embitel
delivered with passion

OPTIMIZING MAGENTO 2 WEBSITE PERFORMANCE USING MAGEPACK BUNDLING

Table of Contents

1. Introduction

In the world of ecommerce, website optimization and page load speed are deal breakers that are correlated with your e-store conversion rates and user experience rankings. In fact, Google considers site speed as a critical ranking factor for its search and display ads. This is why a lot of eCommerce store owners and web developers fret over a slow loading page speed.

Talking about ecommerce stores running on the Magento platform, there is a very popular website optimization technique vouched by every Magento developer. Yes, we are talking about Advanced Javascript Bundling.

Advanced Javascript Bundling enhances page load speed by reducing the number of server requests along with the bundle size required to load each page in the browser.

In this whitepaper, we will discuss about optimizing the page load speeds of Magento based e-commerce stores using Magepack Bundling, an open source implementation using Javascript Bundling.

We will first venture to understand all about the Advanced Javascript bundling concept . We will then go on to explore what makes MagePack , the go-to solution for Magento ecommerce website optimization.

2. Understanding JavaScript Bundling with MagePack

JavaScript bundling is an optimization technique you can use to reduce the number of server requests for JavaScript files. In this technique, you can decide where to load a page specific bundle and thus keep the webpages more dynamic and faster . To reduce the number and size of requested assets for each page loaded in the browser, MagePack will build bundles with which each page in our website will need to download a common bundle and a page specific bundle for each & every page accessed. For better performance we need to enable js minify & merge. It will work with both JS minification & merge.

2a. Why Default Magento JS Bundler is not always preferred?

Magento is a resource-heavy ecommerce platform which will require the right set of resources and configurations for optimum performance.

Magento bundling will reduce the number of connections per page, but for each page request it will load all the bundles. Although, the page requested might depend on one or two of the bundles, the browser will load all the bundles synchronously. This means the page will not be rendered, until all the bundles has been synchronously downloaded by the browser. This In turn will have an adverse effect on page load speed, increasing the bounce rates of the website, and impacting the user experience & overall performance of the site.

Even in the attached link, it is described that creating a huge js file will affect the performance. For reference: https://github.com/magento/magento2/issues/4506

3. Why Magepack?

There are various proven ways available to reduce the impact of the page renderblocking components like Javascript , one of them being - bundling using Magepack. This method of Javascript bundling using Magepack is a remarkable attempt in optimizing the page load speed and making Magento frontend faster like never before.

Did You Know?

- Sites with Magepack shows 91% Lighthouse Performance Score which is indicative of a website delivering great user experience.
- It works seamlessly with Magento's Merge JS and Minify JS functionalities (enabled)
- Is proven to trim down total page load time by 75%
- Known to reduce the number of Javascript File sizes, from 177 to 3,by 98%. ·
- Advanced versions of Magepack (2.x) is compatible with major Magento optimizations

4. Setting Up and Installing Magepack :

In the subsequent sections, let's explore how to configure Magepack on Magento 2 Enterprise Commerce Cloud and further versions.

4a. Magepack Setup on Magento 2 Enterprise Commerce Cloud

We need to configure the Cloud deployment process to run the Magepack command in the build hook, so for this we need to setup npm & magepack on the cloud version first.

For installing latest version of npm & magepack, please add the below content which will instruct how we can upgrade npm to the latest version & install magepack on cloud server. Below is the code with npm version 14

---Add the below content in .magento.app.yaml -----

```
unset NPM_CONFIG_PREFIX
curl -o- https://raw.githubusercontent.com/creationix/nvm/v0.33.8/install.sh | dash
export NVM_DIR="$HOME/.nvm"
[ -s "$NVM_DIR/nvm.sh" ] && . "$NVM_DIR/nvm.sh"
nvm current
nvm install 14.4.0
npm install -g magepack
php ./vendor/bin/ece-tools build:generate
magepack bundle
php ./vendor/bin/ece-tools build:transfer
```

After making the above changes , we need to generate magepack.config.js file and we need to have magepack.config in local setup repo. Next Section will describe how to setup and install Magepack for Magento 2.3.x and upper on local.

NOTE: Magepack commands only work during the hook build, if we try to use ssh to the cloud environment after the deployment, it won't work. So commit magepack.config.js to the repo.

If you've multi language setup on cloud, we need to do Static Content Deployment (SCD) on build as well, so we should modify the value of SCD_STRATEGY & add SCD_MATRIX for this.

