A. A

B. B

C. Api-Dokumentation

C.1. Mofa::Controller

Modul zum Zugriff auf Dienste für die Mitfahrzentrale

Beschreibung

Dieses Modul beeinhaltet den Vermittlungsdienst, sowie weitere Funktionen, mit welchen auf den externe Dienste zugegriffen werden kann.

Vermittlung zwischen Mitfahrgelegenheiten und Gesuchen

`@ mapping_lift($)`

Parameter: `Offer`-Objekt

Rückgabewert: sortierte Liste von `Mapped`-Objekten

Findet passende Gesuche in der Datenbank und berechnet die Umwege, die der Anbieter machen müsste, um diese Mitfahrgelegenheiten durchzuführen.

`@ mapping_search($)`

Parameter: `Request`-Objekt

Rückgabewert: sortierte Liste von `Mapped`-Objekten

Findet passende Angebote in der Datenbank und berechnet die Umwege, die die Anbieter machen müssten, um diese Mitfahrgelegenheit durchzuführen.

Geocoding

`$ geocode(@)`

Parameter: Adressbeschreibung (`string,..`)

Rückgabewert: `Geocode`-Objekt

Konstruiert eine Geocode-Anfrage, und vervollständigt die Adresse mit Koordinaten mit Hilfe der Datenbank oder eines externen Geocodingdienstes

Handy-Lokalisierung

`$ get_position($)`

Parameter: Handy-Nummer (`string`)

Rückgabewert: `Area`-Objekt

Fragt das Gebiet, in dem das Handy sich befinden kann, beim Lokalisierungsdienst ab.

SMS-Versand

`$ send_push_sms($$$)`

Parameter: Handy-Nr (`string`), Nachricht (`string`), URL (`string`)

Rückgabewert: `string` mit Antwort des SMS-Servicecenters

Versendet eine WAP-Push-SMS mit der Nachricht und dem Link an die Handynummer.

Entfernungsbestimmung (Routenplanung)

`($$) street_dist($$)`

Parameter: 2 `Point`-Objekte

Rückgabewerte: (Strassenentfernung in Km (`float`), Fahrtzeit in Minuten (`int`)

Die Entfernung wird im Cache nachgeschlagen, falls dies fehlschlägt bei Internet-Services erfragt und falls dies fehlschlaegt wird die Luftlinien-Entfernung zurueckgegeben. Daher ist der Wert fuer Entfernung auf jeden Fall definiert, der Wert fuer Zeit nicht immer.

Kartenbilder

`$ get_map_img($@)`

Parameter: `Point`-Objekt, [Radius (`int`), [Höhe (`int`), Breite (`int`)]]

Rückgabewert: URL zu einem png-Bild einer Karte (`string`)

Fragt die URL zur Karte mit dem angegebenem Mittelpunkt, die ein Gebiet mit dem angegebenem Radius (in Kilometern) zeigt, beim Yahoo-Mapimagedienst an. Mit den Paramtern Höhe und Breite kann die Größe des Bildes in Pixeln festgelegt werden.

C.2. Datenbankabstraktion

Mofa::Model::Object

Basisklasse für die Klassen zur Datenbankabstraktion

Beschreibung

Stellt Funktionen zum Speichern und Auslesen eines Objektes aus einer Datenbanktabelle zur Verfügung.

Die Methoden `unquoted_fields`, `quoted_fields`, `foreign_keys` und `table` müssen von den Klassen, die von Object erben, überladen werden.

Skalare Attribute:

Für alle öffentlichen Attribute gibt es eine gleichnamige Methode, die immer den Wert des Attributes zurückgibt, und den Wert des Attributes setzt, wenn man ihr einen Parameter übergibt.

`id` - (`int`) Primary Key des Datensatzes zu diesem Objekt.

