MiNSoft-System
Systemkonzept
Version 1.1 vom 11.1.2003

Autoren:
Mario Boller-Olfert
Eleonore Olfert

Inhaltsverzeichnis

1. Allgemeines

1.1. Zweck des Systems

Das MiNSoft-System wurde eingeführt für die Entwicklung von Software-Systemen für Virtuelle Mikronationen im World Wide Web. Es kann jedoch auch für andere Systeme eingesetzt werden.

Das MiNSoft-System basiert auf einem Multi-Schichten-Modell in einer verteilten Web-Umgebung.

1.2. Dokumentgestaltung

Für alle Dokumente ist zur Erstellung Word 2000 zu verwenden. Die Dokumentvorlage MiNSoft.dot ist für alle Dokumente zu verwenden.

1.3. Sprache

Die Sprache aller Dokumente ist deutsch. Die Sprache in der Implementation ist englisch.

1.4. Dokumentations-Vereinbarungen

<name>	In spitze Klammern gesetzte Namen sind Stellvertreter für konkrete Werte.
{item1,item2}	In geschweifte Klammern gesetzte Werte sind optional.
[item1, item2]	In eckige Klammern gesetzte Werte sind Alternativen. Es muss genau eine verwendet werden
// Kommentar	Hinter doppelten Schrägstrichen stehende Werte sind Kommentare.

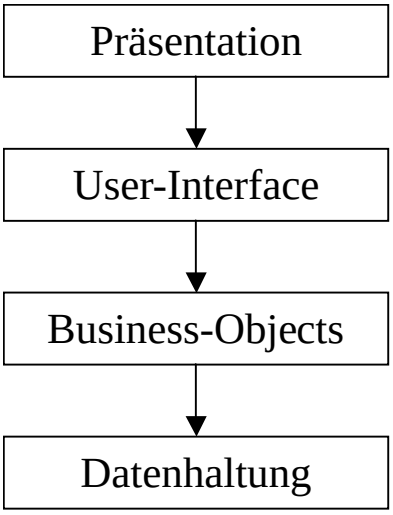
1.5. Offene Punkte

In Zukunft muss ein Versions-Verwaltungs-System eingeführt werden. Die Verwendung von CVS ist zu prüfen, um eine Versionsverwaltung im Web zu ermöglichen.

2. Systemkonzept

2.1. Allgemines Konzept

Das MinSoft-System verfolgt strikt ein Multi-Schichten-Konzept (englisch: Multi-Tier).



In der **Präsentation** wird die Darstellung der Daten behandelt. Hier können Daten auch sprach- und kulturabhängig dargestellt werden. Auch die Darstellung abhängig vom einem Design des Kunden kann hier gehandhabt werden. In MiNSoft ist diese Schicht ein HTML-Formular, dass in einem WebBrowser dargestellt wird.

Das **User-Interface** übernimmt die Daten von der Präsentation und steuert den Dialog-Ablauf. Es kommuniziert mit den Business-Objects zur Erfüllung der angeforderten Aufgaben. In MiNSoft ist dies ein PHP-Skript. Der Ablauf wird sowohl unter einem Apache-Webserver als auch unter dem Microsoft Internet Indformation Server (IIS) 5.0 getestet. Als Kommunikationsmittel zu den Business-Objects werden die ausgelieferten Module XMLRPC verwenden.

