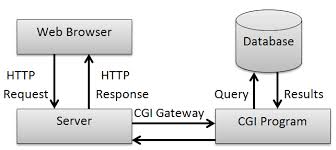
# Java EE and servlets

## Web server and CGI



CGI (common gateway interface) is an interface specification between a web server (HTTP server) and an executable program of some type that is to handle a particular request. It describes how certain properties of that request should be communicated to the environment of that program and how the program should communicate the response back to the server and how the server should 'complete' the response to form a valid reply to the original HTTP request.

Programs implementing a CGI interface can be written in any language runnable on the target machine. They must be able to access environment variables and usually standard input and they generate their output on standard output. Compiled languages such as C were commonly used as were scripting languages such as Perl, often using libraries to make accessing the CGI environment easier.

One of the big disadvantages of CGI is that a new program is spawned for each request so maintaining state between requests could be a major performance issue. The state might be handled in cookies or encoded in a URL, but if it gets to large it must be stored elsewhere and keyed from encoded url information or a cookie. Each CGI invocation would then have to reload the stored state from a store somewhere. For this reason, and for a greatly simple interface to requests and sessions, better integrated environments between web servers and applications are much more popular.

## Web server vs web container vs application server

A web server is a software that serves HTTP request, usually by returning HTML pages. A pure web server can deliver only static web pages; however, with add-on modules, a web server can serve dynamic web.

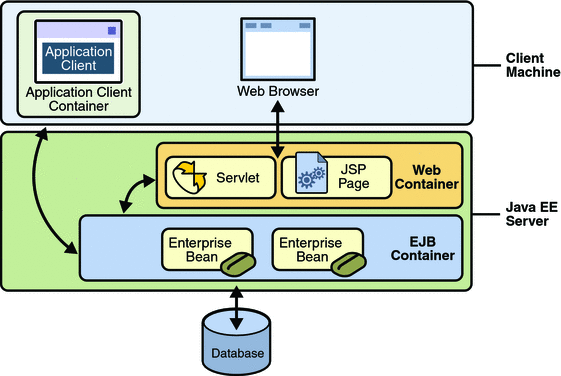
A web container is a web server with servlet, JSP technology. A web container, or a servlet container, manages Java classes, called servlets, that generate dynamic HTML pages for HTTP requests. Receiving a HTTP request, a web container will call a corresponding servlet that handle the request. A more advanced technology is Java Server Page (JSP). (Note: a servlet is a Java class with HTML code, a JSP is a HMTL page with Java code).

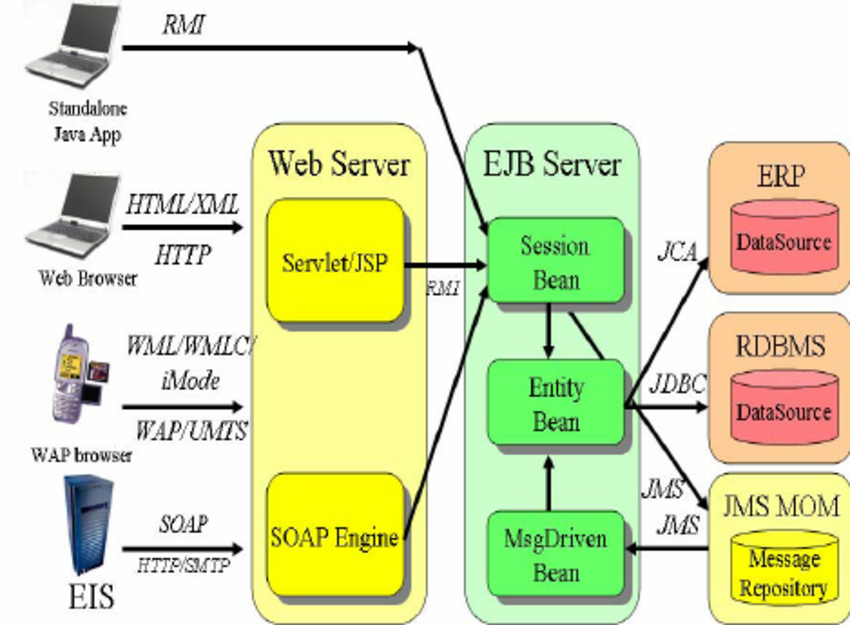
An application server is a web server with Java EE technology; it has two components: Web container for processing presentation layer on server and Enterprise Java Bean (aka Application container) for processing business logic on server.

