Connecteur PHP

Cet article décrit le connecteur php, son installation, son paramétrage et son utilisation.

Description du connecteur

Le connecteur php est une application php livrée avec un exemple qui permet de l'illustrer et mettre en oeuvre les fonctionnalités. Ce connecteur permet:

- De réaliser un SSO
- De récupérer les informations de session concernant l'utilisateur
- S'inclure dans l'application le cadre graphique du portail
- De transmettre des informations au portail

Description de la livraison

Le connecteur se compose de deux répertoires:

- kphplib5, il s'agit du connecteur qui doit être installé sur le serveur php
- exemple, il s'agit d'une application qui utilise le connecteur à titre d'exemple

L'application exemple comporte deux vues, l'une réduite, l'autre maxi.
La configuration du connecteur se fait dans l'application qui l'utilise, c'est cette configuration que nous allons détailler

Configuration du connecteur

dans chacun des deux scripts php (vue_reduite.php et vue_maxi.php), la configuration du connecteur est incluse au début du script, il s'agit du fichier config/cfg.php.
Ce fichier doit être paramétré en fonction des conditions particulières d'utilisation

```
define("KPHPLIB_PATH","C:program fileseasyphpwwwkphplib5");
define("KPHPLIB_PATH_INCLUDE","C:program
fileseasyphpwwwkphplib5include");
define("KPHPLIB_PATH_CLASSE","C:program
fileseasyphpwwwkphplib5classe");
define("KPHPLIB_PATH_CLASSE_SSO","C:program
fileseasyphpwwwkphplib5classesso");
define("KPHPLIB_PATH_CLASSE_DATA","C:program
fileseasyphpwwwkphplib5classedata");
define("KPHPLIB_PATH_CLASSE_UTIL_XML","C:program
fileseasyphpwwwkphplib5classeutilxml");
define("KPHPLIB_PATH_CLASSE_UTIL_HTTP","C:program
fileseasyphpwwwkphplib5classeutilhttp");
```

Ces chemins indiquent les fichiers du connecteur php, remplacer C:program fileseasyphpwwwkphplib5 par le chemin physique sur le serveur de production.

```
// chemin vers le fichier JSP sur Kportal qui va renvoyer le flux
XML
define("SSO_PATH_XML","/adminsite/sso/validation_ticket.jsp?kticket=");
```
Ce chemin ne devrait pas modifié, il est complété avec le host décrit plus bas

```
//url de page d'erreur
define("ERROR_URL","/jsp/page_erreur.jsp");
```
Indique la page à retourner en cas d'erreur dans le connecteur

```
//url de page de login k-portal
define("LOGIN_URL","/adminsite/login.jsp");
```
indique de la même manière l'url de login du portail

```
define("PATH_HTML_KCONNECT","/adminsite/dsi/kconnect.jsp");
define("PATH_HTML_PARAM","&krequete=");
define("DATA_FILE_PREFIX","req_");
define("DATA_FILE_EXT",".xml");
```
Cette section ne doit pas être modifiée

```
//host de l'appli
define("HOST","kportaldemo42:8080");
define("PATH_ABSOLU_APPLI","http://localhost/kphp5/");
```
Il s'agit du host principal de l'application qui validera le ticket
il faut également indiquer le path du connecteur php accessible par le portail

```
// On déclare l'url du serveur sur lequel tourne K-Portal. Ici
il tourne sur le même serveur que notre application PHP
define("SSO_URL_SERVER","kportaldemo42:8080");
define("SSO_HOST_SERVER","kportaldemo42");
define("SSO_PORT_SERVER","8080");
```
Ici on précise le host de notre portail, en version complète et en séparant le host et le port

Utilisation de l'api php

Une fois le connecteur configuré, on peut utiliser l'api dans une application, dans cette section, les vues maxi et réduite sont parcourues pour expliciter les différentes fonctions du connecteur.

