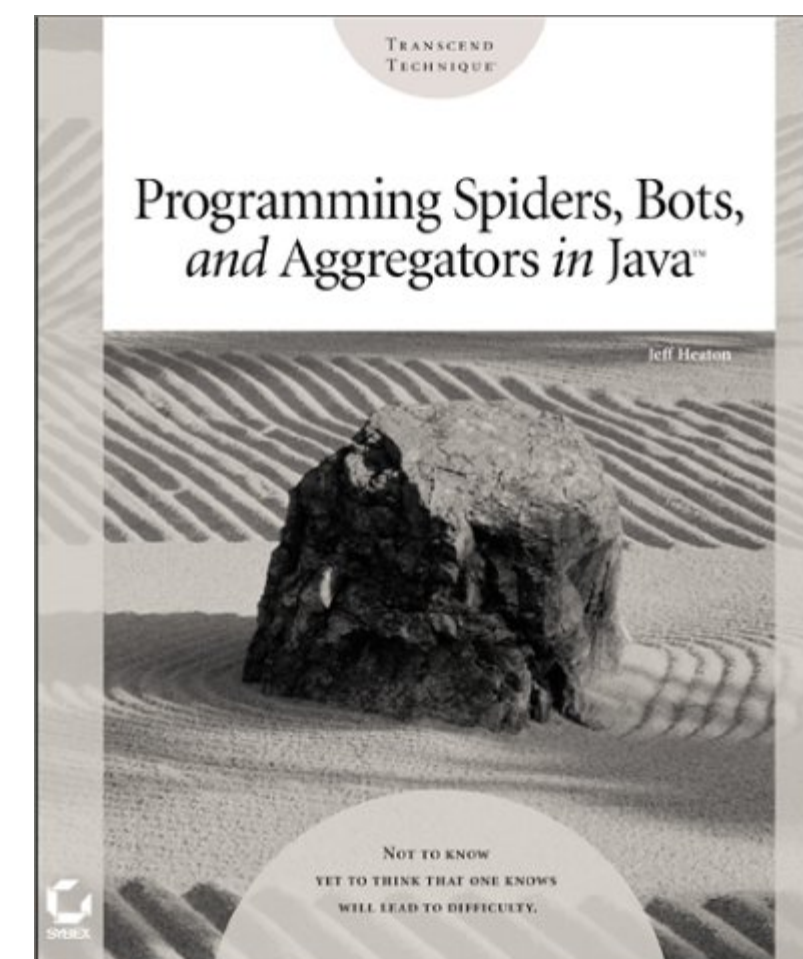


# Adaptable Agents and Bots

A research project by Jeff Heaton (heaton@heat-on.com or http://www.jeffheaton.com).

This poster shows the results of my research into Bots and Agents. This research was conducted during an independent study class, advised by Professor Bill Darte, for the Master of Information Management degree at Washington University. The specific goal of this project was to create an adaptive Bot that would overcome many of the weaknesses inherent in typical bots. I published the result of this research in the book [Programming Spiders, Bots and Aggregators in Java](#) (Sybex, 2002), shown here.



The results of this project were published in the book [Programming Spiders, Bots and Aggregators in Java](#) (Sybex, 2002). Spiders, Bots, and aggregators are all so-called intelligent agents, which execute tasks on the Web without the intervention of a human being. Spiders go out on the Web and identify multiple sites with information on a chosen topic and retrieve the information. Bots find information within one site by cataloging and retrieving it. Aggregators gather data from multiple sites and consolidate it on one page, such as credit card, bank account, and investment account data. This book offers a complete toolkit for the Java programmer who wants to build bots, spiders, and aggregators. It teaches the basic low-level HTTP/network programming Java programmers need to get going and then dives into how to create useful intelligent agent applications. It is aimed not just at Java programmers but JSP programmers as well. The CD-ROM includes all the source code for the author's intelligent agent platform, which readers can use to build their own spiders, bots, and aggregators.

About the Author. Jeff Heaton authored *Programming Spiders, Bots and Aggregators in Java* (Sybex, 2002) and co-authored *Teach Yourself Visual C++ in 21 Days, Professional Reference Edition* (Sams, 2000). Since 1996, Heaton has written numerous articles in national computer trade magazines on Internet topics. Heaton is a corporate seminar presenter and a college instructor. Currently, Heaton is a software designer for Reinsurance Group of America (NASDAQ: RGA), working with the integration of cutting edge such as JSP, J2EE, XML. Heaton has programmed for Fortune 500 companies and Internet start-ups alike. Heaton is a Sun Certified Programmer for the Java 2 Platform and a member of the IEEE.

