Manual Suite 1.2

Sander Latour, ILLC LaCo

August 31, 2013

1 Entities

Certain concepts are used thoughout the system, these concepts are explained in this section.

1.1 User

Users are represented in the system by means of their entry in a database and their session file. A session file is a text document on the server that is automatically created for each new user and contains user specific information. Because this web framework is stateless there is no persistant memory of earlier interactions, other than manifestations of that in the database or on the harddrive.

However, by allowing each request for interaction to be annotated with a user specific key, the system has a way to restore earlier memory linked to that user. This key is usually called the session id and is either stored by a browser in a cookie or remembered in a different way. Most of this happens automatically due to conventions that the webserver, php and the browser hold on to. In the session file, that can be loaded with the session id, the system stored the current required language and the database identifier that is used to represent that user in the database[1].

The database contains information about users' browser capabailities and furthermore provides each user with a primary key that can be used to refer to in all other tables that contain information regarding some interaction with that user. It is that primary key that is also stored in the session file.

1.2 Experiment

The key entitity in the system is the experiment. Experiments are represented as a folder containing specific script files. These script files implement the various phases that are described in section 2. If a phase is not required for an experiment the script file for that phase should be ommited. To store interactions of a participant inside the experiment, the implementation can send events to this system to have them stored. More information on this can be found in sections 1.4 and 3. This system can support many different experiments at the same time. An additional parameter[2] was added the URL scheme of this system to provide the experiment scope. The value of this paremeter is seen as the experiment identifier and is also equal to the name of the folder.

1.3 Door

A door is an entry point to the system. Doors are created to differentiate between various sources of participants. For doors to be effective, they eventually need to specify which experiment to load. This can be done indirectly by letting users click on links to initiate the loading of the correct experiment, or directly by setting the internal variable in case no further user interaction is necessary. Doors can also set a specific return location to which users are sent after completing the phases of the experiment. This can be useful if you would like to offer users to perform a new experiment, or when there are certain actions to be performed afterwards that are specific to the source of the participant (i.e. Participants that are coming from Mechanical Turk need to be send back to that system in order for them to be paid). More information can be found in the `/index.php` file and in section 2.

[1] Table `wex_users`

[2] The parameter is called and referred to as `loader`

1.4 Event

Events are used to communicate the begin and end of experiment instances and the data that is gathered in between. These are stored in the database in two separate tables. One table[3] stores the instances with the start and end times, specific instance parameters and participant identifications, and the other table[4] stores the data that the experiment gathers. For more information on how the events are communicated to the system see section 3.

1.5 Survey

This system supports for pre and post surveys that are connected to the experiments. The surveys are specified in the phase-specific script files in the experiment folder in the form of HTML. When the participants submits the answers to the survey the same script file is executed which will then store the answers in the database. Survey answers are stored in the table[5] with a row per question. That allows for supporting surveys of arbitrary length and opens up the possibility of reusing answers of a participant to a question that was given in a survey of a different experiment.

2 Main Flow

When a person is send to this system to participate in an experiment, he or she is put through a certain flow that contains specific phases. These phases are all optional and are implemented depending on the experiment. Each phase describes the purpose of the content that must be generated and send back to the user's browser. The user will then be inside that phase until a explicit request is send to move to the next phase. The only exception is when a phase is requested that is not implemented for a particular experiment, because in that case the system will automatically move on to the next phase. This section describes these phases.

2.1 `preview` and `begin` phase

The preview phase allows for participants to get some information on the experiment they will be performing in before actually starting on it. It usually also checks if the user has support for all the browser features that are required for the experiment. This is specifically useful in systems like Mechanical Turk.

Since the content generated for the preview phase is usually very close to the content generated for the begin phase, both phases are expected to be implemented in the `begin.php` script file. The requested phase is still available in a variable that can be accessed in that script file, which can be used to make the distinction. In most cases the distinction is that in the preview phase the generate content has no button to move to the next phase in order to start the experiment, whereas the begin phase does generate that button.

More information on the specific script file can be found in section 6.4.1.

2.2 `pre` phase

The pre phase has the purpose to generate any content that the user needs to interact with before the experiment takes place. Usually this pre interaction is a survey. For more information on the specific script file see section 6.4.2

2.3 `game` phase

The game phase refers to the actual experiment, which was usually in the form of a game. The content generated here should allow the user to interact with the experiment and send events to this system to track those interactions. Typically the game phase results in loading a java applet, but that is completely up to the experiment's implementation of this phase. For more information on this specific script file see section 6.4.3.

2.4 `post` phase

The post phase highly resembles the pre phase, except that it involves content that needs to be interacted with after the experiment.