Vue réduite

```
    setlocale (LC_ALL, "fr");
    include_once("config/cfg.php");



    $bean = new sso();
```

```
    connecteurMgr::validerTicket($bean);



    if ($bean == "") {



        // li n'y avait pas de ticket à valider
        // On regarde si la session a déjà
été initialisée
        $bean = $_SESSION["SSOBEAN"];

        if ($bean == "") {
            $msgErreur = "Accès interdit";
        }
    } else {
       // On analyse le code retour du ticket
        if ($bean->code_retour == "100")
            $msgErreur = "Session fermée";
        if ($bean->code_retour == "200")
            $msgErreur = "Session fermée";
        if ($bean->code_retour == "300")
            $msgErreur = "Accès interdit";
        if ($bean->code_retour == "400")
            $msgErreur = "Problème de connexion";
    }
    if
($bean->code_retour!=""&&$bean->code_retour!="0")
    {
        header("location:http://".SSO_URL_SERVER.ERROR_URL);
    }
```

Pour que le connecteur fonctionne, les sessions doivent être utilisées. La classe sso est la classe principale, on crée une instance avec $bean, la méthode validerTicket se charge du SSO est propose des codes retours que l'application peut exploiter à sa guise.
Le bean est stocké dans la session, ici on regarde si un bean existe déjà dans la session. Sinon, on analyse le code retour.

```
    $user=$bean->code_utilisateur_kportal;
    $nom=$bean->nom;
```

Les informations de l'utilisateur sont accessibles en propriétés de $bean

```
connecteurMgr::genererUrlTicket("KPHP")
```

Cette méthode permet d'insérer un lien vers la vue maxi associée à ce service, qui sera généré par le portail, en proposant un nouveau tichet. Ici KPHP est le code du service, il pourrait également être récupéré à l'aide du bean.

Vue maxi

Dans ce script on utilise les même principes que précédemment, des fonctions complémentaires sont toutefois utilisées

```
    connecteurMgr::invaliderCache("KPHP");
```

Cette méthode permet de réinitialiser le cache du portail pour ce service, qui sera alors recharché pour une prise en compte dynamique d'informations rafraîchies.

```
    $strRubrique="1139212273390";
    $strStructure="";
    $objRequete=new requete($strRubrique,$strStructure);
    $langue = $_SESSION["LANGUE"];
    $secure = $_SESSION["SECURE"];
    if (isset($langue))
        $objRequete->setLangue($langue);
    if (isset($secure))
        $objRequete->setSecure("1"==($secure));




    $strTitre="Encadre 1";
    $strContenu="contenu_encadre 1";
    $objEncadre1=new encadre($strTitre,$strContenu);
    $strTitre="Encadre 2";
    $strContenu="contenu encadre 2";
    $objEncadre2=new encadre($strTitre,$strContenu);




    $objEncadreRecherche=new liste_encadres_recherche();

    $objEncadreRecherche->addEncadre_recherche("0002");




    // nouveau, fournir des variables au portail
    $objDonneesSpecifiques=new
donnesSpecifiques("NOM_APPLI","Incidents nformatiques");




$objRequete->addDonneesSpecifiques($objDonneesSpecifiques);
```

```
    $objRequete->addEncadre($objEncadre1);
    $objRequete->addEncadre($objEncadre2);

$objRequete->addEncadreRecherche($objEncadreRecherche);
```

```
    $GLOBALS["objRequete"]=$objRequete;
```

Ces instructions permettent notamment de:

- forcer la rubrique et la structure d'affichage ($strRubrique,$strStructure)
- forcer la langue du portail
- forcer l'utilisation du https (SECURE)
- Créer des encadrés avec titre et contenu
- Proposer des encadrés de recherche
- Fournir des données spécifiques au portail ("NOM_APPLI","Incidents nformatiques") qui seront exploitées de manière spécifique par ce dernier.

```
$bean->civilite
$bean->code_retour
$bean->code_utilisateur_gestion
$bean->code_utilisateur_kportal
$bean->email
$bean->groupe
$bean->nom
$bean->prenom
$bean->profil
$bean->structure
```

Ce sont les informations utilisateur accessibles par l'application php.

```
    connecteurMgr::lireTemplate("haut");
    connecteurMgr::lireTemplate("bas");
```

Ces commandes permettent d'inclure le cadre graphique du portail.

NB: Si l'application est très solicitée avec des paramètres graphiques récurents, il peut être pertinent de stocker ces contenus graphiques en session pour ne pas solliciter le portail.