## What is a Bot/Agent

- A computer program that browses the Internet
- Capable of doing most transactions that a human user does
- Designed to access a particular site
- Will malfunction if the underlying site changes

## Purpose of this Project, Create a Bot that will....

- Work with many similar sites
- Adapt to changes in the Web site

## How a User Uses the Internet to Tracking a Package

To create a Bot that will automate the process of looking up package shipping information we must first examine how a human user would accomplish this task. process is examined in the following figures. Here we show a user working with the United States Postal Service's Web Site. The human user could just as easily access another shipper's site, such as Federal Express, UPS or Airborne Express.

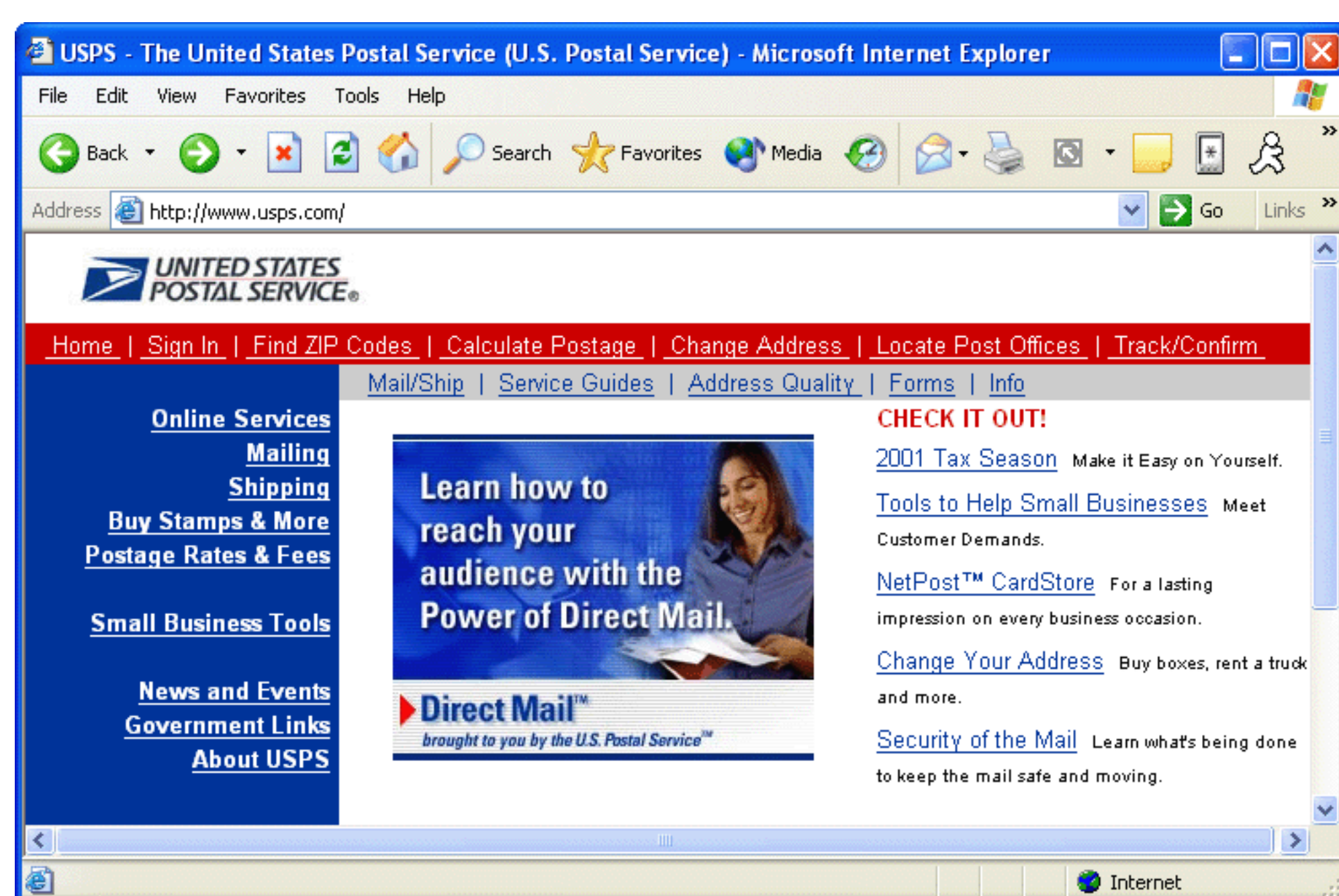


Figure 1: USPS Home Page

This is the USPS home page that a user would reach by browsing to <http://www.usps.com>. From here the user examines the choices and selects the "Track/Confirm" option on the far-right of the red bar. This takes the user to the "Shipping Center" page seen in Figure 2.