Add or Modify the following lines in .magento.env.yaml

- **SCD_STRATEGY:"standard" under build parameter &-**

- **SCD_MATRIX:**

```
"Magento/backend":
language:
  - en_US
  - pt_BR
  - es_MX
"Vendor/theme":
language:
  - pt_BR
"Vendor/theme2":
language:
   - es_MX
```

After installing Magepack on local, just commit the below files .magento.app.yaml,composer.json, composer.lock,magepack.config.js file.

If you're pushing changes for multi language site then make the above changes mentioned for multi language & push the .maganro.env.yaml file along with the other files.

To enable Magepack JavaScript bundling from the command line on the cloud server or in production mode use the following command:

php bin/magento config:set dev/js/enable_magepack_js_bundling 1
Note that you should also enable the other JavaScript options for better performance which includes minify, merge JavaScript files and move js code to the bottom of the page – but don't enable the default bundling!!

MagePack JavaScript bundling should now be enabled on cloud. To check whether it's working, go to any of Magento 2 product pages and look at the source code. Next, do a search for "bundle" and you should see the magepack JavaScript bundles files. For home page or any cms page bundle file name should be bundle-cms.js and bundle-common.js file should be present in all the pages.

4b. Setup and install Magepack for Magento 2.3.x and upper

Make sure there are no console errors before implementing the Magepack Advanced bundling. If any, ensure that you resolve those issues prior to the Advanced bundling implementation

1. **You require Node.js version 10 or higher installed. For installing node ver 10 or higher(below are given the steps for installing version 14)**

- install Node.js 14 on Ubuntu / Debian by installing the required repository : *curl -sL https://deb.nodesource.com/setup_14.x | sudo bash -*
- Install of node js version 14 : *sudo apt install nodejs*
- Verify node js version with command : *node -v & nodejs -v*

2. Install NodeJS MagePack on your local: npm install -g magepack

Sometimes Ubuntu will need some more dependencies before MagePack will install, so run the below command for installing the dependencies:
(if you face any issue while running the above command on Ubuntu then run the below command,otherwise you can skip this)
sudo apt-get install gconf-service libasound2 libatk1.0-0 libatk-bridge2.0-0 libc6 libcairo2libcups2 libdbus-1-3 libexpat1 libfontconfig1 libgcc1 libgconf-2-4 libgdk-pixbuf2.0-0 libglib2.0- 0 libgtk-3-0 libnspr4 libpango-1.0-0 libpangocairo-1.0-0 libstdc++6 libx11-6 libx11-xcb1 libxcb1 libxcomposite1 libxcursor1 libxdamage1 libxext6 libxfixes3 libxi6 libxrandr2 libxren- der1 libxss1 libxtst6 ca-certificates fonts-liberation libappindicator1 libnss3 lsb-release xdg- utils wget

3. If you get error related to Puppeter for example: Error: Cannot find module 'puppeteer':

Puppeteer is a node library that provides a high-level API to control headless Chrome or Chromium over the DevTools Protocol. It can be configured to use full (non-headless) Chrome or Chromium.

For resolving this issue, install Puppeter in your project & run the below commands:

- *sudo npm i puppeteer --save*
- *sudo npm install -g magepack –unsafe-perm=true –allow-root*

Now Magepack will pull down Chromium browser to analyse our website (CMS, Category,Cart & Checkout pages)

4. **Need to install Magepack Magento 2 Module. For installing the module, run the below composer command:**

composer require creativestyle/magesuite-magepack

5. **Patches:**

For installing patches needs to run the below command & add the below data in composer.json as per mentioned according to Magento version. In root directory you need to create patches directory and copy the patches from the composer directory of this url (https://github.com/integer-net/magento2-requirejs-bundling) & paste into this directory