Foreign-Key-Attribute

Für alle Foreign-Key-Attribute gibt es eine gleichnamige Methode, die den Wert des Attributes als Objekt zurückgibt, und den Wert des Attributes setzt wenn man ihr ein Objekt oder einen Skalar, der Primary-Key der entsprechenden Tabelle ist, übergibt.

Zusätzlich gibt es eine Methode AttributnameId(), mit der nur der Primary-Key des Attributes zurückgegeben wird.

Öffentliche Klassenmethoden

`$ new($)`

Parameter: [Hashreferenz]

Rückgabewert: Object-Objekt

Gibt ein neues Objekt vom Typ Öbjectßurück. Falls ein Parameter übergeben wird, werden die Attribute mit den im übergebenem Hash spezifizierten Werten initialisiert.

`$ get($)`

Parameter: Id eines Datensatzes (`int`)

Rückgabewert: Object-Objekt

Liest das Objekt mit der spezifizierten Id aus der Datenbank.

`@ get_ids()`

Rückgabewert: Liste von Ids aller Datensätze in der Tabelle für diese Klasse ((`int,..`))

Liest alle Ids von Objekten des Typs Öbjectäus der Datenbank.

`@ create_table()`

Rückgabewert: Liste von `string` mit Statusinformationen

Legt die Tabelle, in der die Objekte dieser Klasse gespeichert werden, in der Datenbank an.

Öffentliche Objektmethoden

`$ add()`

Rückgabewert: 1 bei Erfolg, <1 sonst (`int`)

Fügt das Objekt zur Datenbank hinzu. Der Wert des Attributs id wird auf die id des neuen Eintrags gesetzt.

`$ update(@)`

Parameter: id_col (`string`), col_to_update (`string`), col_to_update, ...

Rückgabewert: 1 bei Erfolg, < 1 sonst (`int`)

Aktualisiert alle Einträge in der Datenbank, deren Wert für das Attribut id_col mit dem Wert des Attributes in diesem Objekt übereinstimmt. Werden weitere Attribute angegeben, (col_to_update), werden nur diese Attribute mit den Werten dieses Objektes überschrieben, sonst alle.

Protected Klassenmethoden

`@ unquoted_fields()`

Rückgabewert: Liste der Spalten, die kein Text beinhalten ((`string`,..))

`@ quoted_fields()`

Rückgabewert: Liste der Spalten, die Text beinhalten ((`string`,..))

`% foreign_keys()`

Rückgabewert: Paare (Spalte, Tabelle) von Spalten die Foreign Key einer anderen Tabelle sind. ((`string`,`string`,..))

`@ fields()`

Rückgabewert: Liste aller Spalten ((`string`,..)))

`$ table()`

Rückgabewert: Name der Tabelle, in der Objekte dieser Klasse gespeichert werden (`string`)

`$ dbh()`

Rückgabewert: Datenbankhandle auf die Datenbank, in der die Objekte gespeichert werden.

Protected Objektmethoden

`$ dbval($)`

Parameter: Attributname

Rückgabewert: Wert des Attributs, wie er in die Datenbank geschrieben wird.

`$ _get_set($@)`

Parameter: Attributname (`string`), [Skalar]

Rückgabewert: Wert des Attributes

Falls 2 Parameter übergeben werden, wird der Wert des Attributes gesetzt.

`$ _get_set_fkey($$@)`

Parameter: Attributname (`string`), Klassenname (`string`), [Skalar, Skalar]

Rückgabewert: Wert des Attributes als Objekt des entsprechenden Typs.

Falls 4 Parameter übergeben werden, wird der letzte Parameter als Wert des Attributes gesetzt. Der 3. Parameter wird immer ignoriert.

Mofa::Model::Point

Objekte dieser Klasse repräsentieren einen Punkt auf der Oberfläche der Erde. Erbt von: `Mofa::Model::Object`

Beschreibung

This class represents a point on the earth. The coordinates can be referred as latitude-longitude dezimal coordinates with `Mofa::Model::Point::ll()` or as utm-coordinates in meters with `Mofa::Model::Point::utm()`. These methods read without parameter or set with 2 (or 4 in case of utm) parameters. You should only use WGS84-coordinates.