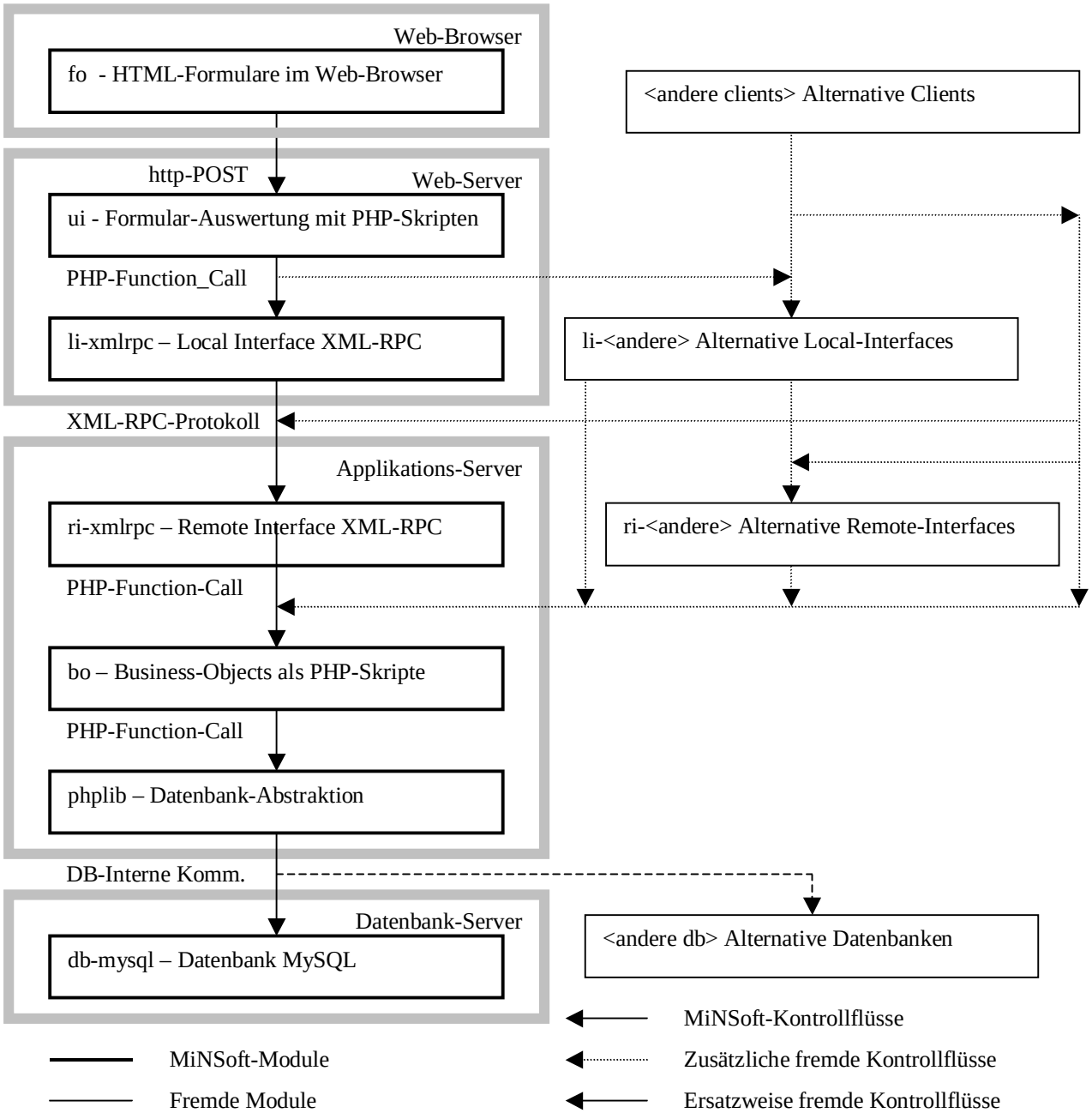
Die **Business-Objekte** implementieren die aus den Use-Cases folgenden Geschäftsabläufe. Hier werden die eigentlichen Anwendungsfunktionalitäten implementiert. In MinSoft wird dies ebenfalls als PHP-Skript implementiert, um weitgehend maschinenunabhängig zu sein.

Die **Datenhaltung** speichert persistente (dauerhafte) Daten über den einzeln Geschäftsvorfall hinaus. In der Regel wird hier eine Relationale Datenbank verwendet. In den von MiNSoft gelieferten Modulen wird MySQL verwendet, aber andere Datenbank sind problemlos möglich.

Jede Ebene des Systems kann auf verschiedenen Rechnern ablaufen, hat genau definierte Schnittstellen und ist einzeln konfigurierbar und austauschbar.

Obwohl MiNSoft damit ein sehr flexibles Konzept bietet, wird im folgenden ausschließlich das Konzept der MiNSoft-Module beschrieben. Die Implementation fremder Module und Kontrollflüsse ist nicht Teil der von MiNSoft angebotenen beziehungsweise unterstützten Systeme. Diese werden von MiNSoft gerne als Auftrags-Projekte erstellt.

2.2. Detaillierte Schichten-Grafik



Die Pfeile symbolisieren den Prozessablauf in MiNSoft. Eine oben stehende Komponente ruft jeweils untergeordnete Komponente auf und kann dabei Daten zu ihr übertragen und wieder zurück bekommen.

2.3. Schichten-Beschreibung

2.3.1. fo – HTML-Formulare im Web-Browser

Die Oberfläche wird als HTML-Formulare gespeichert, die auch in mehreren Sprachvarianten vorliegen können. Die Formulare sind mit Hilfe von PHP-Variablen für variablen Teile parametrisiert.

2.3.2. ui – Formular-Auswertung mit PHP-Skripten

Die Steuerung der Formularfolge, die Ausgabe von Werten und die Verarbeitung erfasster Werte wird mit Hilfe von PHP-Skripten bewerkstelligt.

Es wird das PHP-Session-Konzept zur Verwaltung der Authentifizierung und der Daten mehrerer voneinander abhängiger Formulare verwendet.

Alternativ sind Clients denkbar, die unterschiedliche Schnittsellen verwenden. Dies reicht von einem Interface zu einem Local-Interface übder Aufruf über u.a. XML-RPC auf ein Remote-Interface bis zu einem direkten Aufruf der Business-Objekte. Zum Beispiel könnte man ein Visual-Bais-Modul mit User-Interface schreiben, das über XML-RPC zugreift.

2.3.3. li-xmlrpc – Local-Interface mit XML-RPC

Das Local-Interface wird von der Benutzeroberfläche (hier von ui-Modulen per Funktionsaufruf aufgerufen und sendet über XML-RPC einen Remote-Funktionsaufruf an das Remote-Interface.

