

SITEX - Documentation



index.php



/index.php

Fichier d'entrée de l'application. Toutes les requêtes passent par là. Ce fichier se charge de démarrer la session, charger la bonne configuration, créer les droits de l'utilisateur actuellement loggé, etc



request.inc.php



/INC/request.inc.php

Regroupe des fonctions de gestion des requêtes, des droits, de l'authentification.

function *gereRequete()*

Fonction gereRequete: toutes les requêtes venant du JS passe par ici.

La fonction reçoit une requête en paramètre qui est évaluée dans la liste des requêtes prises en charges

Liste des requêtes supportées :

Requête	Action
sem04	Affiche un message dans le #contenu du site
sem03	Affiche un message dans le #contenu du site
TPsem05	Retourne le formulaire du TP05
gestLog	Connecte ou déconnecte l'utilisateur
formSubmit	Redirige vers la gestion de formulaire
displaySession	Affiche en #debug le contenu de la session
clearLog	Clear les logs stockés dans la session
resetSession	Destroy et redémarre une nouvelle session
config	Load le fichier de config et renvoie le formulaire
testDB	Effectue plusieurs appels de procédure en DB pour tester son fonctionnement

Paramètres

string \$rq : requête provenant du JS

Return

int : La fonction retourne -1 si l'utilisateur n'as pas le droit d'effectuer une action

void : La fonction ne retourne rien le reste du temps

function *gereSubmit()*

Cette fonction gère tous les différents formulaires du site.

Tous les formulaires du site ont leur action qui renvoie vers la requête **gereSubmit**. On récupère le **senderId** dans cette fonction, ce qui nous permet de déterminer quel traitement effectuer parmi les formulaires supportés.

Liste des formulaires supportés :

Formulaire	Action
formTP05	Récupère dans la DB, les cours pour le groupe sélectionné et renvoie une action makeTable au JS
modifConfig	Sauve la nouvelle config dans le fichier, reloads la nouvelle config dans la Session et envoie les changements dans l'action layout
formLogin	Vérifie que le username et mot de passe sont corrects et authentifie l'utilisateur (cfr. authentication)

function *peutPas()*

Cette fonction détermine si un utilisateur à le droit d'effectuer une requête en se basant sur la liste de droits liée à l'utilisateur **\$_SESSION['droits']**. Dans le cas où l'utilisateur n'as pas le droit, on bloque l'exécution de la requête et on envoie une boîte de dialogue à l'utilisateur via l'action **peutPas**.

On gère aussi les cas où l'utilisateur est en cours de réactivation, une boîte de dialogue différente est alors envoyée à l'utilisateur

Paramètres

string \$req : requête de l'utilisateur veut effectuer

Return

boolean : Renvoie **TRUE** si l'utilisateur **NE PEUT PAS** effectuer la requête.

boolean : Renvoie **FALSE** si l'utilisateur **A LE DROIT** d'effectuer la requête.

function *KLogin()*

Cette fonction se charge de l'affichage du formulaire de login. La fonction charge le template du formulaire et l'envoie dans un retour **formLogin**.

function *KLogout()*

Cette fonction se charge de déconnecter l'utilisateur. La fonction supprime les données de l'utilisateur dans la session et envoie un retour **logout** qui affiche une boîte de dialogue qui confirme le logout de l'utilisateur

function *authentication()*

Cette fonction se charge de l'authentification de l'utilisateur. La fonction reçoit un utilisateur en paramètre, récupère les profils de l'utilisateur, crée les droits de l'utilisateur et stocke toutes les données en session.

Paramètres

array \$user : tableau contenant les informations concernant l'utilisateur

Return

int : Retourne **-1** si l'utilisateur est en cours d'activation

void : Envoie une action **userConnu** avec les données du user

function *sendMakeTable()*

Envoie un array à l'action **makeTable**. Cela affiche le tableau passé en paramètre en HTML5 dans la zone **#contenu**.

Paramètres

array \$tab : Tableau à envoyer au JS


function *callResAjax()*

Appelle la fonction ajax mise à disposition dans les ressources de Delvigne. Les requêtes non supportées par notre application passeront par cette fonction.