[3] Table `wex_instances`
[4] Table `wex_data`
[5] Table `wex_surveys`

2.5 `end` phase

The end phase is not implemented by the experiment. It is merely an indication to the system that the experiment flow is finished. In case a specific return URL has been set, for example by a door, the user is returend to that location.

3 Event handling

Events that are generated during the experiment can be send via a HTTP POST to `http://domain/pathToSuite/event.php`, which will then be stored. There can be different event handlers implemented to deal with how the event will be stored. There are three types of events that can be send. All events are communicated by sending a POST message with the following parameters added to the above mentioned URL.

Parameter	Description
event	Should be set to either `start`, `data` or `end`
domain	Should be set to `experiment`
source	Should be set to source of the event, usually the identifier of the experiment.

NOTE: For processing/java-driven experiments a helper class was created to interface with this event handling. The class can be found in the folders of those experiments and in the experiments root folder under either ExperimentLogger.pde or ExperimentLogger.java.

3.1 ExperimentEventHandler

At the moment of writing there is one event handler implemented called the `ExperimentEventHandler`. This handler stores events in both a database, as described in section 1.4, and in a file[6]. The instantiation of this handler can be found and altered in `event.php`.

3.2 `start` events

The start event indicates the beginning of an experiment instance. An instance is a combination of an experiment and a participant at a certain time. A participant can have multiple instances with the same experiment, by doing the experiment multiple times. The following parameters can be communicated in the body part of the POST message:

Parameter	Description
participant_id	The internal identifier for the participant that is used inside the experiment, which could be different from the user id in the database of this system.
params	Any parameters that are necessary to identify the variation of the experiment in order to reproduce it.

The POST request will return the experiment instance identifier as it is stored.

3.3 `data` events

The data event indicates interaction of some sort within the experiment instance. The following parameters can be communicated in the body part of the POST message:

Parameter	Description
participant_id	The internal identifier for the participant that is used inside the experiment, which could be different from the user id in the database of this system.
data	Any data that you would like to store.

[6] At the moment of writing this was `data.dat`

3.4 `stop` events

The data event indicates the end of the experiment instance. The following parameters must be communicated in the body part of the POST message:

Parameter	Description
id	The experiment instance identifier that was returned after the start event

4 Database

The database can be accessed online via the link `https://www.science.uva.nl/phpmyadmin/`, log in with the username and password on domain `dbm.science.uva.nl`.

5 Analytics Extension

On top of the normal suite functionality, an analytics tool was added. This tool was intended to support students and others in retrieving data from various experiments. There are three main component that need to be selected in order to use this tool to get data out of the database.

Source The source from where you want to retrieve the data. The only source supported so far is "WEX", which retrieves data from the same database as used by the suite event handling.

Interpreter The entries in the suite data table contain raw data that is formatted differently for different experiments. The interpreter processes that raw data and converts it to a data matrix. There are currently two interpreters implemented identified by the names "AL" and "SPR" that interpret data from AlienGame and SelfPacedReading experiments respectively.

View The data matrix that comes out of the interpreter is than transformed in a data output format by a view. Currently only exports to "HTML" and "CSV" are supported.

The analytics tool can be used by retrieving the URL `http://domain/pathToSuite/analytics` with the following parameters:

Parameter	Description
experiment	The experiment from where the data came from, identified by the standard experiment identifier under which the data events were also stored.
source	Should be set to `WEX`
interpreter	Should be set to `AL` or `SPR`
view	Should be set to `HTML` or `CSV`

Example: `http://domain/pathToSuite/analytics/?experiment=ALExp1&source=WEX&interpreter=AL&view=CSV`

6 Folders and files

6.1 /

6.1.1 /database_handlers.php

File containing the class that handles communication with the database. This is also where the database account credentials are set.

6.1.2 /data.dat

At the time of writing this was the file that was written by the ExperimentEventHandler event handler that logs to both a database and a text file. This is in principle not necessary anymore, but was kept as backup.

6.1.3 /event_handlers.php

File containing the ExperimentEventHandler class. It is intended that you add new event handlers to this file if necessary.

6.1.4 /event.php

Entry point for events. Also the place where event handlers are registered.

6.1.5 /index.php

The index file is responsible for most of the redirecting logic as it decides which content to load. It deals with the loading of doors, handles stepping through the stages and makes sure the database connection is present. The index file is the main entry point for all interactions, except for the storage of events[7]. See the comments in the index.php

6.1.6 /stat.php

This is a file that is accessed via javascript (XMLHTTPRequest) from most experiments' begin scripts to store the browser capabilities, such as javascript and java support, of a user in the database. The reason why this script is necessary is because these capabilities are not directly observable from the initial client's request, therefore a small javascript script is included in the `begin.php` script of an exercise that, if it is executed, reports back via this `stat.php` file.