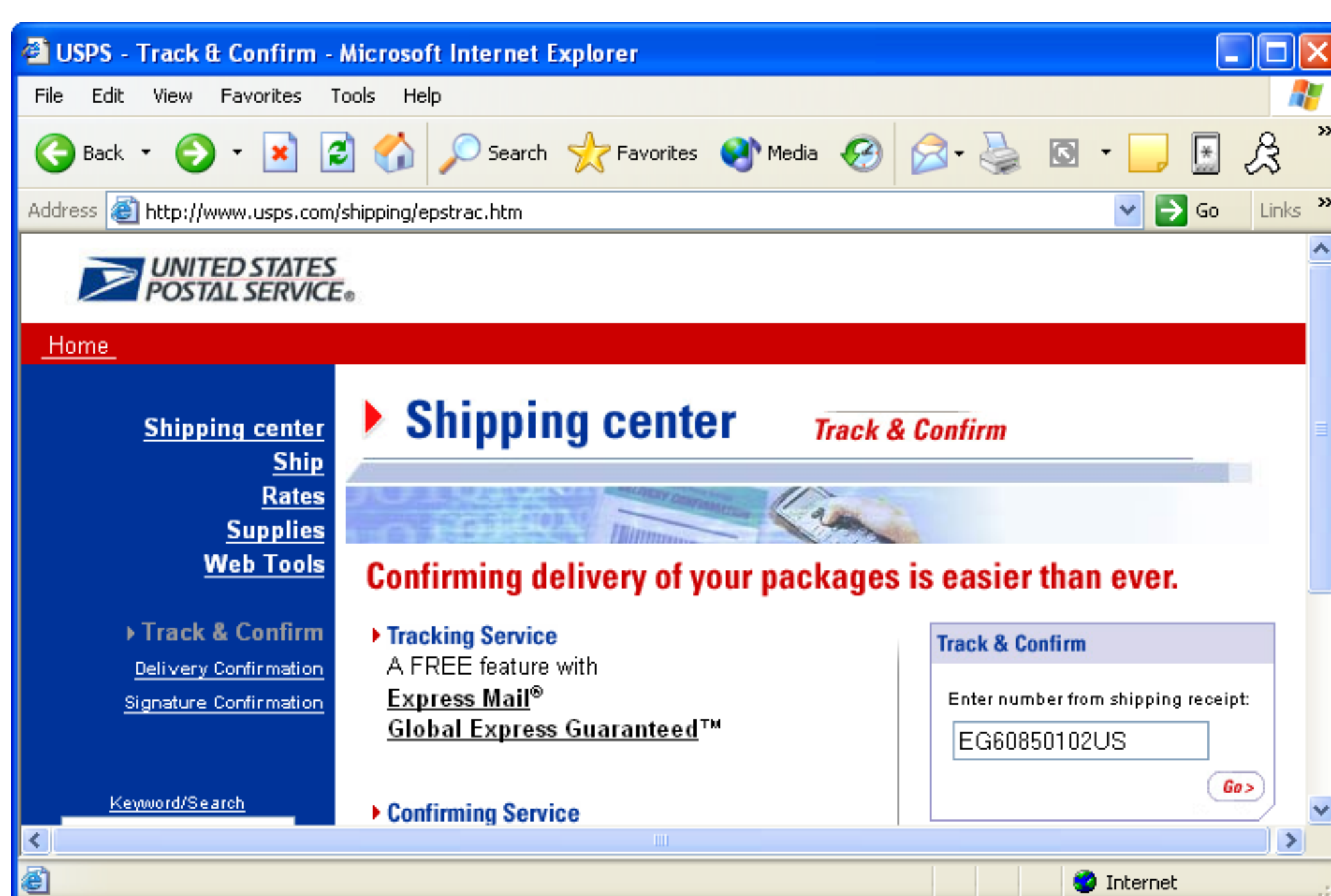


Figure 2: Shipping Center

This is the "Shipping Center" page that allows a user to enter their tracking number. Once a valid tracking number is entered the user is taken to the "Delivery Status" page seen in Figure 3.