Alternativ ist auch ein Local-Interface denkbar, das direkt die Buesiness-Objekte aufruft.

2.3.4. ri-xmlrpc – Remote-Interface mit XML-RPC

Das Remote-Interface wird von der Benutzeroberfläche (hier von ui-Modulen) über das XML-RPC-Protokoll aufgerufen und ruft seinerseits die Business-Objekte auf.

2.3.5. bo – Business-Objects

Die Business-Objects implementieren die Use-Cases der Geschäftsprozesse. Sie werden als PHP-Skripte implementiert. Hier werden auch alle Prüfungen auf Authentifizierung und unzulässige Parameterwerte durchgeführt. Dann wird auf Daten aus der Datenbank über die Datenbank-Abstraktionsebene zur Implementation der Funktionalität zugegriffen.

2.3.6. phplib – Datenbank-Abstraktion

Aus der PHP Base Library wird ausschliesslich das Datenbank-Abstraktions-Modell verwendet, da das Session-Management über die Standard-PHP-Modell verwendet wird. Zur Zeit steht nur das Modul für MySQL-Datenbanken optimiert zur Vefügung, jedoch können auch die anderen Module aus der phplin verwendet werden.

2.3.7. db_mysql – Datenbank

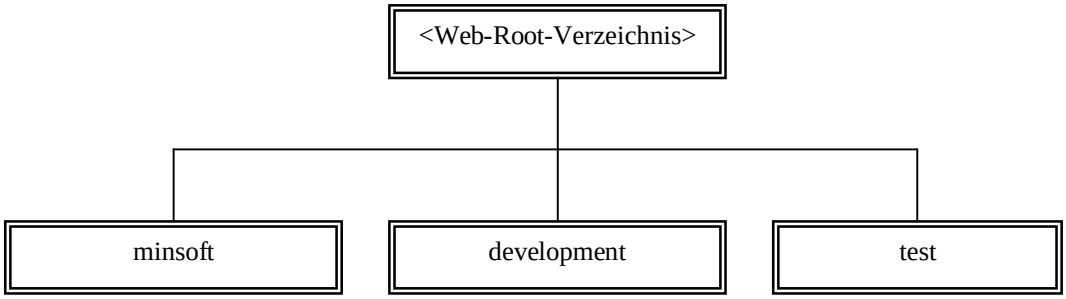
Als Datenbank wird für die aktuelle Implementation MySQL eingesetzt.

Es ist möglich andere Relationale Datenbanken einzusetzen, soweit für sie ein php-lib-Modul für Datenbankzugriff existiert.

3. Projektstruktur

3.1. Gesamt-Verzeichnisstruktur

Alle Daten von MiNSoft werden in eine gemeinsamen Projekt-Struktur abgelegt. Diese Struktur existiert in 3 verschiedenen Ausprägungen mit jeweils der gleichen Unterstruktur.

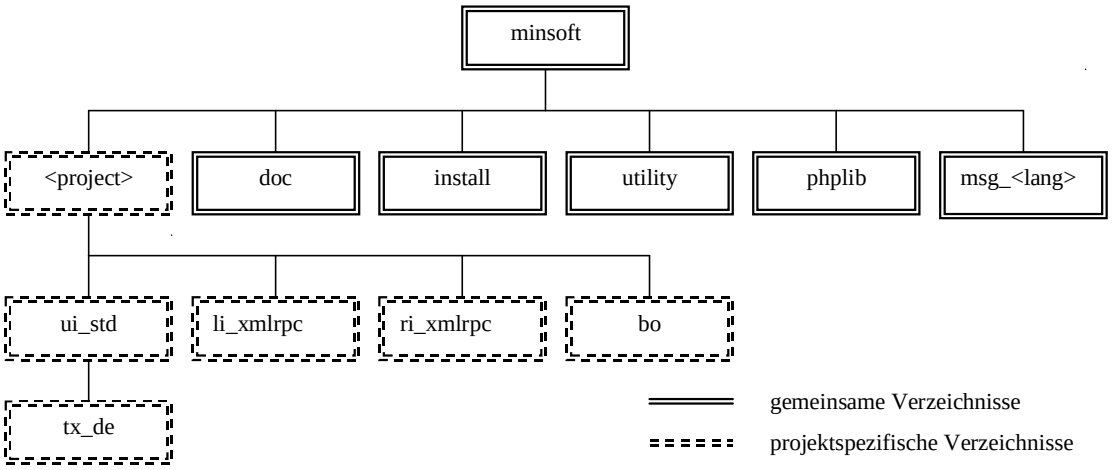


Die Verzeichnisse haben folgende Bedeutung:

Name	Bedeutung
Web-Root-Verzeichnis	„/“-Verzeichnis des Webservers
minsoft	Produktionsversion
development	Entwicklungsstand
test	Qualitätssicherungs-Stand