Paramètres

string \$rq : Requête demandée

function *chargeTemplate()*

Charge le fichier de template dont le nom est passé en paramètre. Le fichier doit se trouver dans le dossier  INC/ et avoir un nom commençant par **template.** et se terminant par **.inc.php**. La fonction renvoie false si le fichier n'existe pas.

Paramètres

string \$name : Nom du fichier de template à charger

Return

string : Renvoie le contenu du fichier de template

bool : Renvoie **false** si le fichier n'existe pas

function *tpSem05()*

Dans cette fonction se trouvent tous les envois nécessaires au fonctionnement du formulaire du tp5. La fonction charge le template du formulaire tp5 (cfr. **chargeTemplate()**). La fonction fait un appel à la

procédure **allGroups** et renvoie les données dans l'action **data**. Ces 'data' servent à remplir le sélecteur de groupes dans le formulaire.



droits.inc.php



/INC/droits.inc.php

Cette librairie permet de gérer les droits des utilisateurs et de générer les menus selon les droits. Ce fichier est protégé, il ne peut être ouvert directement et doit être inclu par le fichier index.

function *isAuthenticated()*

Vérifie qu'un utilisateur est connecté.

Return

bool : Renvoie **true** si un utilisateur est connecté.

function *isAdmin()*

Vérifie si l'utilisateur à un profil Administrateur.

Return

bool : Renvoie **true** si un utilisateur ayant le profil Administrateur est connecté

function *isSousAdmin()*

Vérifie si l'utilisateur à un profil Sous-administrateur.

Return

bool : Renvoie **true** si un utilisateur ayant le profil Sous-administrateur est connecté

function *isReactiv()*

Vérifie si l'utilisateur a le status de réactivation.

Return

bool : Renvoie **true** si un utilisateur au status réactivation est connecté

function *isMdpp()*

Vérifie si l'utilisateur a le status de mot de passe perdu.

Return

bool : Renvoie **true** si un utilisateur au status de mot de passe perdu est connecté

function *isEdit()*

Vérifie si l'utilisateur a les droits d'édition pour les informations du site. (Fonction non utilisée).

Return

bool : Renvie **true** si l'utilisateur est Administrateur ou n'est pas en réactivation

function *creeDroits()*

Crée les droits de l'utilisateur en fonction de son profil et de ses status et stocke ces droits en session.

Return

int : Renvoie **-1** si l'utilisateur n'est pas authentifié

int : Renvoie **-2** si l'utilisateur n'est pas en réactivation ou si l'utilisateur est Administrateur

function *creeMenu()*

Génère dynamiquement les menus à afficher en fonction du profil et des status de l'utilisateur.

string : Renvoie une chaîne de caractères contenant les menus à afficher au format HTML5



config.inc.php

Classe Config :

- Gère la récupération de configuration depuis des fichiers **.ini**
- Gère la génération d'un formulaire HTML5 pour modifier la config
- Gère la sauvegarde dans un fichier **.ini** de la nouvelle config

Attribut **\$filename**

Nom du fichier de configuration.

Valeurs : null | string

Attribut **\$fileExist**

Vrai si le fichier de config existe.

Valeurs : bool

Attribut **\$config**

Tableau multidimensionnel contenant la configuration une fois chargée.

Valeurs : array

Attribut **\$saveError**

Contient les erreurs rencontrées durant le processus de sauvegarde.

Valeurs : int

function *__construct()*

Constructeur de la classe Config.

Si il reçoit un filename en paramètre, il le stocke dans l'attribut et vérifie son existence.

Paramètres

string filename : Nom du fichier de config. (Optionnel)

function *getFilename()*

Getter du filename.

Retourne le nom du fichier de configuration.

Return

null : Pas de nom de fichier enregistré **string** : Le nom du fichier

function *isFileExist()*

Getter de l'attribut fileExist.

Return

bool : Renvoie **true** si le fichier de config existe **bool** : Renvoie **false** si le fichier de config n'existe pas

function *getConfig()*

Getter de la configuration.