Klassenmethoden

`$ new($)`

Parameter: Hashreferenz

Rückgabewert: `Point`-Objekt

Beispiele:

```
$pt =
    Mofa::Model::Point->new({
        utm_x=>413794, utm_y=>5470842,
        utm_e=>32,     utm_n=>"U"
    });
$pt = Mofa::Model::Point->new({
        utm_x=>413794, utm_y=>5470842
    });
# 32, "U" will be used as values for utm_e and utm_n.

$pt = Mofa::Model::Point->new({x=>49.39, y=>8.81});
$pt = Mofa::Model::Point->new({
          coord=>{X=>"49 25 37N", Y=>"7 45 02E"}
      });
```

`($$) dms2dez($$)`

Parameter: Koordinaten in Grad Minute Sekunde (`string`, `string`)

Rückgabewert: Koordinaten als Dezimalzahl (`float`, `float`)

Beispiel:

```
($x, $y) =
  Mofa::Model::Point::dms2dez("49 23 03N", "7 48 44E");
```

`($$) utm2ll(%)`

Parameter: Hash mit Attributen utm_x, utm_y, utm_e, utm_n

Rückgabewert: Koordinaten als Dezimalzahl (`float`, `float`)

Beispiel:

```
($x, $y) =
    utm2ll({
        utm_x=>413794, utm_y=>5470842,
        utm_e=>32,     utm_n=>"U"
    });
```

`($$$$) ll2utm(%)`

Parameter: Hash mit Attributen x, y

Rückgabewert: Utm-Koordinaten

Beispiel:

```
($meters_x, $meters_y, $square_e, $square_n)
    = ll2utm({x=>49.39, y=>8.81});
```

Objektmethoden

`($$$$) utm(@)`

Parameter: [utm_x (`int`), utm_y (`int`), [utm_e (`int`), utm_n (`string`)]]

Rückgabewert: (utm_x, utm_y, utm_n, utm_e)

Beispiele:

```
Lesen:  ($utm_x, $utm_y)   = $pt->utm;
Setzen: $pt->utm(413794, 5470842, 32, 'U');
        $pt->utm(413794, 5470842);
```

`($$) ll(@)`

Parameter: [x (`int`), y (`int`)]

Rückgabewert: (x, y)

Beispiele:

```
Lesen:  ($x, $y) = $pt->ll;
Setzen: $pt->ll(49.32, 7.32);
```

Mofa::Model::Area

Ein Gebiet auf der Erdoberfläche. Erbt von: Mofa::Model::Point

Objektmethoden

`$ contains($@)`

Parameter: Point-Objekt, [Radius (Int)]

Rückgabewert: 1, falls der übergebene Punkt in dem Gebiet, vergrößert nach allen Seiten um den Radius, liegt. 0 sonst.

`@ asPolygon(@)`

Parameter: Schrittweite (Int)

Rückgabewert: Liste von Point-Objekten, die ein Poylgon beschreiben, das dieses Gebiet annähert.

Mofa::Model::CircularArea

Repräsentiert ein Gebiet in Form eines Ring-Ausschnitts auf der Erdoberfläche, (Typischerweise verwendet zur Beschreibung des Ergebnis einer Positionsbestimmungsanfrage). Erbt von `Mofa::Model::Area`.

Skalare Attribute

`inRadius (int)` - innerer Radius des Rings in Metern

`outRadius (int)` - äußerer Radius des Rings in Metern

`startAngle (int)` - Winkel zu Nord, ab dem das Gebiet beginnt

`stopAngle (int)` - Winkel zu Nord, bei dem das Gebiet endet.

Klassenmethoden

`$ new($)`

Parameter: Hashreferenz

Rückgabewert: `CircularArea`-Objekt

Die Hashreferenz muss zusätzlich zu den Attributen, die zur Initialisierung eines Point-Objekt notwendig sind, die Attribute inRadius, outRadius, startAngle und stopAngle enthalten.