3.2. Verzeichnisstruktur eines Hauptzweiges

Die gesamte Verzeichnisstruktur unter einem dieser Zweige ist hier für Verzeichnis minsoft dargestellt:

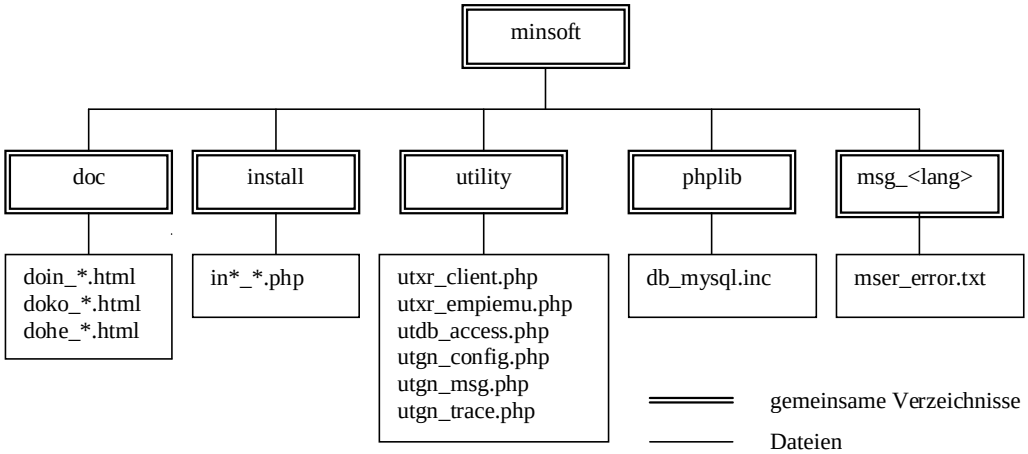


Die Verzeichnisse haben folgende Bedeutung

Name	Bedeutung
minsoft	Produktionsverzeichnis
<project>	Unterverzeichnis eines speziellen Projektes, z.B. „bank“ für das Bankensystem.
ui_std	Benutzungs-Oberfläche in der Standard-Version. Weitere „ui_*“-Verzeichnisse wären möglich.
tx_de	Verzeichnis mit HTML-Formularen und Oberflächentexten in deutsch. Weitere Verzeichnisse, z.B. für englisch wären möglich.
li_xmlrpc	Local-Interface-Dateien
ri_xmlrpc	Remote-Interface-Dateien
bo	Business-Objekte
doc	Dokumentation mit Konzepten, Installations-Anweisungen und Hilfetexten
install	Installations-Skripte in PHP
utility	Hilfsprogramme für diverse allgemeine Funktionalität.
phplib	Verzeichnis mit Dateien für das Datenbank-Interface
msg_<lang>	Sprachspezifisches Verzeichnis mit Dateien mit Message-Texten.

3.3. Allgemeine Dateien

Unter den gemeinsamen Verzeichnissen liegen die folgenden Dateien.



Die Dateien haben folgende Bedeutung

Name	Bedeutung
doin_*.html	Installationsanleitung
doko_*.html	Konzept-Papier
dohe_*.html	Hilfetexte
ib*_*.php	Installations-Skript
utxr_client.php	XML-RPC Client-Hilfsprogramme
utxr_epiemu.php	XML-RPC-EPI-Emulation
utdb_access.php	Datenbank-Zugriffs-Hilfsmittel
utgn_config.php	Allgemeine Konfigurationsdatei
utgn_msg.php	Message-Hilfsfunktionen
utgn_trace.php	Tracing-Hilfsfunktionen
db_mysql.inc	Allgemeines Datenbank-Interface für MySQL
mser_error.txt	Fehlertexte