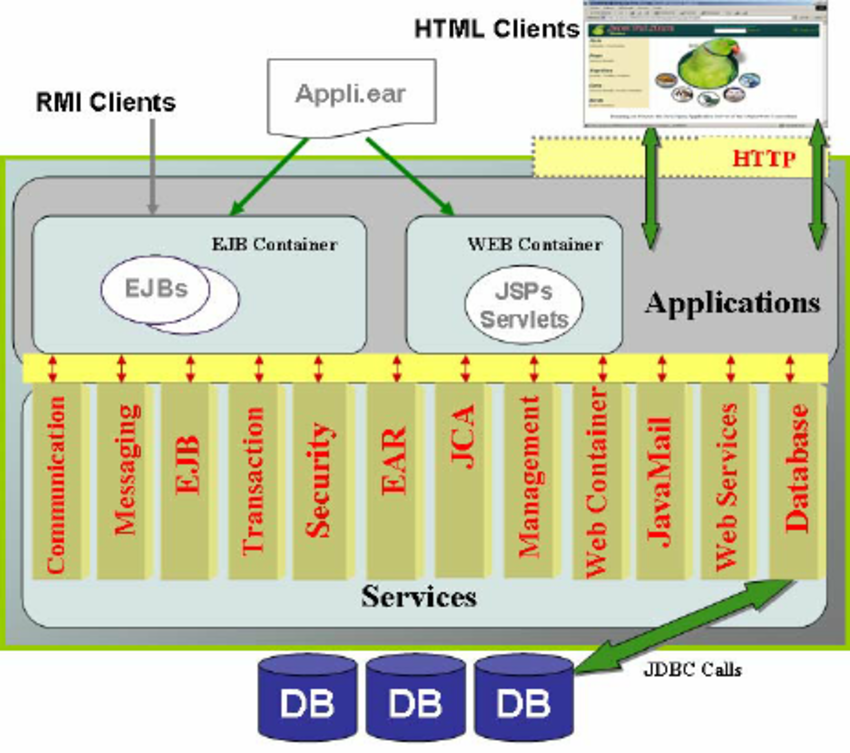
Most popular:

* Web server: Apache, NginX, IIS
* Web container: Tomcat, Jetty
* Application server: WebLogic (Oracle), Websphere (IBM), WildFly (Red Hat), GlassFish (Oracle)

## Java EE







software that serves HTTP request usually by

### Web container (apache tomcat, jetty) vs web server (apache httpd)

"Apache" is the name of a foundation that write open-source software.

Apache web server (apache HTTPD server) is a web server written in portable C. It mostly serves static content by itself, but there are many add-on modules (some of which come with Apache itself) that let it modify the content and also serve dynamic content written in Perl, PHP, Python, Ruby, or other languages.  
  
Apache Tomcat is primarily a servlet/JSP container. It's written in [Java](http://www.javaranch.com/). It can serve static content, too, but its main purpose is to host [servlets](http://www.coderanch.com/forums/f-7/servlets) and JSPs. Although it's possible to get Tomcat to run Perl scripts and the like, you wouldn't use Tomcat unless most of your content was Java.  
  
It's actually possible to use both Apache and Tomcat together, so that Apache serves the static content, and Tomcat the Servlets and JSPs. Depending on various factors, this may or may not be a good idea.