6.1.7 /user.php

In this file the operations on the session, like login and logout, are implemented.

6.2 /analytics

The analytics component of the suite was described in section 5. The files required for that component are placed in this folder. The `analytics/index.php` file loads the right source, interpreter and view objects based on the parameters in the url, that are defined in `analytics/source.php`, `analytics/interpreter.php` and `analytics/view.php` respectively.

6.3 /doors

In this folder all doors are defined in their own separate script files. Doors are explained in section 1.3. Previously used doors were `doors/nemo.php` and `doors/mt.php` to deal with participants coming from NEMO and Mechanical Turk respectively.

6.4 /experiments

All experiments are located in their own subdirectory within the experiments folder. The name of that subdirectory must[8] be equal to the experiment identifier that will be used throughout the system. The experiment identifier can be any name that does not contain the characters '..' or '/'. Each subdirectory (i.e. each experiment) can, besides experiment specific data, also contain specific *phase views*. These views are included by the loader when a specific phase is requested. If you want to use the default loader you need to stick to the default names. The *phase views* are briefly described in the following subsections. When a certain view is not present in the subdirectory, the default loader will automatically move forward to the next view (e.g.: When the view for phase *pre* is missing, the loader will attempt to load the view for phase *game*).

6.4.1 /experiments/EXPERIMENT/begin.php

The *begin.php* file contains the view for the *preview* and *begin* phase. The preview phase indicates that an impression of the experiment should be given, but no further action must be possible. The begin phase differs from the preview phase in that it should contain a link to the next phase. Typically this view is useful for notifying the participant of certain (technical) prerequisites before they start with the experiment. If you see no purpose for this phase, simply don't include this file in the experiment's directory. In this phase the following PHP variables are available:

[7] In that case the file event.php is the entry point

[8] This is a requirement if you want to use the default loading system. If you want to overrule that system see section 6.1.5 on how to add exceptions to the loading system.

Variable	Description
$_GET['phase']	The original requested phase, which in this situation can either contain `preview` or `begin`.
$_SESSION['language']	The language of the user. Typically only Dutch (nl) and English (en) are supported.
$PATHS_BASE_URL	The url pointing to the main folder of this suite.
$PATHS_EXPERIMENT_URL	The url pointing to the main folder of the loaded experiment
$PATHS_GAME_URL	The url pointing to the next phase in the experiment (*deprecated*)
$PATHS_NEXT_PHASE_URL	The url pointing to the next phase in the experiment

6.4.2 /experiments/EXPERIMENT/pre.php

The *pre.php* file contains the view for the *pre* phase. The pre phase is suited for any interaction you would like your participants to have before starting the actual experiment. Typically this view is useful for showing a questionair prior to the experiment. Advised usage would be to also handle the posted questionair within the view itself so that afterwards you can send the participant to the next phase simply by setting the variable $_GET['phase'] to 'game'. If you see no purpose for this phase, simply don't include this file in the experiment's directory. In this phase the following PHP variables are available:

Variable	Description
$_SESSION['language']	The language of the user. Typically only Dutch (nl) and English (en) are supported.
$PATHS_BASE_URL	The url pointing to the main folder of this suite.
$PATHS_FORM_SUBMIT_URL	The url pointing to the current view.

6.4.3 /experiments/EXPERIMENT/experiment.php

The *experiment.php* file contains the view for the *game* phase. The game phase is meant for the actual experiment. The default load will ensure that additional parameters given in the URL are kept during the consecutive phases, which means that the view for this phase can use these parameters where necessary. There is a generic experiment view available for experiments implemented with processing java applets. In this phase the following PHP variables are available:

Variable	Description
$_SESSION['language']	The language of the user. Typically only Dutch (nl) and English (en) are supported.
$PATHS_BASE_URL	The url pointing to the main folder of this suite.
$PATHS_EVENT_URL	The url pointing to event file of this suite to which the experiment can send its events.
$PATHS_COMPLETE_URL	The url pointing to the location where the participant must be sent after finishing the experiment.
$PATHS_EXPERIMENT_URL	The url pointing to the main folder of the loaded experiment

> **NOTE:** A template for this file for java applet experiments can be found in `experiments/template_applet_experiment.php`

6.4.4 /experiments/EXPERIMENT/post.php

The *post.php* file contains the view for the *post* phase. The post phase is highly similar to the pre phase, with the only two differences being that it is loaded after the participant has completed the experiment and that the $_GET['phase'] variable should be set to "end" after the questionair was handled.