Renvoie un tableau multidimensionnel de la configuration actuellement chargée.

Return

array : La configuration

string : Si aucune configuration chargée, Renvoie **Config non chargée**

function *getSaveError()*

Getter de l'attribut saveError.

Renvoie une valeur différente de zéro si le processus de sauvegarde a rencontré une erreur.

Return

int : Le numéro de l'erreur

function *load()*

Charge le fichier de config demandé dans l'attribut config.

La fonction va soit chargé la config du fichier stocké dans l'attribut filename, soit le fichier passé en paramètre.

Paramètres

string filename : Nom du fichier de config à charger. (Optionnel)

Return

array : Renvoie la config si le chargement s'est bien passé

bool : Renvoie **false** si la fonction de lecture du fichier a rencontré une erreur

string : Renvoie un string si le fichier de config n'existe pas.

function **save()**

Sauvegarde la nouvelle configuration (contenue dans **\$_POST**) à l'emplacement demandé.

Si une erreur est rencontrée durant le processus, renvoie une chaîne avec l'erreur.

Paramètres

string filename : Fichier dans lequel sauvegarder la config

Return

string : Le message d'erreur

function **getForm()**

Génère dynamiquement un formulaire en se basant sur le fichier de config actuellement chargé.

Return

string : Renvoie une chaîne de caractère contenant le formulaire HTML5

function **getBloc()**

Génère les différents blocs du formulaire.

Paramètres

string k : Nom du bloc

array v : Tableau des éléments de ce bloc

Return

array : Renvoie un tableau des lignes du blocs généré

function **saveErrorMessage()**

Retourne le bon message d'erreur selon le numéro passé en paramètre.

Paramètres

int error : Numéro de l'erreur

Return

string : Message d'erreur



db.inc.php

Classe DB :

- Gère la connection à la base de données
- Gère les appels de procédures

Attribut **\$db**

Contient les informations de connection de la DB depuis la config.

Valeurs : array

Attribut **\$pdoException**

Contient les exceptions renvoyées par la connexion à la DB.

Valeurs : Exception | PDOException

Attribut **\$iPdo**

Contient l'instance de l'objet PDO.

Valeurs : PDO

function **__construct()**

Constructeur de la classe DB.

Initialise la connection à la DB sur base des infos de connexion venant de la config.

function **getException()**

Gettre de l'attribut pdoException.

Renvoie l'exception catché durant la connexion à la BD ou le call de procédure.

Return

string : Le message de l'exception

function **getServer()**

Retourne le serveur sur lequel l'application tourne actuellement.

Return

string : Renvoie **localhost** ou l'adresse ip du serveur

function **call_v1()**

Effectue un call à la procédure **mc_allGroups**.

Return

array : Retour de la procédure

function *call()*

Effectue un call intelligent avec les paramètres passé à la fonction.

Paramètres

string name : Nom de la procédure à appeler

array param : Tableau contenant les paramètres à passer à la procédure (Optionnel)

Return

array : Retour de la procédure



sender.inc.php

Cette librairie gère l'envoi actions au JS.

function *display()*

Affiche la chaine passée en paramètre dans la zone **#contenu** du site.

Paramètres

string txt : Chaine de caractère à afficher

function *error()*

Affiche la chaine passée en paramètre dans la zone **#error** du site.

Paramètres

string txt : Chaine de caractère à afficher

function *debug()*

Affiche la chaine passée en paramètre dans la zone **#debug** du site.

Paramètres

string txt : Chaine de caractère à afficher

function *kint()*

Affiche la chaine passée en paramètre dans la zone **#kint** du site.

La paramètre de cette fonction doit **TOUJOURS** être un retour de la fonction **d()** de la librairie kint.

Paramètres

string txt : Retour de la fonction **d()**

function *toSend()*

Fonction d'envoi d'informations vers JS au retour de l'appel AJAX.

Paramètres

string txt : Chaîne de caractère à transmettre.

string action : Action à laquelle transmettre la chaîne. Par défaut, action **display**.