# Tomcat

## Configuration

In the following, a “**Context”** meansa web application.

### Directories and Files

These are some of the key tomcat directories:

* **/bin** - Startup, shutdown, and other scripts. The \*.sh files (for Unix systems) are functional duplicates of the \*.bat files (for Windows systems). Since the Win32 command-line lacks certain functionality, there are some additional files in here.
* **/conf** - Configuration files. The most important file in here is server.xml - main configuration file for the container.
* **/logs** - Log files are here by default.
* **/webapps** - This is where your webapps go.

### CATALINA\_HOME and CATALINA\_BASE

Throughout the documentation, there are references to the two following properties:

* **CATALINA\_HOME**: Represents the root of your Tomcat installation, for example /home/tomcat/apache-tomcat-9.0.10 or C:\Program Files\apache-tomcat-9.0.10.
* **CATALINA\_BASE**: Represents the root of a runtime configuration of a specific Tomcat instance. By default, CATALINA\_HOME and CATALINA\_BASE point to the same directory. Set CATALINA\_BASE manually when you require running multiple Tomcat instances on one machine.

If you set CATALINA\_HOME, CATALINA to different locations, the CATALINA\_HOME location contains static sources, such as .jar files, or binary files. The CATALINA\_BASE location contains configuration files, log files, deployed applications, and other runtime requirements. Because all instances with single CATALINA\_HOME location share one set of .jar files and binary files, you can easily upgrade the files to newer version and have the change propagated to all Tomcat instances using the same CATALIA\_HOME directory.

### Start up Tomcat

Configuration and start-up is described in [RUNNING.txt](https://tomcat.apache.org/tomcat-9.0-doc/RUNNING.txt)file in the Tomcat folder.

### How to Use CATALINA\_BASE

The CATALINA\_BASE property is an environment variable. You can set it before you execute the Tomcat start script, for example:

* On Unix: CATALINA\_BASE=/tmp/tomcat\_base1 bin/catalina.sh start
* On Windows: CATALINA\_BASE=C:\tomcat\_base1 bin/catalina.bat start

