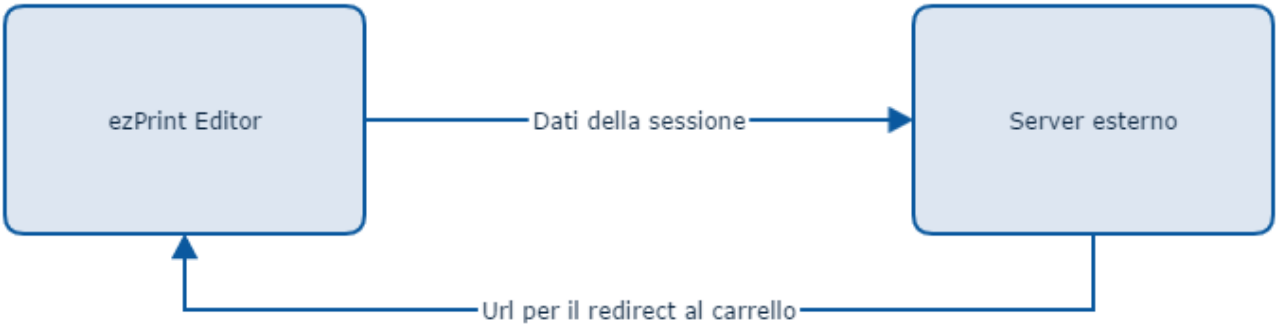
Gestione carrello esterno

Di seguito sono riportati gli esempi di utilizzo per il corretto funzionamento di un eventuale carrello esterno associato ad una sessione di editing.



Al momento dell'inizializzazione del carrello ezPrint effettua un post all'url presente all'interno della sessione inviando i seguenti dati:

Smart Editor:

- **sessionId:** Id della sessione.
- **editorType:** Tipo di editor utilizzato nella sessione.
- **productId:** Id del prodotto a cui è associato il progetto.
- **templateId:** Id del template a cui è associato il progetto, per le sessioni di photoEditor questo campo è sempre **0**.
- **projectId:** Id del progetto a cui fa riferimento la sessione, questo dato è necessario per richiedere il rendering del progetto.
- **userId:** Id dell'utente associato alla sessione.
- **pageCount:** Numero di page totali del prodotto esclusa la copertina.
- **coverPageCount:** Numero di pagine di copertina.
- **projectName:** Nome del progetto assegnato dall'utente.
- **languageCode:** Codice della lingua in cui è stato visualizzato l'editor in questa sessioni, utile per localizzare eventuali messaggi di errore da parte del carrello esterno.
- **firstPageThumbnailUrl:** Contiene la url dell'anteprima in bassa risoluzione della prima pagina del progetto. L'immagine fornita ha una dimensione sul lato più piccolo di 200px, l'altro lato viene calcolato in proporzione.
- **hash:** Checksum in formato MD5 dei dati inviati.

Photo Editor:

- **sessionId:** Id della sessione.
- **editorType:** Tipo di editor utilizzato nella sessione.
- **productId:** Id del prodotto a cui è associato il progetto.
- **projectId:** Id del progetto a cui fa riferimento la sessione, questo dato è necessario per richiedere il rendering del progetto.
- **userId:** Id dell'utente associato alla sessione.
- **estimationData:** Array di oggetti che rappresentano le foto presenti nel pacchetto, così composti:
    - **printFormat:** Nome del formato associato alla foto.
    - **printFormatId:** Id del formato associato alla foto.
    - **material:** Id del materiale associato alla foto.
    - **materialName:** Nome del materiale associato alla foto.
    - **copies:** Numero di copie acquistate.
- **languageCode:** Codice della lingua in cui è stato visualizzato l'editor in questa sessioni, utile per localizzare eventuali messaggi di errore da parte del carrello esterno.
- **hash:** Checksum in formato MD5 dei dati inviati.

Oltre ai parametri sopra riportati verranno aggiunti gli eventuali parametri presenti nel campo **shoppingCartExtraParams**.

