PI Infrastructure

The Project Implicit (PI) site, *implicit.harvard.edu*, draws over a million volunteer study participants (Ps) per year. A typical study takes about ten minutes to complete and, over ten years, two billion rows of data have cumulated from 14 million study sessions. This overview introduces mechanisms underlying Project Implicit infrastructure to researchers developing online studies on this platform.

Let's see what PI has to offer. Open the test link https://pi.psyc.virginia.edu/implicit in a web browser. Although this link resembles the official site (also known as *production* or *prod*), test data are not stored permanently or reported. Click on Demonstration and then click on go to the demonstration tests. Then, after you assent to I wish to proceed, click to start any one of 14 Implicit Association Test (IAT) studies[1]. The *startpage* displays a bright green checkmark and a link that opens the study popup window that appears over the main browser window. You can quickly complete the study by clicking the OK and Continue buttons, skipping all questions. On the test site, you can skip IAT blocks by pressing Enter. Soon you'll reach a final debriefing page that displays a bar chart. Notice that the popup closes and that the final debriefing page appears in the primary browser window.

The IAT is PI's raison d'être but, since inception, various surveys and other task paradigms have been introduced in the "Research" part of the site.

The *startpage* is required in PI research pool studies. However, some studies for other samples don't use the popup and the study is placed in the main browser window. Additionally, one can frame the study within the main browser so that the address location never changes.

Without contributing much data, you navigated from the start of the study to the finish and saw the following *tasks*: Intro, Instructions, IAT, Demographics, Questionnaire, and Debriefing. The study session which exists in the memory of both the web browser and the web server and ensures that the study context is maintained as Ps navigate from task to task in the browser. Some tasks (e.g., IAT, Demographics) display content and gather data from Ps while others (e.g., Intro, Instructions) simply display content. The development of a *Study*, in essence, is the development of *Task* components. On the PI system, *Study* and *Task* are represented as text files that are modified using a text editor. Text files and editors are over 50 years old, essential for software development, and likely will remain readable centuries into the future.

If Ps take an extended break for say 30 minutes, the study session can time out and be lost; navigation to the next task would no longer be possible. Although not done commonly, it is possible to program a task to send a request to the server every 5 minutes to keep the study session alive. An advanced feature called *Continuation* is now available for studies that use login ids. Ps can resume very long studies from different computers, across different occasions.

Although word processors can be used to edit files, they use proprietary formats that are not compatible with the open source software used by web browsers and web servers. Modern text editors handle plain vanilla text as well as the complex characters of world languages. They do so by using utf-8 encoding that is explained here.

Had you responded to the questions and completed the IAT, you would have contributed 246 rows to the database. Understanding how data are generated by Ps, transmitted to the server, and

[1] The Study file has optional properties to prevent direct linking. Ps are required to view the pages preceding *selectatest*; that is a demo IAT study cannot be directly started by clicking on a link from a search engine or third party web site.

represented in the database is foundational. For example, consider subject 13256204 who completed the age IAT. The subject number or *session_id* is automatically assigned each time a study session is started. It is sequential and uniquely associated with the particular browser session initiated by the participant. If this participant starts a different session (e.g., retakes the age IAT or does another IAT), a new *session_id* is created. The data are accessed as four tab-delimited text files: *sessions.txt* (1 row), *sessiontasks.txt* (8 rows), *explicit.txt* (37 rows), and *iat.txt* (200 rows). Notice that *session_id* is present in every row of each file and makes possible the construction of one consolidated file with one row per *session_id*.

In research pool studies, there is a additional file called *demographics.txt* that has a *user_id* in every row. When a participant signs up for the research pool they fill out a set of demographics that are stored in this file. The contents of this file are not associated with any particular session. When a user logs in and is assigned a research study, their *user_id* is saved in *sessions.txt* for that particular study to allow merging of user demographics with study data.

Let us examine this process in more detail, using the *Firefox* web browser with the *httpfox* plugin. *Firefox* is a program in your client computer and, when active, occupies memory and uses processing time. At the other end are server computers housed in a Harvard data center in downtown Boston. The *Apache* and *Tomcat* web server programs are active on these machines, occupying memory and using processing time. For redundancy and scalability, the PI infrastructure has two identically configured computers, *atomistic* and *gestalt*, hosting the web servers. Requests from client computers are redirected to one or the other computer by a hardware load balancer. A client browser will consistently use a particular web server as the load balancing logic depends on client *IP address*. Click here for the name of your web server. In addition to the two web servers, specialized *Oracle* database servers store PI data. During operation, *Tomcat* communicates with *Oracle* to save and retrieve data.

If one is unavailable, the other one can pick up the slack. And, if there is a need we can scale up by adding web servers.

A unique IP address is associated with every machine on the internet. This address looks like www.xxx.yyy.zzz where the www etc. are numbers from 0 to 255. The IP address is used to route internet traffic. Access to particular research studies can be restricted using IP address patterns.