3.3.1. Datei utgn_config.php

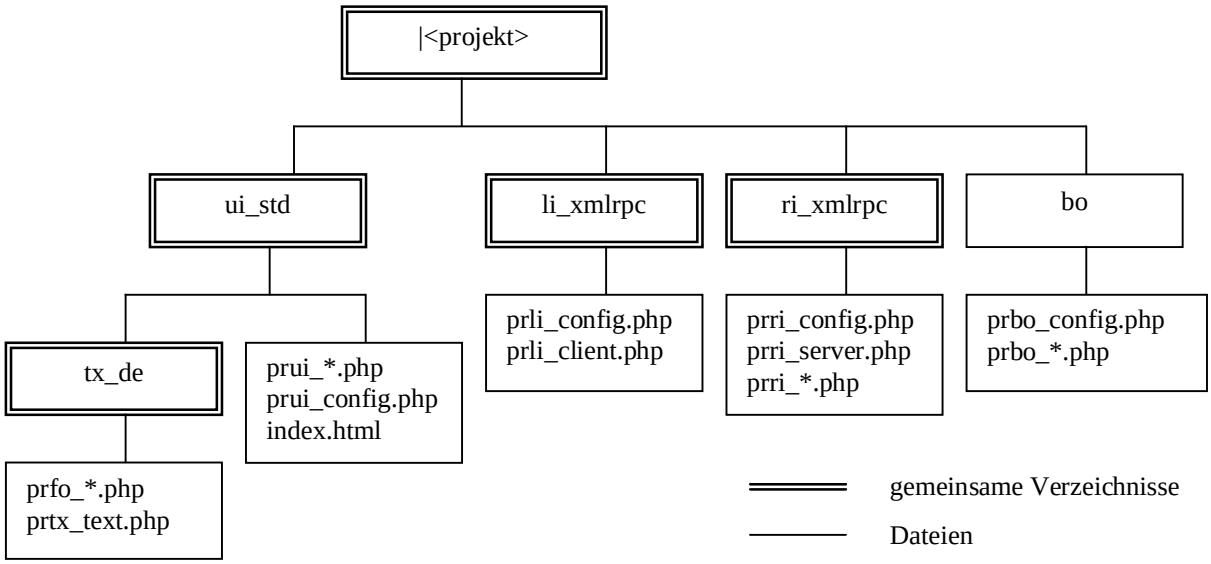
In der allgemeinen Konfigurationsdatei stehen die zum Zugriff auf die Datenbank notwendigen Parameter.

```
$utcv_phplib_path = $_SERVER["DOCUMENT_ROOT"]."/minsoft/phplib";
$utcv_db_system = "db_mysql.inc";
$utcv_db_host = "localhost";
$utcv_db_database = "DB17373";
$utcv_db_user = "techuser";
$utcv_db_password = "test";
$utcv_tracepath = '\tmp\trace.txt';
```

Name	Bedeutung
$utcv_phplib_path	Pfad ddes Verzeichnisses mit den Datenbank-Interface-Modulen
$utcv_db_system	Dateiname des Interfaces für das verwendete Datenbanksystem
$utcv_db_host	Netzwerkname des Datenbank-Hosts
$utcv_db_database	Name der Datenbank, die die Tabellen für das MiNSoft-System enthält
$utcv_db_user	Technischer Benutzername zum Zugriff auf die Datenbank
$utcv_db_password	Passwort des technischen Benutzers zum Zugriff auf die Datenbank
$utcv_tracepath	Pfad und Dateiname für die Ausgabe von Traces

3.4. Projektspezifische Dateien

Alle Projektspezifischen Dateien beginnen mit zwei Stellen Projektkennung. Im unten dargestellten Verzeichnisbaum wurde „pr“ für das Projekt eingesetzt.



Name	Bedeutung
prfo_*.php	HTML-Formulare als php-display-Funktion
prtx_text.php	Texte für die Oberfläche
prui_*.php	User-Interface-Module
prui_config.php	User-Interface-Konfiguration
index.htm	Hauptseite des Projekts
prli_config.php	Local-Interface-Konfiguration
prli_client.php	Local-Interface-Client-Modul
prri_config.php	Remote-Interface-Konfiguration
prri_server.php	Remote-Interface-Server-Modul
prri_*.php	Zusätzliche Remote Interface-Module
prbo_config.php	Konfiguration der Business-Objekte

prbo_*.php	Module der Business-Objekte

3.4.1. Datei prtx_text.php

Die Datei enthält ein PHP-Skript mit Definitionen für Bezeichnungen von Texten, Zuweisungen von Texten an das Array <projectid>tx_text und die Zugriffs-Funktion <projectid>tx_getText.

Beispiel für Datei „batx_text.php“:

```
<?php
define(„batx_title1“, 1);
define(„batx_labelName“, 1234);
batx_text[1] = „Hauptmenü“;
batx_text[1234] = „Benutzername“;

function batx_getText($textId)
{
   return batx_text[textId];
}
?>
```

3.4.2. Dateien prfo_<formular_name>

Die Datei enthält ein PHP-Skript mit der Definition eines Formulars zusammen mit globalen Variablen für Werte in diesem Formular. Pflichtwert ist der Pfad der PHP-Skripte „$<projectid>fv_script_path“. Alle Formulare sollen in XHTML geschrieben sein.

Beispiel für Datei bafo_login:

```
<?php
global $bafv_script_path;
global $bafv_name;
global $bafv_password;

bafo_displayForm()
{
echo <<<END
<html>
<head><title>Login</title></title>
<body>
<h1>Login</h1>
<form action=”$bafv_script_path/baui_login.php” method=”post”>
<input type=”text” name=”bafv_name” value=”$bafv_name” /><br />
<input type=”text” name=”bafv_password” value=”$bafv_name” /><br />
<input type=”submit”>
</form>
</head>
</html>
END
;
?>
```

Im Beispiel wurde aus Gründen der Übersichtlichkeit auf DOCTYPE und alle Header-Definitionen verzichtet. Diese müssen natürlich in einem realen Fall vorhanden sein.