## Tutorial

https://javajee.com/category/java-web-java-ee

## Servlets and JSP

## Web container (tomcat, jetty) vs Application server (glassfish, )

Java EE = Web Container + Enterprise Java Bean + other services

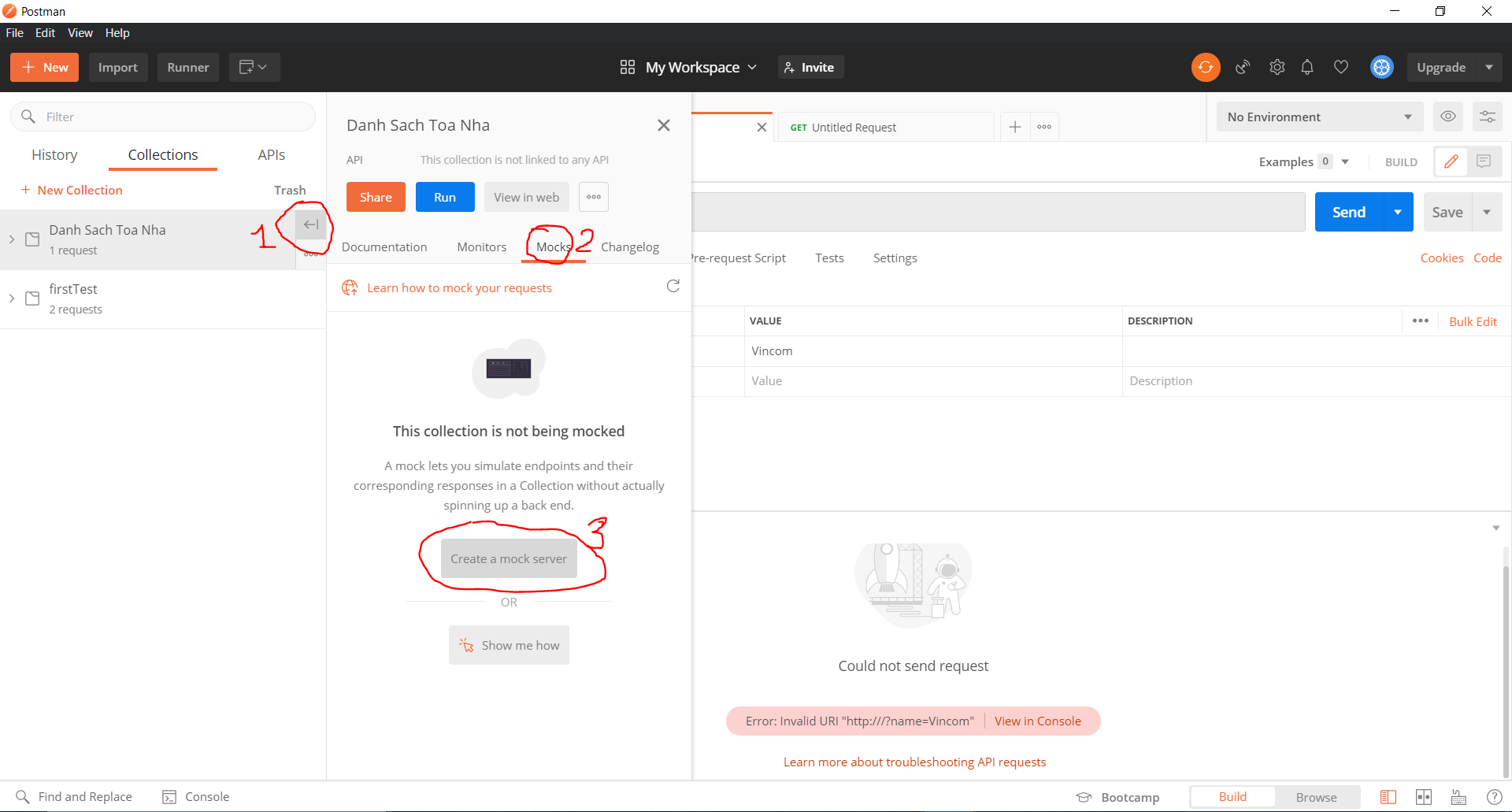
# POSTMAN - Create a mock server, mock response

To create a mock server and then some mock response from the server, you need to:

* Create a mock server
* Add a request that has a link to the server

## Create a mock server

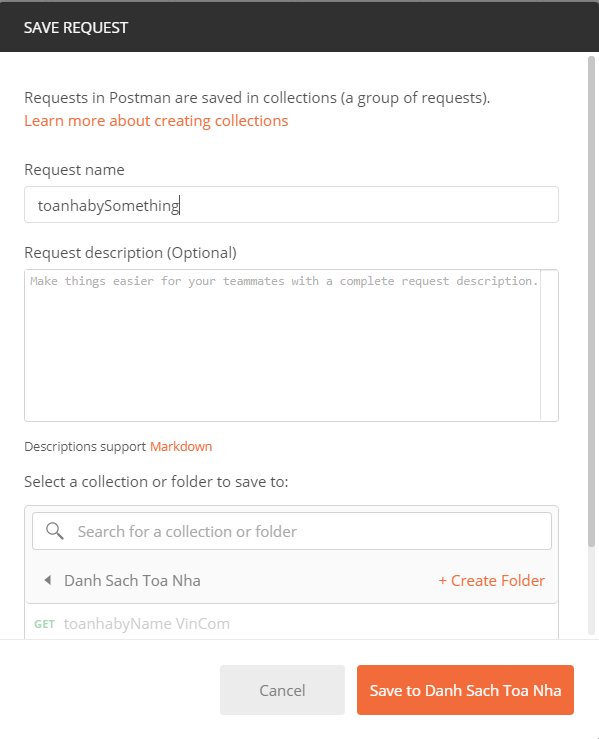
Create a collection first and then create a mock server or you create a mock server based on an existing collection:



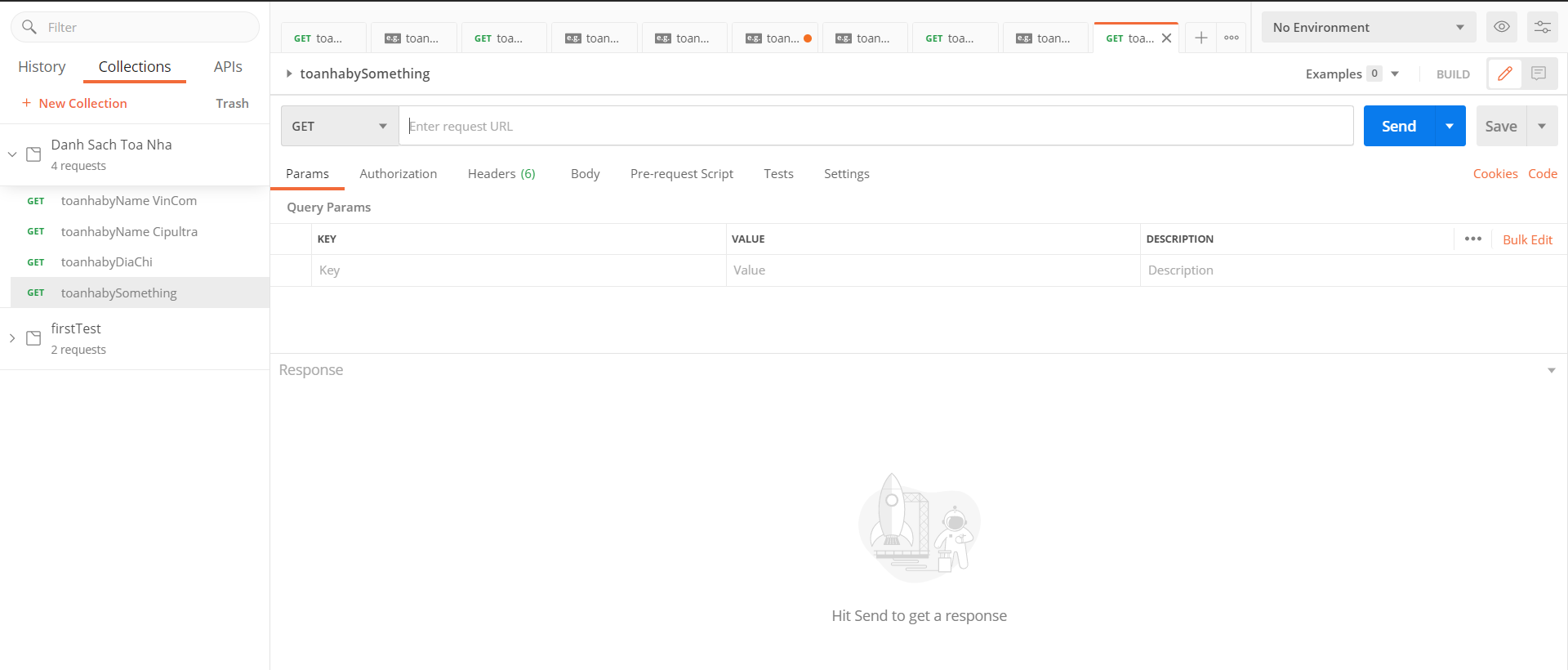
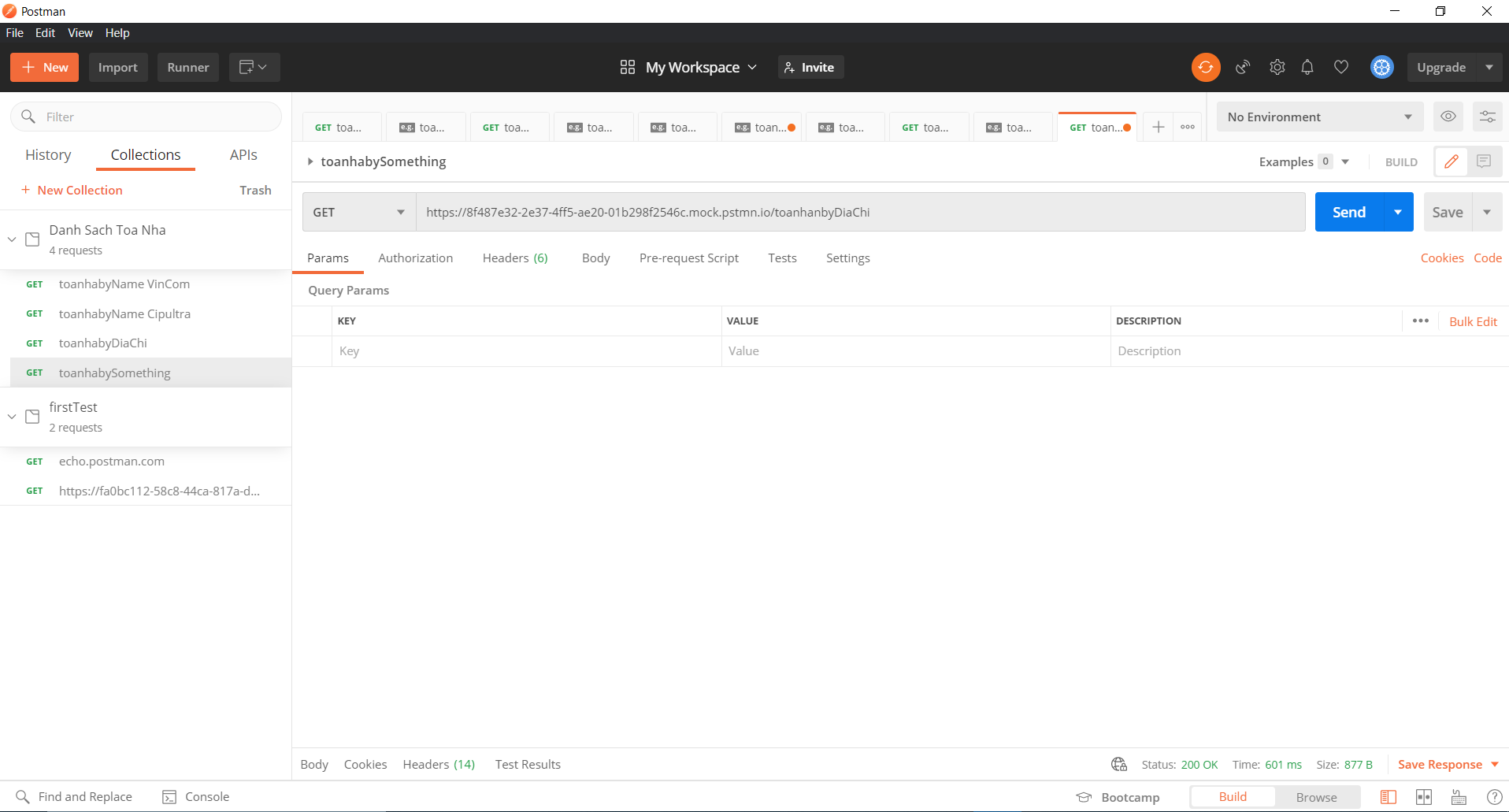
At the last step, note the URL for the mock server, which is needed in the next step.