`$ newcircle($$)`

Parameter: `Point`-Objekt, Radius (`int`)

Rückgabewert: `CircularArea`-Objekt, das die Kreisfläche mit dem Radius um den angegebenen Punkt repräsentiert

Mofa::Model::Address

Objekte dieser Klasse repräsentieren einen Punkt auf der Erdoberfläche mit Koordinaten und mit Adresse. Erbt von: `Mofa::Model::Point`.

Skalare Attribute

`nr (int)` - Hausnummer

`description (string)` - kurze Beschreibung zur Darstellung wo die vollständige Adresse nicht hinpasst z.B. "Kaiserslautern Bus Hauptbahnhof"

`street (string)` - Straße

`city (string)` - Stadt

`zip (string)` - Postleitzahl

`region (string)` - Bundesland

`district (string)` - Kreis

`state (string)` - Staat (Abkürzung, Deutschland = 'de')

Mofa::Model::MeetingPt

Punkt auf der Erdoberfläche mit Adresse und Name, der als Treffpunkt zwischen Mitfahrer und Autofahrer geeignet ist. Erbt von: `Mofa::Model::Address`.

Skalare Attribute

`name (string)` - Name des Punktes, der ihn eindeutig identifiziert und knapp beschreibt

`distance (float)` - Entfernung in Kilometern vom Aufenthaltsort (wird nicht in DB gespeichert.)

Klassenmethoden

`@ get_points_in_area($@)`

Parameter: `Area`-Objekt, [Max. Vergrößerung (`int`), [Min. Treffer (`int`)]]

Rückgabewert: Liste von `Point`-Objekte innerhalb des Area-Objektes.

Sucht Punkte im übergebenem Area-Objekt. Sind noch nicht genug Punkte gefunden worden, wird das Area-Objekt um einen bestimmten Radius vergrößert, und die Suche wiederholt, bis genug Punkte gefunden wurden (Min Teffer), oder die Maximale Vergrößerung erreicht wurde. Alle gefunden Punkte werden zurückgegeben.

Mofa::Model::Distance

Strassenentfernung und Fahrtzeit zwischen zwei `Point`-Objekten. Erbt von: `Mofa::Model::Object`.

Skalare Attribute

`distance (float)` - Abstand in Kilometern mit 2 Stellen nach dem Komma

`time (int)` - Fahrtzeit in Minuten

Foreign-Key-Attribute

`start (Mofa::Model::Address)` - Startpunkt

`startId (int)`

`destination (Mofa::Model::Address)` - Zielpunkt

`destinationId (int)`

Klassenmethoden

`$ get($$)`

Parameter: Ids 2er `Point`-Datensätze (`int`)

Rückgabewert: `Distance`-Objekt mit dem Abstand zwischen beiden Punkten, falls der Abstand bereits in der Datenbank steht. Undef sonst.

`$ add($$$$)`

Parameter: Id des Start und des Zielpunkt (`int`), Entfernung (`int`), Zeit (`int`)

Rückgabewert: 1, falls erfolgreich in Datenbank eingetragen. <1, sonst.

Trägt den Abstand zwischen diesen beiden Punkten in die Datenbank ein. Falls schon ein Eintrag existiert, wird er aktualisiert.

Mofa::Model::GeocodetPt

Punkte auf der Erdoberfläche mit Koordinate und Adresse, sowie mit einer Genauigkeit, wie gut sie zu der Geocode-Anfrage passen. Objekte dieser Klasse werden von `Mofa::Model::Geocode`-Objekten aggregiert. Erbt von: `Mofa::Model::Address`.

Skalare Attribute

`accuracy (int)` - Gibt an, wie gut das Objekt zur Geocode-Anfrage passt.

Mofa::Model::Geocode

Antwort auf eine Geocode-Anfrage. Erbt von: `Mofa::Model::Object`.