3.4.3. Datei prui_config.php

In der User-Interface-Konfiguration steht neben projektspezifischen Einträgen der Pfad der Utilities:

$utility_path=$_SERVER['DOCUMENT_ROOT'].'/minsoft/utility';

3.4.4. Datei prli_config.php

In der Konfigurationsdatei für das Local-Interface steht neben dem Pfad der Utilities die nötigen Adressdaten zum Ansprechen des XML-RPC-Remote-Interfaces. („pr“ steht für ein bestimmtes Projektkürzel).

```
$utility_path=$_SERVER['DOCUMENT_ROOT'].'/minsoft/utility';
$xmlrpc_host = 'www.ymir.de';
$xmlrpc_path = '/minsoft/<project>/ri_xmlrpc/prri_server.php';
$xmlrpc_port = '80';
```

Name	Bedeutung
$utility_path	Pfad zu den Utilities
$xmlrpc_host	Host-Name des Webservers, auf dem das Remote-Interface abgelegt ist
$xmlrpc_path	URI-Pfad zum Dokument, das den Server initiiert
$xmlrpc_port	Port-Adresse des Web-Servers für HTTP-Protokoll

3.4.5. Datei prbo_config.php

In der Konfigurationsdateien für das Business-Objekt steht der Pfad der Utilities.

```
$utility_path=$_SERVER['DOCUMENT_ROOT'].'/minsoft/utility';
$db_prefix=’pr’;
```

Name	Bedeutung
$utility_path	Pfad zu den Utilities
$db_prefix	Namenspräfix, der vor alle Tabellennamen des Projekts gesetzt wird.

3.5. Namenskürzel

Alle Namen beginnen mit der zweistelligen Projekt-Identifikation. Zur Zeit gibt es folgende Projektidentifikationen:

Projectid	Bedeutung
ut	Hilfsprogramme
ba	Bankensystem
is	Introspection
re	Rechner-Beispiel

Auf die Projekt-Identifiakation folgen zwei Buchstaben, die den Namensraum bezeichnen. Es gibt folgende Namensräume:

Namensraum	Bedeutung	Objekttypen
in	Initialisierungs-Datei	Datei, Methode
ui	User-Interface-Datei	Datei, Methode
ri	Remote-Interface	Datei, Methode
bo	Business-Object	Datei,Methode
db	Datenbank-Datei	Datei, Methode
fo	Formular-Datei	Datei, Methode
fv	Formular-Variable	Variable
tx	Text	Id, Array, Methode
ms	Message	Id, Array, Methode
xr	XML-RPC	

Darauf folgt ein Unterstrich und ein frei vergebbarer Name.

Beispiele:

babo_login.php Bankensystem-Login-Programm

bafo_login.php Bankensystem-Login-Formular

isbo_login.php Introspection-Login-Programm

bafv_userName Formularvariable für Benutzername

4. Hilfsprogramme

Die Hilfsprogramme liegen im Verzeichnis „utility“. Zu ihrer Verwendung muss die Config-Datei utgn_config.php angepasst werden.

4.1. utgn_config.php – Hilfsprogramm-Konfiguration

Die Konfigurationsdatei kann zur Zeit folgende Variablen enthalten:

Variable	Bedeutung
$utcv_phplib_path	Pfad zum PHP-LIB-Verzeichnis
$utcv_db_host	Datenbank-Hostadresse
$utcv_db_database	Datenbank-Name
$utcv_db_user	Benutzername für Datenbank-Zugriff
$utcv_db_password	Passwort für Datenbank-Zugriff
$utcv_trace_path	Pfad der Trace-Datei

4.2. utxr_client.php – Hilfsmodul für XML-RPC-Aufrufe

Dieses Modul bietet Funktionen zur Vereinfachung des Aufrufs über XML-RPC-Schnittstelle an.

Das Modul enthält die Definition der Klasse UTXRClient. Dieses muss zur Verwendung der Methoden instantiiert werden.

Es gibt folgende Methoden:

Methode	Bedeutung
UTXRClient()	Initialisierung der Klasse
isFault($response)	Prüfen, ob eine PHP-Struktur einen XML-RPC-Fault enthält.
setServer(host,path,port)	Setzen von XML-RPC-Server-Host, -Pfad und http-Port
call(method,name)	Aufruf einer Methode mit Methodennname und Parametern als PHP-Struktur. Es wird eine XML-Struktur generiert und unter Verwendung der mit setServer gesetzten Werte eine http-Verbindung zum Server aufgebaut. Zurückgegeben wird das Ergebnis der Ausführung als PHP-Variable oder eine Struktur mit einer Fehlerbeschreibung. Der Rückgabewert kann mit der Methode isFault auf Fehlerrückgabe geprüft werden.

4.3. utxr_epiemu.php – XML-RPC-epi-Emulation

Dieses PHP-Skript emuliert die XML-RPC-epi-Library für den Fall, dass diese Extension nicht installiert werden kann. Die Schnittstelle emuliert alle von MiNSoft benötigte Funktionalität. Die Schnittselle ist beschrieben unter http://xmlrpc-epi.sourceforge.net/main.php?t=php_api

Für ihre Verwendung ist die xml-Extension in PHP mit der expat-Library Voraussetzung.

