

Useful Tips for becoming Quality magento developer

Use dependency injection:	Inject dependencies into your classes for better testability and maintainability
Follow Magento's coding standards:	Adhere to coding conventions for consistency and readability
Test your code thoroughly:	Write unit and integration tests to ensure code quality and prevent regressions
Use Magento's built-in features:	Take advantage of Magento's features like layout XML, UI components, and API endpoints before creating custom extensions
Utilize the Magento community:	Engage with the active Magento community for help, support, and resources

Getting the current customer session

```
$objectManager = \Magento\Framework\App\ObjectManager::getInstance();
$customerSession = $objectManager->get(\Magento\Customer\Model\Session::class);
$customerId = $customerSession->getCustomerId();
```

Retrieving a product by ID

```
$objectManager = \Magento\Framework\App\ObjectManager::getInstance();
$productRepository = $objectManager->get(\Magento\Catalog\Api\ProductRepositoryInterface::class);
$product = $productRepository->getById($productId);
```

Sending an email programmatically

```
$objectManager = \Magento\Framework\App\ObjectManager::getInstance();
$transportBuilder = $objectManager->get(\Magento\Framework\Mail\TemplateTransportBuilderInterface::class);
$transport = $transportBuilder->setTemplateIdentifier('my_email_template')
    ->setTemplateOptions(['area' => 'frontend', 'store' => $storeId])
    ->setTemplateVars(['data' => $myData])
    ->setFrom('sender@example.com')
    ->addTo('recipient@example.com')
    ->getTransport();
$transport->sendMessage();
```

Adding a custom layout block

```
<referenceContainer name="content">
    <block class="Magento\Framework\View\Element\Template" name="my_custom_block"
    template="Magento_Template::my-custom-block.phtml" />
</referenceContainer>
```

Creating a custom UI component

```
define([
    'uiComponent'
], function (Component) {
```



Creating a custom UI component (cont)

```
> 'use strict';
return Component.extend({
    // Your component logic here
});
});
```

Using a custom module's configuration

```
$scopeConfig = $objectManager->get(\Magento\Framework\App\Config\ScopeConfigInterface::class);
$myModuleConfigValue = $scopeConfig->getValue('my/module/config/path', \Magento\Store -
re \Module \ScopeInterface::SCOPE_STORE);
```

Logging messages for debugging

```
$logger = $objectManager->get(\Psr\Log\LoggerInterface::class);
$logger->debug('My debug message');
$logger->info('My info message');
$logger->error('My error message');
```

Creating a custom admin grid

```
<listing xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="urn:magento:module:Magento_Ui:etc/ui_configuration.xsd">
</listing>
```

Creating a custom API endpoint

```
// In your module's webapi.xml
<route url="/V1/my-endpoint" method="GET">
    <service class="My \Module \Api \MyEndpointInterface" method="getList"/>
    <resources>
        <resource ref="anonymous"/>
    </resources>
</route>
```



By Bhavi
cheatography.com/bhavi/

Not published yet.
Last updated 19th January, 2024.
Page 3 of 3.

Sponsored by CrosswordCheats.com
Learn to solve cryptic crosswords!
<http://crosswordcheats.com>