Klassenmethoden

`$ new($@)`

Parameter: Geocode-Anfrage (`string`), Liste von passenden GeocodedPt-Objekten

Rückgabewert: neues `Geocode`-Objekt.

`$ get($)`

Parameter: Geocode-Anfrage (`string`)

Rückgabewert: Zu dieser Geocode-Anfrage passendes `Geocode`-Objekt

Alle zu dieser Anfrage passenden Punkte werden in der Datenbank gesucht, und zum neuem Geocode Objekt hinzugefügt.

Objektmethoden

`@ points()`

Rückgabewert: Liste der `GeocodedPts` in diesem `Geocode`-Objekt.

`$ push(@)`

Parameter: Liste von `GeocodedPts`

Fügt die übergeben GeocodetPts zu diesem Objekt hinzu.

Mofa::Model::Person

Personen mit Accounts, die Angebote und Gesuche in die Mitfahrzentrale einstellen können. Erbt von: `Mofa::Model::Object`.

Skalare Attribute

`type (int)` - Typ - ob nur Gesuche, nur Angebote, oder beides

`bday (int)` - Geburtsdatum

`id (string)` - (Statt Id (Int) wie Mofa::Model::Object): Login

`password (string)` - Password

`name (string)` - Name

`prename (string)` - Vorname

`cellular (string)` - Handynummer (an die SMS-Benachrichtungen gesendet werden)

`msisdn (string)` - ID, mit der das Handy identifiziert wird, wenn es lokalisiert werden soll.

`banking_name (string)` - Kontoinhaber

`bank_code (string)` - BLZ

`account_nr (string)` - Kontonummer

Foreign-Key-Attribute

`address (Mofa::Model::Adress)` - Adresse der Person

`addressId (int)`

Mofa::Model::Lift

Mitfahrgelegenheiten (kann Angebot oder Gesuch sein). Erbt von: `Mofa::Model::Object`

Skalare Attribute

`startTime (int)` - Startzeit

`arrivalTime (int)` - Ankunftszeit

`timeAccuracy (int)` - Maximale aktzeptierte Abweichung von der Startzeit

`seats (int)` - Freie Sitzplätze

`fee (int)` - Gebühren für die Mitnahme

Foreign-Key-Attribute

`start (Mofa::Model::MeetingPt)` - Startort

`startId (int)`

`destination (Mofa::Model::MeetingPt)` - Zielort

`destinationId (int)`

Klassenmethoden

`@ get_touching($$$$)`

Parameter: Koordinaten des unteren rechten und oberen linken Punkt eines zu den UTM-Achsen parallelen Rechtecks

Rückgabewert: Liste von allen Objekten der Klasse auf die die Methode aufgerufen wird (zum Bsp: `Mofa::Modell::Offer` oder `Mofa::Model::Request`), die das spezifizierte Rechteck berühren, und möglicherweise noch einige die es nicht berühren, die jedoch nicht ausgefiltert wurden.

Mofa::Model::Offer

Mitfahrangebote. Erbt von: `Mofa::Model::Lift`

Foreign-Key-Attribute

`provider (Mofa::Model::Person)` - Anbieter dieses Mitfahrangebotes

`providerId (string)`

Mofa::Model::Request

Mitfahrgesuche. Erbt von: `Mofa::Model::Lift`.

Foreign-Key-Attribute

`requester (Mofa::Model::Person)` - Person, die diese Mitfahrgelegenheit sucht.

`requesterId (string)`

Mofa::Model::Mapped

Mitfahrgelegenheit die gerade vermittelt wird. Erbt von: `Mofa::Model::Object`

Skalare Attribute

`accepted (int)` - Ist die Vermittlung abgeschlossen?

`detour (int)` - Umweg für den Autofahrer in Kilometern.

`addtime (int)` - Zusätzliche Fahrtzeit für den Autofahrer.