Wenn irgend möglich ist aus Performanz- und Stabilitäts-Gründen die Verwendung der Original-Library vorzuziehen.

4.4. utdb_access.php - Datenbankzugriffs-Klasse

Das Modul enthält die Definition der Klasse DBAccess, die von der Klasse DB_Sql der phplib abgeleitet ist. Die Klasse legt die Zugriffs-Parameter der Datenbank fest und erbt alle Methoden der Klasse aus der phplib.

Anwendungs-Beispiel:

```
<?php
// utility-Pfad einlesen
require(„babo_config.php“);

// Datenbank-Klasse einlesen
require(“$utility_path/utdb_access.php”);

// Datenbank instantiieren
$db = &new DBAccess;

// SQL-Kommando absetzen (auch z.B. CREATE TABLE würde gehen)
$db->query(“SELECT username FROM {$db_prefix}konto”);
// auf Fehler abfragen
if($db->errno != 0)
{
   echo “Database-Error {$db->errno} / {$db->error}: {$db->msg}”;
   exit;
}
// alle records durchgehen
while($db->next_record())
{
   // einen Wert aus dem Record lesen
   echo $db->Record[“username”];
}
?>
```

4.5. utgn_trace.php - Schreiben von Trace-Meldungen

Es wird eine Trace-Funktionalität angeboten. Zu Ihrer Verwendung muss in utgn_config.php die Variable „utcv_tracepath.php“ auf den Dateinamen mit vollständigem Pfad der Trace-Datei gesetzt sein. Achtung: Der Webserver muss Schreiben in auf die Datei in dem Verzeichnis gestatten!

Das Modul enthält nur eine Funktion:

Methode	Bedeutung
trace(place,text)	Beim jedem Aufruf öffnet die Funktion eine Datei für anhängen. Sie schreibt Zeitpunkt, Methodenname und Nachrichtentext in die Datei und schließt die Datei wieder. Diese Funktion sollte aus Performanz-Gründen nur in der Modul-test-Phase verwendet werden.
utgn_dump(var)	Die Variable wird durchlaufen und lesbar ausgegeben. Dies funktioniert auch mit Arrays und Objekten

4.6. utgn_error.php – Behandeln von Fehlern

Es wird ein Array erzeugt, das kompatibel zur XML-RPC-Fehlerbehandlung ist. Der Aufruf erfolgt mit zwei Parametern:

Methode	Bedeutung
getFault(errorId)	Es wird die der Fehlernummer entsprechende Fehlermeldung ermittelt und ein Fault-Array erzeugt.

Die Fehlertexte stehen in der Datei minsoft/msg_de/message.txt. Jede Zeile beginnt mit der Fehlernummer, dann folgt ein Tab-Zeichen und dann der Text der Fehlermeldung. Mit der übergebenen errorId wird auf diese Datei zugegriffen, nach der Fehlenummer gesucht und der dazu gehörige Fehlertext ermittelt.

Es wird ein assoziatives Array erzeugt, mit einem Element „faultCode“ und einem Element „faultString“.

Beispiel:

```
getFault(1001);
```

kann zurückgeben:

```
array(„faultCode => 1001, „faultString“ => “X ist keine Zahl”)
```

Mit dem Array wird xmlrpc_encode_reuqest aufgerufen und man erhält ein gültiges XML-RPC-Fehler-Response.

5. Anforderungen an Fremdsysteme

5.1. Anforderungen an PHP

- Das System ist mit PHP in den Versionen 4.1.2 und 4.2.3 getestet
- Das Session-Management mussaktiviert sein.
- Die mysql-Schnittstelle muss aktiviert sein. Es wurde mit API-Version 3.22.29 getestet.
- Die xml-Extension (expat-Library) muss aktiviert sein.

5.2. Anforderungen an MySQL

- Das System wurde mit MySQL in Version 3.22.32 getestet

Anhang A – XML-Remote Procedure Calls

Das MiNSoft-System benutzt die von php angebotene Schnittstelle für Remote Procedure Calls (XML-RPC). XML-RPC spezifiert das:

1. aufrufen von Funktionen
2. übergeben von Parametern
3. formatieren der Rückgabewerte
4. rückmelden von Fehlern
5. ausgeben von Dokumentation (Introspektion)

Grundsätzlich ist der Aufruf der Funktion im XML Format mit gültigen XML-Tags definiert. Das XML Dokument enthält einen Funktionsnamen und eine Liste von Parametern.

Der Aufruf läuft in den folgenden Schritten ab. Dabei werden die Punkte 2, 3, 7 und 8 durch die Methode „call“ in utxr_client.php abgedeckt.

1. Das UI-PHP-Programm stellt die Parameter für den Call in ein nicht-assoziatives Array
2. Es wird die Funktion xmlrpc_encode_request(method, parameters) aufgerufen. Nun ist der Request XMLRPC-konform als XML-Dokument verpackt.
3. Das PHP-UI-Programm öffnet eine Socket-Verbindung zum Applikations-Server für das HTTP-Protokoll. In der Regel geschieht dies durch Öffnen von Port 80. Das PHP-UI-Programm sendet dann einen HTTP-Header für einen POST-Request mit Content-Type: text/xml. Dann sendet er das XML-Dokument, dass den XMLRPC-MethodCall enthält.