Come risposta al post il sistema si aspetta una struttura json così composta:

- **shoppingCartRedirectUrl:** url a cui effettuare il redirect per procedere con l'acquisto.
- **errorMessage:** Eventuale messaggio di errore localizzato nella lingua corretta tramite il campo **languageCode**.

PHP

```
        header('Access-Control-Allow-Origin: *');

        $shoppingCartParams = $_POST;
        $shoppingCartOutput = array(
                'shoppingCartRedirectUrl' => '',
                'errorMessage' => ''
        );

        if (isset($shoppingCartParams['hash'])) {
                $dataHash = $shoppingCartParams['hash'];

                unset($shoppingCartParams['hash']);
                ksort($shoppingCartParams);

                if (md5(http_build_query($shoppingCartParams)) == $dataHash) {
                        $shoppingCartOutput['shoppingCartRedirectUrl'] = 'http://localhost/fakeCart.php';
                } else {
                        $shoppingCartOutput['errorMessage'] = ($shoppingCartParams['languageCode'] ==
'en_GB') ? 'Error Message' : 'Messaggio Errore';
                }
        } else {
                $shoppingCartOutput['errorMessage'] = 'Invalid Params';
        }

        echo json_encode($shoppingCartOutput);
```

Node.js

```
var express = require('express');
var bodyParser = require('body-parser');
var _ = require('lodash');
var param = require('node-qs-serialization').param;
var md5 = require('md5');

var app = express();

app.use(bodyParser.urlencoded({extended: true}));
app.use(bodyParser.json());

app.post('/fakeCart', function(req, res) {
    var shoppingCartParams = req.body;
        var shoppingCartOutput = {
                shoppingCartRedirectUrl: '',
                errorMessage: ''
        };

        if (shoppingCartParams.hash !== undefined) {
                var dataHash = shoppingCartParams.hash;

                shoppingCartParams = sortObjectByKey(_.omit(shoppingCartParams, 'hash'));

                if (md5(param(shoppingCartParams)) == dataHash) {
                        shoppingCartOutput.shoppingCartRedirectUrl = 'http://localhost/fakeCart.php';
                } else {
                        shoppingCartOutput.errorMessage = (shoppingCartParams.languageCode == 'en_GB') ?
'Error Message' : 'Messaggio Errore';
                }
        } else {
                shoppingCartOutput.errorMessage = 'Invalid Params';
        }

        res.status(200).json(shoppingCartOutput);
});

function sortObjectByKey(obj) {
        var keys = [];
        var sortedObj = {};
        for(var key in obj){
                if(obj.hasOwnProperty(key)){
                        keys.push(key);
                }
        }
        keys.sort();
        $.each(keys, function(i, key){
                sortedObj[key] = obj[key];
        });
    return sortedObj;
}

app.listen(80);
```

.NET (C#)

```
SortedDictionary<string, object> shoppingCartParams = GetPostParametersSomehow();
Dictionary<string, string> shoppingCartOutput = new Dictionary<string, string>();

shoppingCartOutput["shoppingCartRedirectUrl"] = " ";
shoppingCartOutput["errorMessage"] = " ";

if (shoppingCartParams["hash"] != null)
{
        int dataHash = (int)shoppingCartParams["hash"];

    shoppingCartParams.Remove("hash");
    int hash = ToQueryString(shoppingCartParams).GetHashCode();
    if (hash == dataHash)
    {
                shoppingCartOutput["shoppingCartRedirectUrl"] = "http://localhost/fakeCart.aspx";
    }
    else
    {
                shoppingCartOutput["errorMessage"] = (shoppingCartParams["languageCode"].ToString() ==
"en_GB") ? "Error Message" : "Messaggio Errore";
    }
}
else
{
        shoppingCartOutput["errorMessage"] = "Invalid Params";
}

public static string ToQueryString(SortedDictionary<string, object> source)
{
        return string.Join("&", source.Select(kvp => string.Format("{0}={1}", kvp.Key, kvp.Value.
ToString())));
}
```