Foreign-Key-Attribute

`offer (Mofa::Model::Offer)` - Mitfahrangebot das vermittelt wird.

`offerId (int)`

`request (Mofa::Model::Request)` - Mitfahrgesuch das vermittelt wird.

`requestId (int)`

Objektmethoden

`$ add()`

Rückgabewert: 1 bei Erfolg, < 1 sonst.

Fügt diesen Mapped-Datensatz zur DB hinzu, falls noch kein Mapped-Datensatz mit gleichem Anbieter und Mitfahrer und mit gleichem Ziel und Start vorhanden ist. Sonst gibt es -3 zurück und ändert das aktuelle Mapped-Objekt in das zu dem entsprechenden passenden Datensatz gehörige.

Klassenmethoden

`@ get_by_provider($)`

Parameter: `Person`-Objekt

Rückgabewert: Liste von Mapped-Objekten mit Angeboten dieser Person

`@ get_by_provider($)`

Parameter: `Person`-Objekt

Rückgabewert: Liste von Mapped-Objekten mit Gesuchen dieser Person

`$ accept($)`

Parameter: Id eines Mapped-Datensatzes (`int`)

Markiert diesen Mapped-Datensatz in der DB als aktzeptiert

`$ deny($)`

Parameter: Id eines Mapped-Datensatzes (`int`)

Markiert diesen Mapped-Datensatz in der DB als abgelehnt

C.3. Mofa::View

Stellt Funktionen zur Ausgabe von WML-Seiten zur Verfügung

`$ get_link(@)`

Parameter: Datei, auf die verlinkt werden soll (`string`), Liste von Referenzen auf 2-Elementige Listen mit Paaren der Form (key, value) für zu übermittelnde Daten

Rückgabewert: String mit Get-Link zu der angebene Seite (mit Session-Id)

`$ post_link(@)`

Parameter: Datei, auf die verlinkt werden soll (`string`), Liste von Referenzen auf 2-Elementige Listen mit Paaren der Form (key, value) für zu übermittelnde Daten

Rückgabewert: String mit Post-Link zu der angebene Seite (mit Session-Id)

`$ back_link()`

Rückgabewert: `string` mit zurück-Link

`$ logout_link()`

Rückgabewert: falls angemeldet: `string` mit Link um sich abzumelden; sonst: leerer `string`

`$ card_login()`

Rückgabewert: `string` mit WML-Card: komplettes Login-Formular

`$ card_anon_start()`

Rückgabewert: `string` mit WML-Card: Startseite optimiert für nicht angemeldete Benutzer

`$ card_search($$$)`

Parameter: Vorgaben für Startpunkte (Referenz auf Liste von `Mofa::Model::Address`-Objekten), Vorgaben für Endpunkte (Referenz auf Liste von `Mofa::Model::Address`-Objekten), URL an die Formular gesendet werden soll (`string`)

Rückgabewert: `string` mit WML-Card: Suchformular mit Start, Ziel, Zeit

`$ card_entry($$)`

Parameter: Vorgaben für Startpunkte (Referenz auf Liste von `Mofa::Model::Address`-Objekten)), Vorgaben für Endpunkte (Referenz auf Liste von `Mofa::Model::Address`-Objekten), URL an die Formular gesendet werden soll (`string`)

Rückgabewert: `string` mit WML-Card: Eingabeformular für neue MFG

`$card_infos()`

Rückgabewert: `string` mit WML-Card: Infos/Impressum

`$ card_bookmark()`

Rückgabewert: `string` mit WML-Card: Enthält Link der als Bookmark zum automatischem Login gespeichert werden kann.

`$ card_personal_menu()`

Rückgabewert: `string` mit WML-Card: Startseite nach dem Einloggen.

`login()`

Checks if the submitted parameters match a login-id and password from the DB, saves the login-id to the session and prints login_splash() in case of success or login_failure() if no success.

`personal_index()`

Gibt ein WML-Deck mit einigen WML-Cards als Startseite nach dem einloggen aus. Falls der Benutzer nicht eingeloggt ist wird stattdessen login_failure() ausegegeben.

