When you’re building a web server with Express, most of what you’ll be doing starts

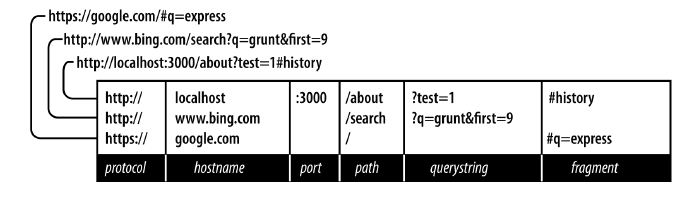
with a request object and ends with a response object. These two objects originate in

Node and are extended by Express. Before we delve into what these objects offer us, let’s

establish a little background on how a client (a browser, usually) requests a page from

a server, and how that page is returned.

The Parts of a URL



*Protocol*

The protocol determines how the request will be transmitted. We will be dealing exclusively with *http* and *https*. Other common protocols include *file* and *ftp*.

*Host*

The host identifies the server. Servers on your computer (localhost) or a local network may simply be one word, or it may be a numeric IP address. On the Internet, the host will end in a top-level domain (TLD) like *.com* or *.net*. Additionally, there may be *subdomains*, which prefix the hostname. *www* is a very common subdomain, though it can be anything. Subdomains are optional.

*Port*

Each server has a collection of numbered ports. Some port numbers are “special,” like 80 and 443. If you omit the port, port 80 is assumed for HTTP and 443 for HTTPS. In general, if you aren’t using port 80 or 443, you should use a port number greater than 1023. It’s very common to use easy-to-remember port numbers like

3000, 8080, and 8088.

*Path*

The path is generally the first part of the URL that your app cares about (it is possible to make decisions based on protocol, host, and port, but it’s not good practice). The path should be used to uniquely identify pages or other resources in your app.

*Querystring*

The querystring is an optional collection of name/value pairs. The querystring starts with a question mark (*?*), and name/value pairs are separated by ampersands (*&*). Both names and values should be *URL encoded*. JavaScript provides a built-in function to do that: encodeURIComponent. For example, spaces will be replaced with plus signs (*+*). Other special characters will be replaced with numeric character references.

*Fragment*

The *fragment* (or *hash*) is not passed to the server at all: it is strictly for use by the browser. It is becoming increasingly common for single-page applications or AJAX heavy applications to use the fragment to control the application. Originally, the fragment’s sole purpose was to cause the browser to display a specific part of the document, marked by an anchor tag (<a id="partName ">).

HTTP Request Methods

The HTTP protocol defines a collection of *request methods* (often referred to as *HTTP*

*verbs*) that a client uses to communicate with a server. Far and away, the most common

methods are GET and POST.

When you type a URL into a browser (or click a link), the browser issues an HTTP GET

request to the server. The important information passed to the server is the URL path

and querystring. The combination of method, path, and querystring is what your app

uses to determine how to respond.

The data for GET request is at queryString

The data for POST request is at request.body