   XML-RPC definiert welche http Header Elemente bei einem http POST übermittelt werden müssen:

   ```
   HTTP Command :  POST <path>/<server>.php HTTP/1.0
   User-Agent :    PHP XMLRPC 1.0
   Host :          <host>
   Content–Type :  text/xml
   Content-Length: <XML-Dokumenten-Länge>
   ```

4. Der Web-Server auf dem Applikations-Server fängt diese Daten auf, analysiert den Header, findet als Adresse hinter dem POST ein Dokument mit Extension .php, ruft darum den PHP-Interpreter mit dem im Header angegebenen Skript auf und übergibt diesem das Dokument auf Standard-Input.
5. Das Server-PHP-Programm erzeugt einen xmlrpc-server mit xmlrpc_server_create, registriert alle ihm bekannten Funktionen bei diesem Server mit xmlrpc_server_register_method und ruft dann xmlrpc_server_call_method mit dem xml-Dokument auf.
6. Der xmlrpc-server wandelt das XMLRPC-Dokument in den Methodennnamen und ein PHP-Array mit den Parametern um, ruft die laut Registrierung dem Namen entsprechende Methode auf. Bei Rückkehr aus der Methode wandelt es den Rückgabewert in eine MethodResponse-XML-Dokument um und gibt es an das Server-PHP-Programm zurück. Diese sendet das Dokument über die noch offene Socket-Verbindung als http-Response an das UI-PHP-Programm zurück.
7. Da auch fälschlicherweise bei Parse-Fehlern ein Dokument mit Format text/html zurückgegeben werden kann (dies lässt sich in PHP nicht ganz vermeiden), muss auf formale Fehler bei der Rückgabe geprüft werden.
8. Das UI-PHP-Programm wandelt mit xmlrpc_decode_request jetzt das Dokument zurück in ein PHP-Array. Wird nur ein einzelner Wert zurück gegeben, dann wird auch nur eine einzelner Wert geliefert.
9. Durch Analye des Rückgabewertes, wird entschieden, ob ein Fehler auftrat.

Anhang B – Datenbank-Schnittstelle von PHPLIB

Es existieren viele verschiedene relationale Datenbanken z.B. MYSQL, POSTGRESQL, Oracle,... auf verschiednen Servern, mit denen die MiNSoft-BOs arbeiten sollen. Daher ist es sinnvoll, eine Zwischenschicht zwischen die Datenbank und MiNSoft-BOa zu legen. In dieser Zwischenschicht werden die verallgemeinerten Datenbankaufrufe für eine spezielle Datenbank aufbereitet.

Die Schnittstelle zwischen MiNSoft-BO und der benutzten Datenbank sind die Datenbankmethoden der PHPLIB. Entsprechend der gewählten Datenbank wird jeweils ein anderes Modul von PHPLIB in die php Module inkludiert. Die Module der PHPLIB wurden für MiNSoft optimiert durch löschen nicht benötigter Funktionen. Zur Zeit existiert so angepasst nur das Modul „db_mysql.php“. Die php Module, die die Business Objekte enthalten haben keinen direkten Zugriff auf die Datenbank, sondern gehen immer über diese Module.

Von der Klasse DB_Sql wird die Klasse DBAccess abgeleitet. In der Konfigurationsdatei utgn_config.php wird angegeben, in welchem Verzeichnis unter der Document Root und in welcher Datei sich die Basisklasse DB_Sql befindet. Zudem werden alle Angaben für ein connect mit der Datenbank festgelegt.

```
$utcv_phplib_path = $_SERVER["DOCUMENT_ROOT"]."/minsoft/phplib";

$utcv_db_system = "db_mysql.inc";

$utcv_db_host = "localhost";

$utcv_db_database = "DB17373";

$utcv_db_user = "techuser";

$utcv_db_password = "test";
```

Die Klasse DBAccess übergibt die Parameter an die Basisklasse und regelt den Umgang mit Fehlermeldungen. Der Benutzer erzeugt eine Instanz der Klasse DBAccess und benutzt dann die Methoden der Klasse DB_Sql um Abfragen an die Datenbank durchzuführen.

Das connect wird nicht explizit aufgerufen, sondern durchgeführt, sobald die erste Abfrage aufgerufen wird und noch keine Verbindung zur Datenbank besteht.

Nach dem Aufruf einer Datenbankabfrage muss jedes Mal die Fehlernummer geprüft werden. Eine Fehlernummer ungleich Null bedeutet, es ist ein Fehler aufgetreten, die Verarbeitung wurde abgebrochen.

War die Datenbankabfrage erfolgreich, so erhält man ein Menge von Datenbank-Records, die in einer Schleife mit jeweils „next_record“ geladen und verarbeitet werden.

Index

Fehler! Keine Indexeinträge gefunden.