In client server applications, the client web browser makes a request (e.g., start the study) to the server application. The server responds by providing the requested information if it is available; otherwise it displays an error. The memories of the client and the server programs contain varied representations, including study content and user data. Study content on the server is packaged in a folder containing text files and images. When a study is started for the very first time, the study contents are loaded into memory and then served as task pages to the client browser. For each task, a request needs to be made by the client browser. This usually happens when the the study is started or when the OK or Continue button is clicked to proceed to the next task.

This material is abstract and hard to visualize. As you begin to develop studies, things will start making sense and you can revisit this document.

All requests by the client to the server are made using the secure https protocol. Thus, the links all begin with https://, not http://. Information exchanged between the client browser and the

Our security and privacy notice can be viewed here.

server are encrypted and cannot be read by others. The lock icon on the browser provides details of the *secure sockets layer* (SSL) certificate and assures the user that the site can be trusted. This encryption/decryption at the user's end is done by the client's browser. At the server, SSL processing is handled by *Apache*. Other than SSL processing, *Apache* doesn't do anything else. It transmits the unencrypted form of the request to *Tomcat* on the same machine. *Tomcat* processes the information from the client and returns the results of its processing that is to be returned to the user *Apache*, unencrypted. *Apache* encrypts the results and sends it to the user who receives it within tens or hundreds of milliseconds. All of the study software and content is processed by *Tomcat*. Both *Apache* and *Tomcat* are open source software, as is the operating system in which they operate, *Linux*.

The Apache foundation has played an enormous role in the expansion of the internet by producing robust, free, open-source software.

Arguably the most important piece of server software is the commercial *Oracle* database that runs on specially configured servers. Both the *production* and data *warehouse* instances of the *Oracle* database exist on these servers. The *warehouse* is an ever-expanding store of all the web data ever collected by Project Implicit. The *warehouse* does not interact with *Tomcat*. In contrast, the *production* database actively interacts with *Tomcat* but only contains data from the past 3 months. When *Tomcat* needs to save the 40 rows of data from an IAT block that it received from a participant, it sends these data to the *production* database. To maximize performance and security, researchers do not have direct access to the *production* database. When researchers access data, they do so by downloading it from the *warehouse* using the RDE. Each night, data is transferred from the *production* database to the *warehouse*. At any given point in time, the warehouse contains data up to approximately 2 pm of the previous day. Do not be alarmed if your study is up but no data appears the following day. Do express concern if your data does not surface even after a couple of days.

The size of the production database is kept relatively small for performance reasons. Recall that it is production and not warehouse that interacts extensively with Tomcat on both atomistic and gestalt.

Researchers do not build their studies in the production environment at Harvard. The studies are built and tested on other servers and then transferred to the production environment by a system administrator when they are ready. This maximizes the stability, cleanliness, and security of the production environment. You will be directly interacting with test servers that are hosted at UVA. The main test server is *dw2.psyc.virginia.edu*; a secondary test server, used in training, is *pi.psyc.virginia.edu*. Researchers are provided accounts and develop/test studies using a secure ftp client to transfer files and a text editor to modify them. The status of research studies on the Harvard production site can be tracked at a very granular level by using a link such as this.

The user variable in the details.jsp link is somewhat misleading. All studies and/or tasks that match the user value are listed in the page. Here, user= *attitudechange* and therefore any study or task that contains the string attitudechange in the study or task id is displayed.

Study development occurs in the *dev2/user/yourname/yourstudy* folder. You edit files on your computer and upload them to your dev2 study folder. Once your study is done and tested and has approval from the "powers that be" it is ready to be deployed. The approval process varies depending on where the study is being posted. Studies in the PI research pool, for example, must conform to design principles that help keep that volunteer participant pool active and engaged in research. For private studies, on the other hand, the approval process is more exclusively a technical evaluation of whether the study materials are prepared properly. Ultimately, however, responsibility for testing and confirming that the study operates as intended entirely rests with the individual researcher. In particular, if you develop studies for others, ensure that they certify in writing (e-mail is fine) that the procedures work as intended and the data are valid.

We have adopted strict rules about naming folders, files, variable names and task ids and so on. Many of these rules are not logically necessary but are essential to minimize errors and keep efficiency high. The details of these rules are covered elsewhere but in a nutshell: short but informative names in lowercase, no funny characters except periods and hyphens.

Once a study is approved, there is a staging point on the same test machine called *dev1* that is used for final testing by the administrator; this testing is usually very cursory and fleeting but can be extensive on occasion, as in the launch of a entire new section such as the forthcoming mental health site. Now the studies are deployed to production and an email is sent to the core developers and the researcher. On production, research studies can either be part of the research site pool or they can be standalone studies that are accessed by participants who are directed to them using a web link. These standalone studies can be deactivated using the *disable* attribute in the Study file. If testing has been done properly, data will start to accumulate and researchers will be able to access it using the *Virtual Lab*.

Here are some questions to check your understanding

1. What is the difference between *sessions* and *sessiontasks*?
2. How does *Apache* process incoming requests from participants?
3. What are the main functions of the *dw2* server? the *pi* server?
4. How do the *production* and *warehouse* database instances differ?
5. What are two types of research studies?
6. Contrast a web browser with a web server? Can both co-exist on the same computer?