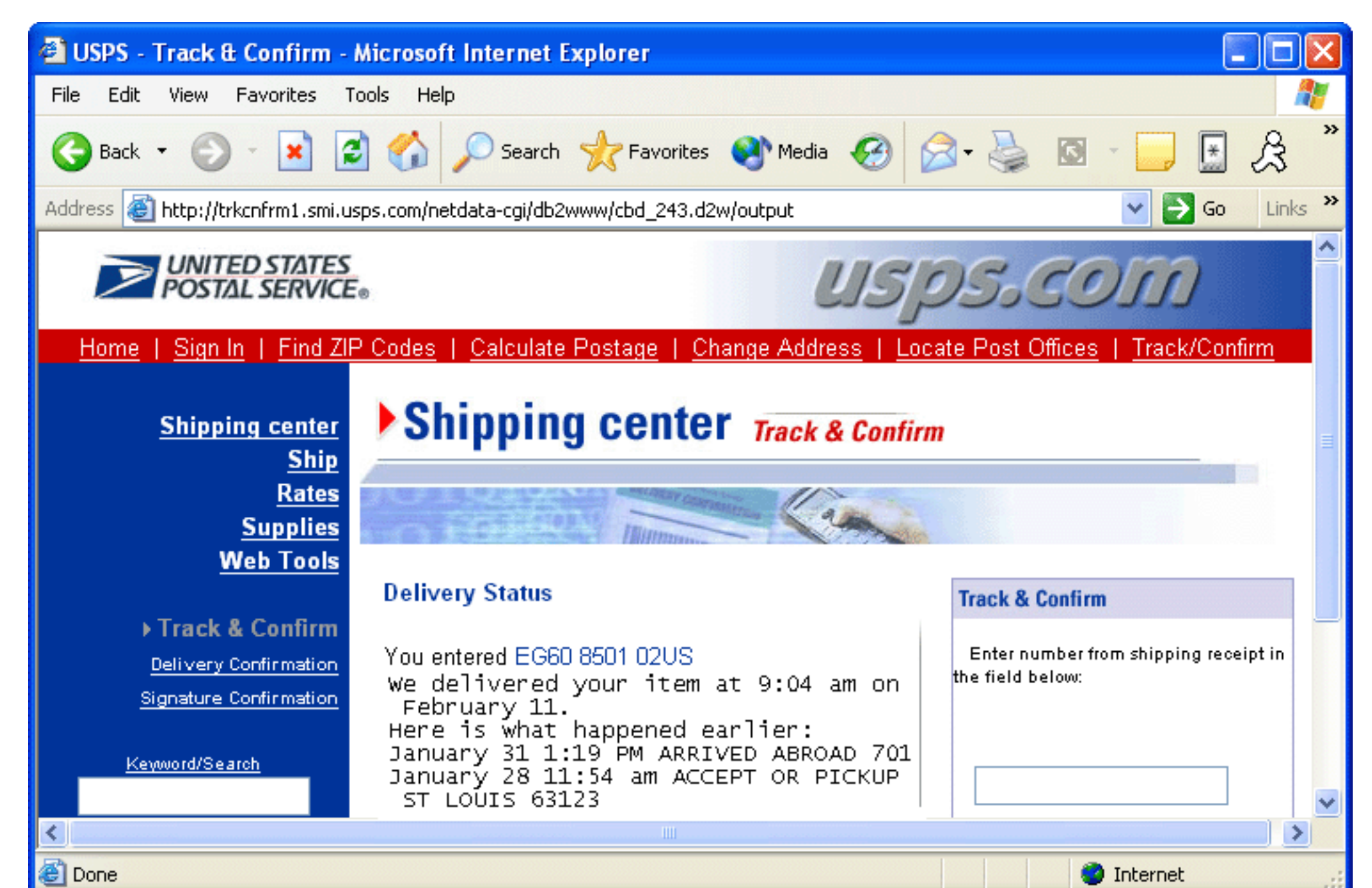


Figure 3: Delivery Status

The "Delivery Status" page is the final destination for the user. This page tells the user the status of the tracking number that they entered.

## A Typical Bot to Automate Package Tracking

Now that the programmer understands the user's path a Bot can be programmed. This Bot will carry out the users steps completely automated. Listed below are the steps that a Bot programmer would go through as they created a Bot that could retrieve shipping information. This culminates in step 4 that shows the program that implements the Bot.

