



Technical Documentation

Document ID:5452
Document Version:1.0
Author: Surabhi
Kuruba Bhavani

Creating Module:

In this project we have created a module called "Myproject" in the "Project" namespace. The path of the module is -app/code/Project/Myproject

Below steps are used for creating a module-

- ◆ First we create registration.php for registering the module.

```
You, 2 weeks ago • First commit ...  
<?php  
/**  
 * Copyright © Magento. All rights reserved.  
 * See COPYING.txt for license details.  
 */  
\Magento\Framework\Component\ComponentRegistrar::register(  
\Magento\Framework\Component\ComponentRegistrar::MODULE, 'Project_Myproject', __DIR__  
);
```

- ◆ Then we have to create module.xml inside etc folder -etc/module.xml

```
<?xml version="1.0"?>  
<!--  
/**  
 * Copyright © Magento, Inc. All rights reserved.  
 * See COPYING.txt for license details.  
 */  
-->  
<config xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="urn:magento:framework:Module/etc/module.xsd">  
    <module name="Project_Myproject">  
        <sequence>  
            <module name="Magento_Sales"/>  
            <module name="Magento_Quote"/>  
        </sequence>  
    </module>  
</config>
```

- ◆ Then we have to create the router for that in the we have to create router.xml in the etc folder-etc/frontend/routes.xml

```
project > etc > frontend > routes.xml  
<?xml version="1.0"?>  
<config xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="urn:magento:framework:App/etc/routes.xsd">  
    <router id="standard">  
        <route frontName="helloworld" id="helloworld">  
            <module name="Project_Myproject"/>  
        </route>  
    </router>  
</config>
```

- ◆ After routes.xml we have to create controller for controlling the routes inside the Controller folder..i.e. Controller/Index/Index.php

```

<?php
namespace Project\Myproject\Controller\Index;

use \Magento\Framework\App\Action\HttpGetActionInterface;
use \Magento\Framework\View\Result\PageFactory;

class Index implements HttpGetActionInterface
{
    /**
     * @var \Magento\Framework\View\Result\PageFactory
     */
    protected $resultPageFactory;

    /**
     * @param PageFactory $resultPageFactory
     */
    public function __construct(PageFactory $resultPageFactory)
    {
        $this->resultPageFactory = $resultPageFactory;
    }

    /**
     * Prints the information
     * @return Page
     */
    public function execute()
    {
        return $this->resultPageFactory->create();
    }
}

```

- ◆ Then we created the layout file. Layout files are responsible for defining the structure and content of page and blocks.
I.e. app/code/vendor/modulename/view/frontend/layout

```

<?xml version="1.0"?>
<page xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" layout="2columns-left" xsi:noNamespaceSchemaLocation="urn:magento:framework:
<head>
    <title>index</title>
</head>
<body>
    <referenceContainer name="content">
        <block class="Project\Myproject\Block\Article"
            name="articleData"
            template="Project_Myproject::view.phtml" />
    </referenceContainer>
</body>
</page>

```

- ◆ After we have created the view.phtml file inside the view/frontend/templates/view.phtml

```

<h2><?php echo 'helloworld'; ?></h2>
<h2><?php echo $block->getArticles(); ?></h2>

```

- ◆ After that we have to create article.php file. It is responsible for rendering specific part of a page, such as a product list, a navigation menu, or a search bar. Here is the path: app/code/Project/Myproject/Block/Article.php

```

<?php
namespace Project\Myproject\Block;

use \Magento\Framework\View\Element\Template;

class Article extends Template
{
    /**
     * Constructor
     *
     * @param Context $context
     * @param array $data
     */
    public function __construct(
        \Magento\Backend\Block\Template\Context $context,
        array $data = []
    ){
        parent::__construct($context, $data);
    }

    /**
     * @return Post[]
     */
    public function getArticles()
    {
        return 'getArticles function of the Block class called successfully';
    }
}
?>

```

Creating admin theme:

We have to create new theme directory. Here is the path: app/design/adminhtml directory and create a new directory for our theme.

```
<?xml version="1.0"?><theme xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="urn:magento:framework:  
<title>Embitel Admin Section</title>  
<parent>Magento/backend</parent>  
</theme>
```

Registration.php

path: app/design/adminhtml/Project/Admintheme/Registration.php

```
<?php  
\Magento\Framework\Component\ComponentRegistrar::register(  
\Magento\Framework\Component\ComponentRegistrar::THEME,  
'adminhtml/Project/Admintheme',  
    __DIR__  
);
```

Create one logo for admintheme:

Path: app/design/adminhtml/Project/Admintheme/Web/Images/logo.png

- ◆ Create di.xml in app/code/Project/Admintheme/etc/di.xml

```
<?xml version="1.0"?>  
<config xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="urn:magento:framework:ObjectManager/etc/config.xsd">  
<type name="Magento\Theme\Model\View\Design">  
<arguments>  
<argument name="themes" xsi:type="array">  
<item name="adminhtml" xsi:type="string">Project/Admintheme</item>  
</argument>  
</arguments>  
</type>  
</config>
```

Create admin_login.xml file

Here is the path: app/code/Project/Admintheme/view/adminhtml/layout/adminlogin.xml

It is used to modify the layout of the Magento admin login page. This file is used to modify the layout of the Magento_Backend module and can be overridden in custom admin themes.

```
You, 6 hours ago | 1 author (You)
<?xml version="1.0"?>      You, 6 hours ago • partially done code
<page xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" layout="admin-login" xsi:noNamespaceSchemaLocation="urn:magento:framework:View
<update handle="styles" />
<body>
<referenceBlock name="logo">
<arguments>
<argument name="logo_image_src" xsi:type="string">images/logo.png</argument>
</arguments>
</referenceBlock>
</body>
</page>
```

Adding default.xml file:

This layout file is used to customize the admin header section by adding a custom logo to the top left corner of the page.

```
You, 6 hours ago | 1 author (You)
<?xml version="1.0"?>
<page xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" layout="admin-1column" xsi:noNamespaceSchemaLocation="urn:magento:framework:View
<body>
<referenceContainer name="header">
<block class="Magento\Backend\Block\Page\Header" name="logo" before="-">
<arguments>
<argument name="show_part" xsi:type="string">logo</argument>
<argument name="edition" translate="true" xsi:type="string">Community Edition</argument>
<argument name="logo_image_src" xsi:type="string">images/logo.png</argument>
</arguments>
</block>
</referenceContainer>
</body>
</page>
```


Overriding the phtml file:

Overriding the core module from the Customer module, it exists in-vendor/magento/module-sales/view/adminhtml/templates/order/invoice/create/items.php from this path we are overriding the customer module.

Path:app/design/adminhtml/Project/Admintheme/magento-sales/templates/order/invoice/create/items.phtml

Using this we are rendering an invoice creation form in the magento2 admin panel. The form displays a list of items to be invoiced with details such as product name ,price,quantity etc.

```
> Project > Admintheme > Magento_Sales > templates > order > invoice > create > items.phtml
?php
**
* Copyright © Magento, Inc. All rights reserved.
* See COPYING.txt for license details.
*/

** @var \Project\Myproject\vendor\magento\module-sales\view\Adminhtml\templates\Order\Invoice\Create\Items $block */
** @var \Magento\Framework\View\Helper\SecureHtmlRenderer $secureRenderer */
>
section class="admin__page-section">
    <div class="admin__page-section-title">
        <?php $itemsGridLabel = $block->getForcedShipmentCreate() ? 'Items to Invoice and Ship' : 'Items to Invoice';?>
        <span class="title"><?= $block->escapeHtml(__('%1', $itemsGridLabel)) ?></span>
    </div>
    <div class="admin__page-section-content grid">
        <div class="admin__table-wrapper">
            <table class="data-table admin__table-primary order-invoice-tables">
                <thead>
                    <tr class="headings">
                        <th class="col-product"><span><?= $block->escapeHtml(__('Product')) ?></span></th>
                        <th class="col-price"><span><?= $block->escapeHtml(__('Price')) ?></span></th>
                        <th class="col-ordered-qty"><span><?= $block->escapeHtml(__('Qty')) ?></span></th>
                        <th class="col-qty-invoice"><span><?= $block->escapeHtml(__('Qty to Invoice')) ?></span></th>
                        <th class="col-subtotal"><span><?= $block->escapeHtml(__('Subtotal')) ?></span></th>
                        <th class="col-tax"><span><?= $block->escapeHtml(__('Tax Amount')) ?></span></th>
                        <th class="col-discount"><span><?= $block->escapeHtml(__('Discount Amount')) ?></span></th>
                        <th class="col-total last"><span><?= $block->escapeHtml(__('Row Total')) ?></span></th>
                        <th class="col-barcodenumber"><span><?= $block->escapeHtml(__('Barcodenumber')) ?></span></th>
                    </tr>
                </thead>
                <?php if ($block->canEditQty()): ?>
                    <tfoot>
                        <tr>
                            <td colspan="3">&nbsp;</td>
                            <td><?= $block->getUpdateButtonHtml() ?></td>
                            <td colspan="4">&nbsp;</td>
                        </tr>
                    </tfoot>
                </?php>
            </table>
        </div>
    </div>
</section>
```