Command : composer require cweagans/composer-patches

- For Magento 2.3.3 and below, 7 patches are required: add the below content in composer.json

```
"extra": {
     "magento-force": "override",
     "composer-exit-on-patch-failure": true,
"patches": {
    "magento/magento2-base": {
```

```
    "[Performance] Fix missing shims and phtml files with mage-init directives
(https://github.com/magento/magento2/commit/db43c11c6830465b764ede
32abb7262258e5f574)": "patches/composer/M233/github-pr-4721-base.diff",

    "Refactor JavaScript mixins module
https://github.com/magento/magento2/pull/25587":
"patches/composer/M233/github-pr-25587-base.diff"
},
    "magento/module-braintree": {
"[Performance] Fix missing shims and phtml files with mage-init directives
(https://github.com/magento/magento2/commit/db43c11c6830465b764ede
32abb7262258e5f574)":
"patches/composer/M233/github-pr-4721-braintree.diff"
},
    "magento/module-catalog": {
"[Performance] Fix missing shims and phtml files with mage-init directives
(https://github.com/magento/magento2/commit/db43c11c6830465b764ede
32abb7262258e5f574)":
"patches/composer/M233/github-pr-4721-catalog.diff"
},
    "magento/module-customer": {
"[Performance] Fix missing shims and phtml files with mage-init directives
(https://github.com/magento/magento2/commit/db43c11c6830465b764ede
32abb7262258e5f574)":
"patches/composer/M233/github-pr-4721-customer.diff"
},
    "magento/module-msrp": {
"[Performance] Fix missing shims and phtml files with mage-init directives
(https://github.com/magento/magento2/commit/db43c11c6830465b764
ede32abb7262258e5f574)":
"patches/composer/M233/github-pr-4721-msrp.diff"
    },
    "magento/module-paypal": {
"[Performance] Fix missing shims and phtml files with mage-init directives
(https://github.com/magento/magento2/commit/db43c11c6830465b764
ede32abb7262258e5f574)":
"patches/composer/M233/github-pr-4721-paypal.diff"
    },
    "magento/module-theme": {
      "[Performance] Fix missing shims and phtml files with mage-init
       directives
(https://github.com/magento/magento2/commit/db43c11c6830465b764
ede32abb7262258e5f574)":
"patches/composer/M233/github-pr-4721-theme.diff",
    "fix_baler_jquery_cookie":
"https://gist.github.com/tdgroot/f95c398c565d9bbb83e0a650cdf67617/r
aw/69ee2d001ff509d25d1875743e417d914e20fd85/fix_baler_jquery_cook
ie.patch"
      }
    }
},
```

· *For Magento 2.3.4 and 2.3.5 1 patch is required*

```
"extra": {
   "magento-force": "override",
   "composer-exit-on-patch-failure": true,
   "patches": {
        "magento/magento2-base": {
           "Refactor JavaScript mixins module
https://github.com/magento/magento2/pull/25587":
"patches/composer/M234/github-pr-25587-base.diff"
        }
   }
},
```

· *For Magento 2.4.0 & 2.4.1 we don't need any patches.*

Run composer update, Magento 2 will be patched & Magepack module will be installed.

How to use Magepack

Usage: magepack [generate|bundle] <options...>

Options:

-v, --version : Output the current version.

-h, --help : Show this command summary.

Commands:

generate [options]: Generate optimization configuration based on given page URLs.

bundle [options]: Bundle JavaScript files using given configuration file.

Generating Bundler configuration

First we need to generate bundler configuration, against the existing working environment. We can do it on any machine with access to the target shop. Here main goal is to collect all of the RequireJS dependencies needed for a certain type of page layout. Currently, following bundles are prepared:

· Cms: Includes modules required by CMS pages.

· Category: Contains modules which are necessary for category pages.

· Product : Includes all the components required by product pages.

· Checkout : Comes with all modules necessary for the cart and checkout pages.

In addition, there is the common bundle js created by extracting all modules needed by each of above and loaded on every page

Running the generator

Run the below command for generating bundle files

magepack generate --cms-url="{{CMS_PAGE_URL}}"--category-url="{{CATEGORY_PAGE_URL}}" --product-url="{{PRODUCT_PAGE_URL}}"

There are 3 required options we need to pass:

• cms-url- URL to one of CMS pages.

• Category-url-URL to one of the category pages.

• product-url- URL to one of the product pages.

magepack generate --cms-url="{{CMS_PAGE_URL}}"--category-url="{{CATEGORY_PAGE_URL}}" --product-url="{{PRODUCT_PAGE_URL}}"

Note: Magepack will use the above home page, category page, product page url, add that product to the cart and visit both cart and checkout pages to collect all dependencies.

Running the above command will generate **magepack.config.**js file, where we'll find each of the prepared bundles with the list of modules that was included in them.

Bundling Magepack

Once bundler configuration has been generated , the next step would be to trigger the actual optimization. If you're running the website in developer mode then run static deploy command first(php bin/magento s:s:d -f)by running the following in root directory:

magepack bundle

The above command will iterate over each deployed locale (excluding Magento/blank) and prepare bundles for each of them.

Enabling Magepack

Once Magepack Magento moduleis installed Enable Magepack it using below options :

- **Developer Mode:** If you're running on developer mode, then go to Admin panel under Stores->Configuration->Advanced->Developer and enable it.
- **Production Mode:** If you're running on production mode, then run the below command on CLI

php bin/magento config:set dev/js/enable_magepack_js_bundling 1

Now clear the cache using following command: ***php bin/magento cache:clean***

You should enable all Magento's performance optimizations (except JavaScript bundling of course) for better results. Javascript merge & minify should be enabled but Javascript bundling should be disabled.