### - Step 1: Identify the Users Path

1. Go to the USPS home page
2. Click on "Track Confirm"
3. Enter the tracking number
4. Read the results

### - Step 2: Identify the Web Addresses for the User's Path

1. Go to "<http://www.usps.com>" (home page)
2. Go to "<http://www.usps.com/shipping/epstrac.htm>" (Shipping Center)
3. POST tracking number to results page
4. Go to "[http://trknfrm1.smi.usps.com/netdata/cgi/db2www/cbd\\_243.d2w/output](http://trknfrm1.smi.usps.com/netdata/cgi/db2www/cbd_243.d2w/output)" (Delivery Status)

### - Step 3: Identify Unnecessary Steps

A Bot does not need to complete every item listed in Step 2. Visiting the home page (item 1) simply allows us to see the "Track/Confirm" option. A Bot can simply bypass the "Home Page" and go directly to "<http://www.usps.com/shipping/epstrac.htm>".

### - Step 4: Program the Bot

Now that the programmer understands the steps that a user follows, and unnecessary have been removed. The a Bot can be programmed that will get a tracking status. The pseudocode for a function "getShippingStatus" is shown below. This method accepts a tracking number and returns a status.

```
function getShippingStatus(trackingNum as String) returns String
begin
  //download form
  trackingForm = DownloadURL("http://www.usps.com/shipping/epstrac.htm")

  // fill in tracking number (on form)
  trackingForm.setField("trackingNumber",trackingNum)

  // Submit the completed form
  results = trackingForm.SubmitForm()

  // look for success indicator
  if results contains "delivered" and "your" and "item" then
  begin
    return "delivered"
  end
  else
  begin
    return "not yet delivered"
  end
end
end
```

## An Adaptive Bot

You have just seen how a typical Bot was created that can retrieve shipping information for a given tracking number. This Bot was designed to work only with the USPS Web site. Also this Bot is dependent on the USPS site maintaining the same structure. If the USPS site were redesigned, it is unlikely that the Bot would continue to function properly.

Addressing these two issues was the purpose of my research. I developed a Java framework that allows Bots to be constructed to be much more adaptive. This system works by using Java objects that I call recognizers.

### Using "Recognizers" to Abstract Bot Logic

A recognizer is a class that is designed to recognize a certain type of page. This recognizer looks for certain patterns that are usually present on that sort of page. My framework supports recognizers for several types of pages. Additional page recognizers can easily be added. Some of the recognizers are listed here.

- Login Recognizer - This recognizer will recognize a page that asks for a user id and password. If such a page is recognized this class will attempt to log the user in. This recognizer is not used to track shipping information.
- Enter Your Country Recognizer - Some sites will ask the user to first enter their country. This recognizer will recognize such a page and attempt to enter the user's country. The FedEx site contains such a page, though the Post Office does not.
- Hyperlink Recognizer - This recognizer will look for hyperlinks and other buttons that contain text similar to the text that was provided to this recognizer. This recognizer can be used to find links such as "Tracking", "Package Tracking" etc.
- Tracking Number Entry Recognizer - This recognizer is designed to recognize pages that contain a form for the user to enter tracking numbers into.
- Tracking Status Recognizer - This recognizer is designed to recognize the page that displays the status of a package. This page contains the final information that the Bot is looking for.

### How a Adaptive Bot Uses Recognizers

Figure 4 shows the process that my adaptive Bot uses to look at any type of site. If a different category of site is to be examined, then a different set of recognizers is loaded. One recognizer is set as the prime recognizer. The prime recognizer is the recognizer that can recognize the data that the Bot is ultimately looking for. Once the prime recognizer is satisfied, the Bot's job is done.

As the adaptive Bot tries the recognizers the path, that a human user would have used, is derived. In the event that all pages have been examined, and all recognizers have been exhausted the Bot fails.

### How the Adaptive Bot Tracks Shipments

You have already seen how a typical Bot tracks shipments. Now you will be shown how an adaptive Bot completes this process. The differences between the typical and adaptive Bot approaches can be summarized as follows:

- The typical Bot contains instructions specific to the Post Office site.
- The adaptive Bot is made up of generic "recognizers" that can work with virtually any carrier (i.e. FedEx, UPS, etc).
- The typical Bot would malfunction as a result of small changes to the Postal Service Site.
- The adaptive Bot would create a new path based on any changes to any of the shipping sites it were used with.