Overriding the core module from the Customer module, it exists in-vendor/magento/module-sales/view/adminhtml/templates/order/invoice/create/items.php from this path we are overriding the customer module.
Path: app/design/adminhtml/Project/Admintheme/magento-sales/templates/order/invoice/create/items/renderer/default.phtml

It is used to add a column for entering the barcode number with one input for each item in the order.

```
minhtml > Project > Admintheme > Magento_Sales > templates > order > invoice > create > items > renderer > default.phtml
1 <td class="col-price">
2     <?= $block->getColumnHtml($_item, 'price') ?>
3 </td>
4 <td class="col-qty"><?= $block->getColumnHtml($_item, 'qty') ?></td>
5 <td class="col-qty-invoice">
6 <?php if ($block->canEditQty()) : ?> You, 7 hours ago • partially done code
7     <input type="text" class="input-text admin__control-text qty-input"
8         name="invoice[items][<?= (int) $_item->getOrderId() ?>]"
9         value="<?= (float) $_item->getQty() ?>"/>
10 <?php else : ?>
11     <?= (float) $_item->getQty() ?>
12 <?php endif; ?>
13 </td>
14 <td class="col-subtotal">
15     <?= $block->getColumnHtml($_item, 'subtotal') ?>
16 </td>
17 <td class="col-tax"><?= /* @noEscape */ $block->displayPriceAttribute('tax_amount') ?></td>
18 <td class="col-discount"><?= /* @noEscape */ $block->displayPriceAttribute('discount_amount') ?></td>
19 <td class="col-total last">
20     <?= $block->getColumnHtml($_item, 'total') ?>
21 </td>
22 <td class="col-barcodenumber">
23
24 <?php
25 $qty = (int)$item->getQty();
26 for($i=0; $i<$qty; $i++){
27     ?>
28
29     <input type="text" name="invoice[items][barcode][<?= (int) $_item->getOrderId() ?>][<?php echo $i?>]" maxlength="13" id="pin" pa
30 >
31 <?php } ?>
32
33 </td>
34
```


Creating Model for connecting the Database:

Overriding the php file:

Overriding the core module from the Customer module, it exists in-vendor/magento/module-sales/model/service/invoiceservice.php from this path we are overriding the customer module.

Path: app/code/Project/Myproject/model/service/invoiceservice.php

It is used to prepare an invoice for an order.

```
Service > InvoiceService.php
* @throws \Exception
*/
public function prepareInvoice(
    Order $order,
    array $orderItemsQtyToInvoice = []
): InvoiceInterface {
    $totalQty = 0;
    $invoice = $this->orderConverter->toInvoice($order);
    $preparedItemsQty = $this->prepareItemsQty($order, $orderItemsQtyToInvoice);

    foreach ($order->getAllItems() as $orderItem) {
        if (!$this->canInvoiceItem($orderItem, $preparedItemsQty)) {
            continue;
        }

        if (isset($preparedItemsQty[$orderItem->getId()])) {
            $qty = $preparedItemsQty[$orderItem->getId()];
        } elseif ($orderItem->isDummy()) {
            $qty = $orderItem->getQtyOrdered() ? $orderItem->getQtyOrdered() : 1;
        } elseif (empty($orderItemsQtyToInvoice)) {
            $qty = $orderItem->getQtyToInvoice();
        } else {
            $qty = 0;
        }

        $invoiceItem = $this->orderConverter->itemToInvoiceItem($orderItem);
        $this->setInvoiceItemQuantity($invoiceItem, (float) $qty);
        if (isset($preparedItemsQty["barcode"]) && isset($preparedItemsQty["barcode"][$orderItem->getId()])) {
            $barcode = $preparedItemsQty["barcode"][$orderItem->getId()];
            if (is_array($barcode)) {
                $barcode = implode(' ', $barcode);
            }
            $invoiceItem->setBarcodeNumber($barcode);
        }
        $invoice->addItem($invoiceItem);
        $totalQty += $qty;
    }
}
```

Overriding the php file:

Creating Model for connecting the Database:

Overriding the core module from the Customer module, it exists in-vendor/magento/module-sales/model/service/item.php from this path we are overriding the customer module.