`anon_index()`

Gibt ein WML-Deck mit einigen Cards als allgemeine Startseite aus. Falls logout=1 übermittelt wurde, wird der Benutzer auch noch ausgeloggt.

`logout()`

Löscht die Session. Gibt WML-Deck aus mit der Meldung, dass man ausgeloggt wurde und Auto-Login-Lesezeichen ungültig wurden.

`confirm_register()`

Gibt WML-Deck mit Meldung, dass Registrierung abgeschlossen ist, aus. Leitet an search.pl weiter, wo die Suche fortgesetzt werden kann.

`display_map($$$$$$)`

Parameter: URL zum Kartenbild (`string`), Parameter für dieses Bild (X, Y, Radius, Höhe, Breite) als `int`

Gibt WML-Deck aus, mit dem Bild an der angegebenen URL und Links zum auszoomen, einzoomen und evtl. zur Navigation.

`display_accept(@)`

Parameter: Liste von `Mapped`-Objekten

Gibt WML-Deck mit einer Übersichts-Card und einer WML-Card für jedes Mapped-Objekt an. Bietet Optionen ein Anfrage zu aktzeptieren oder abzulehen.

`display_search_form()`

Gibt WML-Deck mit einer Card mit dem Suchformular aus.

`display_enter_time()`

Gibt ein WML-Deck mit einer Card zur Eingabe der Startzeit einer Mitfahrgelegenheit aus.

`card_enter_point($$$@)`

Parameter: Typ des einzugebenden Punktes (start oder destination) (`string`), Anzuzeigende Beschreibung des einzugebenden Punktes (`string`), id der Card (`string`), Liste von `Mofa::Address`-Objekten die zur Auswahl zur Verfügung stehen.

Rückgabewert: `string` mit einer WML-Card mit einem Formular zur Eingabe oder Auswahl eines Treffpunktes.

`display_enter_point($$$$)`

Parameter: Typ des einzugebenden Punktes (start, destination) (`string`), je 3 Array-Referenzen auf eine Liste von `Mofa::Model::Address`-Objekten mit: Punkten in der Umgebung, Punkten die früher schonmal gewählt wurden, geocodierten Punkten.

Gibt eine WML-Seite zur Eingabe oder Auswahl eines Treffpunktes aus.

`display_search_result()`

Gibt WML-Deck mit der Aufforderung sich zu registrieren aus.

`display_err(@)`

Parameter: Fehlermeldungen (`(string,..)`).

Gibt WML-Deck mit Titel Fehler und jedem übergebenem String in je einer Zeile aus. Bittet Browser, die Seite immmer neu zu laden und nicht im Cache abzulegen.

`display_msg($@)`

Parameter: Überschrift (`string`), Nachrichten (`(string,..)`).

Gibt WML-Deck mit dem Titel und jeder Nachricht in einer eigenen Zeile aus. Deck wird nicht im Cache zwischengespeichert.

`login_splash()`

Gibt WML-Deck aus mit einer Meldung über erfolgreichen Login, und leitet sofort an p_index.pl weiter. Dient dazu, Login und Passwort aus der URL zu entfernen, damit der Benutzer, falls er Lesezeichen speichert, nicht sein Passwort im Klartext im Lesezeichen mitspeichert.

`login_failure`

Gibt WML-Deck mit einer Fehlermeldung aus. Seite wird nicht im Cache gespeichert.

`sendwml(@)`

Parameter: WML-Cards (`(string..)`)

Gibt WML-Deck mit den übergebenen Cards aus.

`sendwml(@)`

Parameter: WML-Cards (`(string..)`)

Gibt WML-Deck mit den übergebenen Cards aus. Deck wird nicht im Cache gespeichert, sondern immer neu geladen.

`myparam(@)`

Siehe Dokumentation der `param`-Methode des `CGI`-Moduls