The exact steps that the adaptive Bot would take are:

1. Connect to <http://www.usps.com>, and run all recognizers
2. The "Hyperlink Recognizer" recognizes the "Track/Confirm" link this page will be loaded
3. The URL "<http://www.usps.com/shipping/epstrac.htm>" (the target of the "Track/Confirm" link, Figure 2 above) is loaded and all recognizers are run.
4. The "Tracking Number Entry Recognizer" recognizes the page and submits the tracking number.
5. The USPS site responds with the status of that tracking number (Figure 3 above). All recognizers are run against this page.
6. The "Tracking Status Recognizer" recognizes this page. The status is extracted.
7. The "Tracking Status Recognizer" was the prime recognizer. Processing is complete.

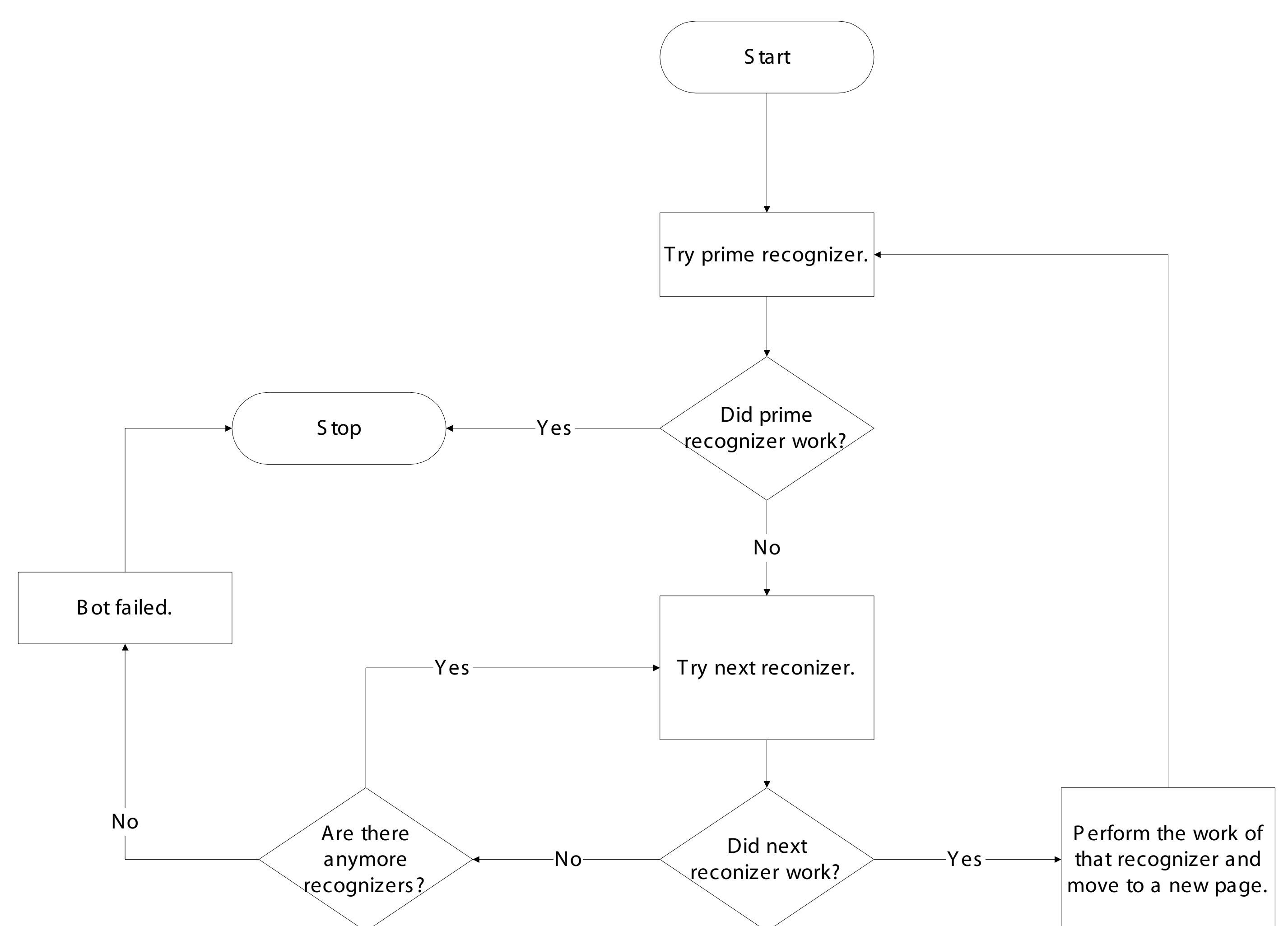


Figure 4: An Adaptive Bot

Copyright 2002 by Jeff Heaton. All Rights Reserved