Path:app/code/Project/Myproject/model/service/item.php

```
> Order > Item.php
<?php

namespace Project\Myproject\Model\Order;

class Item extends \Magento\Sales\Model\Order\Item
{
    public function getMrp()
    {
        return $this->getData("mrp");
    }

    public function setMrp($sku)
    {
        return $this->setData("mrp", $sku);
    }
}
```

Create db_schema.xml:

Path:app/code/project/myproject/etc/db_schema.xml

Adding the two fields that is Barcode and MRP in the sales_order_item and sales_invoice_item

```
> Myproject > etc > db_schema.xml
<?xml version="1.0"?>
<!--
/**
 * Copyright © Magento, Inc. All rights reserved.
 * See COPYING.txt for license details.
 */
-->
<schema xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:noNamespaceSchemaLocation="urn:magento:framework:Setup/Declaration/Schema/etc/schema.xsd">

    <table name="sales_order_item" resource="sales" engine="innodb" comment="Sales Flat Order Item">
        <column xsi:type="decimal" name="mrp" scale="4" precision="20" unsigned="false" nullable="true"
                comment="MRP"/>
        <column xsi:type="int" name="barcodenumber" unsigned="false" nullable="true"
                comment="Barcode"/>
    </table>

    <table name="sales_invoice_item" resource="sales" engine="innodb" comment="Sales Flat Invoice Item">
        <column xsi:type="decimal" name="mrp" scale="4" precision="20" unsigned="false" nullable="true"
                comment="MRP"/>
        <column xsi:type="int" name="barcodenumber" unsigned="false" nullable="true"
                comment="Barcode"/>
    </table>

</schema>
```

Adding Observer:

Path:app/code/project/myproject/Observer/salesdata.php

```
> Myproject > Observer > SalesData.php
<?php
namespace Project\Myproject\Observer;

use Magento\Framework\Event\ObserverInterface;
use Magento\Framework\Event\Observer;
use Magento\Quote\Model\Quote\Item as QuoteItem;

class SalesData implements ObserverInterface
{
    public function execute(Observer $observer)
    {
        /** @var QuoteItem $quoteItem */
        $quoteItem = $observer->getEvent()->getData('quote_item');
        $product = $observer->getEvent()->getData('product');

        $customAttributeValue = $product->getData('mrp');
        $quoteItem->setData('mrp', $customAttributeValue);
    }
}
```

Events.xml:

sales_quote_item_set_product is the event name of the quote_item table

Path:app/code/Project/Myproject/etc/events.xml

```
> Myproject > etc > events.xml
<?xml version="1.0"?>
<config xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:noNamespaceSchemaLocation="urn:magento:framework:Event/etc/events.xsd">
    <event name="sales_quote_item_set_product">
        <observer name="SalesData" instance="Project\Myproject\Observer\SalesData"/>
    </event>
</config>
```

Create di.xml for overriding the php files

Path:app/code/project/myproject/etc/di.xml

```
> Myproject > etc > di.xml
<?xml version="1.0"?>
<config xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="urn:magento:framework:ObjectManager/etc/config.xsd">
    <preference for="Magento\Sales\Model\Service\InvoiceService" type="Project\Myproject\Model\Service\InvoiceService"/>
    <preference for="Magento\Sales\Model\Order\Item" type="Project\Myproject\Model\Order\Item"/>
</config>
```

Create catalog_attributes.xml file:

It is used to represents the attributes that are going to be copied from the product to quote_item table

Path:app/code/project/myproject/etc/catalog_attributes.xml

```
> Myproject > etc > catalog_attributes.xml
<?xml version="1.0"?>
<!--
/**
 * Copyright © Magento, Inc. All rights reserved.
 * See COPYING.txt for license details.
 */
-->
<config xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="urn:magento:module:Magento_Catalog/etc/catalog_attributes.xsd">
    <group name="quote item">
        <attribute name="sku"/>
        <attribute name="type_id"/>
        <attribute name="name"/>
        <attribute name="status"/>
        <attribute name="visibility"/>
        <attribute name="price"/>
        <attribute name="weight"/>
        <attribute name="url_path"/>
        <attribute name="url_key"/>
        <attribute name="thumbnail"/>
        <attribute name="small_image"/>
        <attribute name="tax_class_id"/>
        <attribute name="special_from_date"/>
        <attribute name="special_to_date"/>
        <attribute name="special_price"/>
        <attribute name="cost"/>
        <attribute name="gift_message_available"/>
    </group>
</config>
```

Creation of fieldset.xml file:

Fieldset defined in this file is used to map the MRP field from a quote item to an order item and from an order item to an invoice item.

Path:app/code/project/myproject/etc/fieldset.xml

```
> Myproject > etc > fieldset.xml
<?xml version="1.0"?>
<!--
/**
 * Copyright © Magento, Inc. All rights reserved.
 * See COPYING.txt for license details.
 */
-->
<config xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:noNamespaceSchemaLocation="urn:magento:framework:DataObject/etc/fieldset.xsd">
    <scope id="global">
        <fieldset id="quote_convert_item">
            <field name="mrp">
                <aspect name="to_order_item" />
            </field>
        </fieldset>
        <fieldset id="sales_convert_order_item">
            <field name="mrp">
                <aspect name="to_invoice_item" />
            </field>
        </fieldset>
    </scope>
</config>
```