## Add a request-response example

When you add a request, add it to a collection

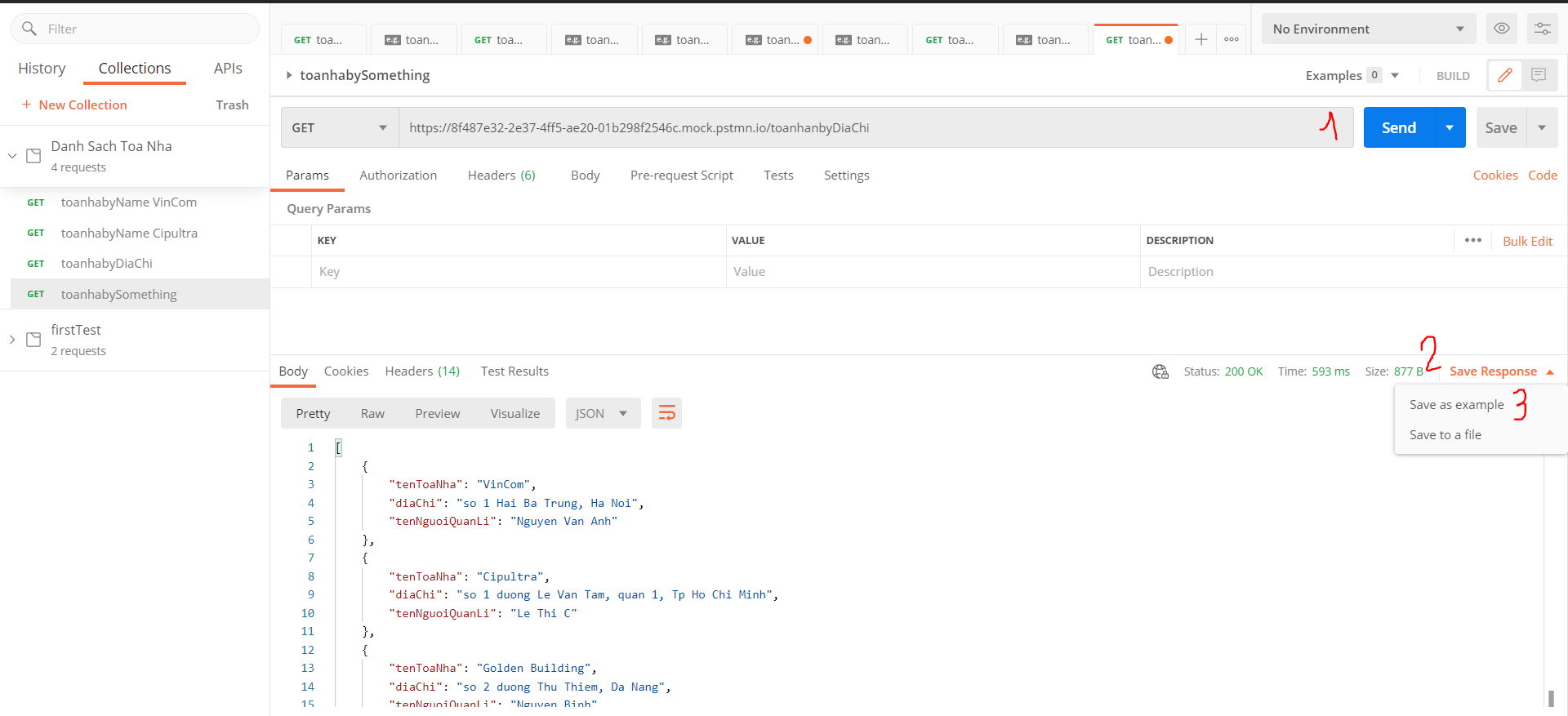


The newly added request has only Query Parameter, but not yet the Response like this:

or 

To open the Response part,

* Step 1: add the address that you created in the “Mock Server” stage
* Step 2: Hit SEND and then “Save Response”/”Save as example”



If the step above is right then the Example Response as below should appear

