marketful fulfillment	Documentación del archivo: class-admin-wc-proxy.php
	Autor: Mauricio Alcala
	Fecha: 13 de agosto de 2018
	Versión: 1.01
	Autoriza: Adolfo Yanes

Descripción general:

Provee de diferentes métodos para poder crear las URL de los diferentes productos y acceder a ellos en WC.

Funciones PHP

```
/*************************************************************************************************************
 * @función MKF_ProductProxy @hereda
 *
 * @atributos
 * private static $instance = NULL;
 * private $plg_id;
 * private $version;
 *
 * Se definin las variables a utilizar
 *
 */
class MKF_ProductProxy extends MKF_DBCore
{

  private static $instance = NULL;
  private $plg_id;
  private $version;
  private $woocommerce;
  private $customer_key;
  private $secret_key;
  private $wc_api_v1;
  private $wc_api_v2;
  private $wc_api_v3;
/*************************************************************************************************************



  /*****************************************************************************
   * @función __construct(@string = (PLUGIN_GSNAME, @string = (PLUGIN_GVERSION)))
   * Manda a llamar al archivo autoload.php
   * Asigna despues los valores a los diferentes atributos.
   * Crea un nuevo objeto al cual le asigna los valores a base_uri y
   * verify.
   */
  public function __construct( $plg_id = PLUGIN_GSNAME, $version = PLUGIN_GVERSION)
  {

    require_once plugin_dir_path( __FILE__ ) . "extras/vendor/autoload.php";

    $this->plg_id  = $plg_id;
    $this->version = $version;
    $this->customer_key = '';
    $this->secret_key = '';
    $this->wc_api_v2 = 'wp-json/wc/v2';
    $this->wc_api_v3 = 'wc-api/v3';

    $this->woocommerce = new GuzzleHttp\Client([
      'base_uri' => esc_url(home_url()),
      'verify'   => true
    ]);

  }
  /*****************************************************************************/
```

```php
/*****************************************************************************
 * @función getInstance()
 * Es una función estatica que accede al objeto $instance y busca que sea nulo,
 * en cuyo caso creal el objeto.
 * En caso contrario retorna el objeto.
 */
public static function GetInstance()
{
  if ( is_null( self::$instance ) )
  {
    self::$instance = new self;
  }
  return self::$instance;
}
/*****************************************************************************/



/*****************************************************************************
 * @función getWC()
 *
 * Retorna el valor del atributo woocommerce.
 */
public function getWC()
{
  return $this->woocommerce;
}
/*****************************************************************************/



/*****************************************************************************
 * @función products_get_request(@string = (NULL))**
 *
 * Define el valor del $url_path que sera en caso de ser TRUE el product:id
 * en caso contrario sera ""
 * Despues devuelve el resultado de la funcion getWC( con los valores de la
 * api_v2 y el URL definido arriba.
 * Despues devuelve el valor de customer_key y de secret_key.
 */
public function products_get_request($product_id = null)
{
  $url_path = (empty($product_id) || is_null($product_id)) ? "" : "/{$product_id}";

  return $this->getWC()->request(
    'GET',
    "fullmarket/{$this->wc_api_v2}/products{$url_path}",
    [
      'auth' => [ $this->customer_key, $this->secret_key]
    ]
  );
}
/*****************************************************************************/



/*****************************************************************************
 * @Funciónproducts_put_request(@string = (NULL), @string = ([]))}
 *
 * La función primero define el valor del URL_PATH con el product_id o en
 * su defecto con valor "".
 * Despues regresa los valores de "auth" y "json"
 */
public function products_put_request($product_id = null, $metadata = [])
{
  $url_path = (empty($product_id) || is_null($product_id)) ? "" : "/{$product_id}";

  return $this->getWC()->request(
    'PUT',
    "fullmarket/{$this->wc_api_v2}/products{$url_path}",
    [
      'auth' => [ $this->customer_key, $this->secret_key],
      'json' => ['product' => $metadata]
```

```
      ]
    );
  }
  /**************************************************************************************/

  /*****************************************************************************
   * @función products_get_body (@string = (NULL), $force_to_array = (TRUE))
   *
   * Esta funcion crea una variable llamada $body que contiene el resultado de
   * la función llamada getBody()
   * Despues retorna en caso de que $force_to_array arroje verdadero
   * el valor del $body con json_decode() caso contrario, lo muestra sin json_decode
   */
  public function products_get_body($product_id = null, $force_to_array = true)
  {

    $body = $this->products_get_request($product_id)->getBody();

    return $force_to_array ? json_decode($body, true) : $body;
  }
}
/**************************************************************************************/
```