Dokumentation IBF-Tool

# Vorwort

Diese Dokumentation soll eine Hilfestellung bei der Weiterentwicklung oder Wartung des Tools geben, sowie die grundlegende Funktionsweise des Programms verdeutlichen.

# Installation und Setup

Das Programm ist mithilfe des Zend Frameworks komplett in PHP, HTML und JavaScript geschrieben. Es verwendet teilweise jQuery für Clientseitige Berechnungen und dynamische Events in Experimenten.

## Voraussetzungen

Es braucht relativ wenig um das Tool lauffähig zu kriegen.

* Apache oder LighTTPD Server
* MySQL Datenbank

### Apache-Konfiguration

Folgende Schritte sind nötig um das Tool auf Apache lauffähig zu machen:

* Aktivierung von mod\_rewrite
* Editieren der httpd\_conf:

/ibftool "<PATHOFSERVER>/ibftool/public"

<Directory "<PATHOFSERVER>/ibftool/public">

AllowOverride All

Order Deny,Allow

Allow from all

</Directory>

Natürlich lässt sich das Tool auch mittels vhost konfigurieren.

### LighTTPD-Konfiguration

Folgende Schritte sind nötig um das Tool auf LighTTPD lauffähig zu machen:

* Aktivierung von mod\_redirect und mod\_setenv
* Folgende Linie ans Ende der lighttpd.conf hinzufügen:

$HTTP["host"] == "ibftool" { server.document-root = "PATH\_TO\_IBFTool/public" setenv.add-environment = ( "APPLICATION\_ENV" => "development" ) url.rewrite-once = ( ".\*\.(js|ico|gif|jpg|png|css|html?|eot|docx?|xlsx?|svg).\*$" => "$0", "^/.\*(\?.\*)" => "/index.php$1", "" => "/index.php" ) }

Der Hostname sollte natürlich entsprechend konfiguriert werden (IP oder fixer Hostnamen mit Umleitung in der hosts-Datei.

## Datenbankkonfiguration

Die Konfiguration der Datenbank wird in der Datei unter application/config/db\_config.ini vorgenommen. Je nach Environment lassen sich unterschiedliche Parameter eintragen.

Das ganze DB-Modell wurde mittels der MySQL Workbench erstellt und benutzt InnoDB als relationales Schema, ist zudem komplett UTF-8 (genauso wie der Rest der Software).

# Allgemeine Funktionsweise

Das ganze Tool hat einen zusammenhängenden Aufbau. Experimente sind grundsätzlich als Treatments anzusehen. Jedes Treatment kann dabei aus einer Anzahl von verschiedenen Modulen bestehen.

Diese Module entsprechen den jeweiligen Controllerklassen im Code (Modul Questionnaire 🡪 QuestionnaireController).

Beim Start eines Treatments werden die dazugehörigen XML-Configs aus der Datenbank geladen (Tabelle ibftool\_treatments\_has\_module Spalte Config) und die Module konfiguriert. Jedes Modul hat dabei seine eigene Konfiguration, in welcher z.B. die Seiten-ID (Static-Module für statische Seiten) bzw. die ID der Fragebogen-Seiten eingetragen werden (Questionnaire). Wiederum andere Module erhalten dadurch nur Konfigurationsparameter (Investmentgame) zu Anfangsvermögen oder der Anzahl Runden.

Jedem Treatment können User zugeordnet werden. Diese können sowohl händisch in der DB erfasst werden, es gibt aber auch einen Registrierungsmechanismus mit E-Mail Validierung für den Betrieb im Web (RegistrationController).

# MVC

Das gesamte Projekt implementiert das Model-View-Controller Pattern. Logik und GUI sind somit strikt getrennt. Models sind eng an die Datenbank geknüpft, Controller verknüpfen die Views (in HTML) mit den Models und sorgen für die Applikationslogik.

Modulbeschrieb

# Questionnaire

Das Questionnaire-Modul wird verwendet um Fragebögen zu erstellen. Es gibt diverse verschiedene Arten von erstellbaren Fragen (SingleChoice, MultipleChoice, Experimentspezifische Fragen wie „RS“- … ).

Unter „library/ibftool/Form/Element“ lassen sich alle dazugehörigen Zend\_Form\_Element-Klassen finden.

Jeder Fragentyp (Spalte „typ“ in der DB-Tabelle questionnaire\_question) wird mittels Magicmethod instanziert. Jede Frage ist somit ein *Zend\_Form\_Element* und implementiert das *Ibftool\_Form\_Element\_Interface\_Questionnaire.*

## Neuer Fragetyp erstellen

Durch das anlegen einer neuen Klasse im Namensschema „ibftool\_Form\_Element\_<NAME>“ lassen sich neue Typen von Fragen anlegen. Die Fragen müssen die public Methode setQuestion($question) implementieren, vorgegeben vom dazugehörigen Interface (Ibftool\_Form\_Element\_Interface\_Questionnaire*).* Jede Frage wird dazu zuerst von ihren Decoratoren gestrippt, danach mit den notwendigen Decoratoren ([Pattern](http://en.wikipedia.org/wiki/Decorator_pattern)) ausgestattet und mittels den Informationen aus dem $question-Objekt gefüllt. Die Instanzierung etc. erfolgt automatisch in der Model-Klasse „Questionnaire\_Question“

## Antworten zu Fragen erfassen

Dies ist leider noch nicht per Backendmodul möglich. Jede Antwort wird mittels Fremdschlüssel einer konkreten Frage zugewiesen, d.h. wäre eine Implementation dieser Funktionalität ins GUI ohne Probleme möglich.

# RS

Module welche im Zusammenhang mit den Experimenten von Remo Stössel, Mitte 2012 verwendet wurden. Dabei wurde [Highcharts](http://www.highcharts.com/) verwendet um dynamisch Javascript basierte Charts zu zeichen.

### RS-Plot

Konfiguration referenziert die Ausgangswerde & die entsprechenden Zielspalten in der Tabelle (d.h. werden Questionnaire-Questions benutzt im die Werte weiter zu ziehen).

<config> <ids> <id>75/147</id> <id>120/148</id> <id>134/149</id> <id>166/168</id> </ids></config>

# InvestmentGame

Das Investmentgame ist ein rundenbasiertes Spiel welches es Spielern ermöglicht, Geld in jeder Runde zu investieren.

Konfigurationssturktur:

<config>

<rounds>20</rounds>

<money>1</money>

<groups>

<group1>

<rate>0.2</rate>

<rate>0.2</rate>

<rate>0.5</rate>

<rate>0.2</rate>

</group1>

</groups>

</config>

## MLA-Experiment

Sehr ähnlich zum Investmentgame, jedoch mit 2 Gruppen welche den Erfolg jeweils nach einer unterschiedlichen Anzahl gespielter Runden angezeigt bekommen.

<config> <money>0.4</money> <rounds>36</rounds> <part>2</part> <cycle\_length>3</cycle\_length> <high\_prob\_rate>250</high\_prob\_rate> <low\_prob\_rate>-100</low\_prob\_rate></config>

Money = Startgeld  
Rounds = zu spielende Runden  
Part = Es gab mehrere „Parts“, dient zur Identifikation welcher Part es ist  
cycle\_length = Anzahl Runden bis Erfolgsanzeige  
high\_prob\_rate = 250% Rendite mit hoher Wahrscheinlichkeit  
low\_prob\_rate = -100% Rendite mit kleiner Wahrscheinlichkeit