TROUBLESHOOTING

If you are encountering any console error due to the usage of the mentioned files & pr if you want any js that should not be bundled , you can remove that js name from magepack.config.js and run **magepack bundle** command.

For eg: If you are encountering the console error due to the usage of the slick.js file, then you can safely remove them from the bundle. You need to tweak a bit on the magepack.config.js file to ignore those files from being bundled.

Replace this:

```
{
  name: 'common',
  modules: {
    slick.js
  }
}
```

With:

```
{
  name: 'common',
  modules: {
    //slick.js
  }
}
```

Then run the "magepack bundle" command from the terminal.

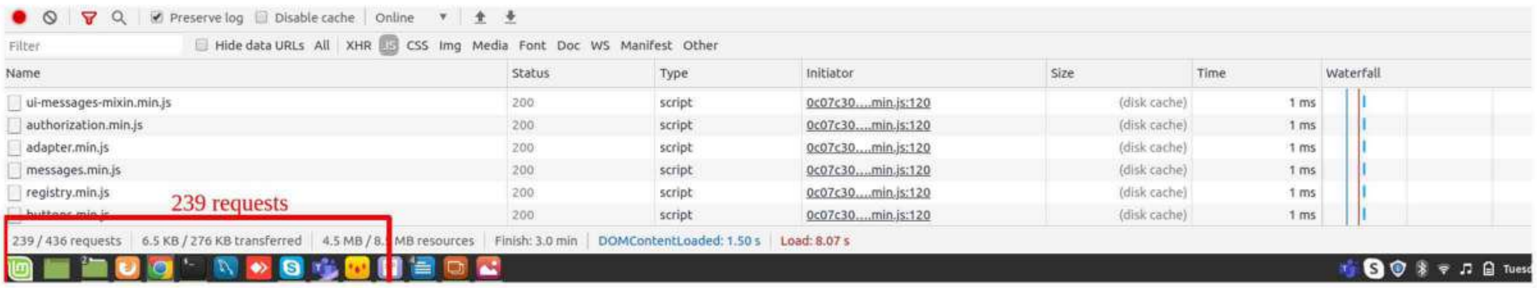
4. Result : Magepack JS Number & Size.

I've tested the site with both the scenarios before & after configuring magepack on the cloud server and attached the results below.

4 a. NUMBER OF REQUESTS

If you check the screenshot below, the number of requests , before configuring magepack, as displayed on the dashboard were 239. The number of requests came down to 20, after the Magepack configuration.

Before:



Preserve log | Disable cache | Online

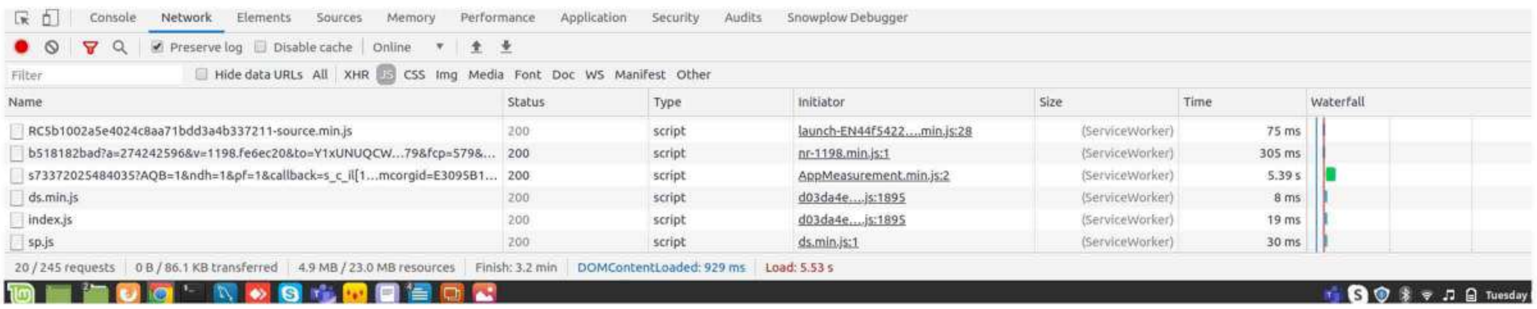
Filter | Hide data URLs | All | XHR | JS | CSS | Img | Media | Font | Doc | WS | Manifest | Other

Name	Status	Type	Initiator	Size	Time
ui-messages-mixin.min.js	200	script	0c07c30....min.js:120	(disk cache)	1 ms
authorization.min.js	200	script	0c07c30....min.js:120	(disk cache)	1 ms
adapter.min.js	200	script	0c07c30....min.js:120	(disk cache)	1 ms
messages.min.js	200	script	0c07c30....min.js:120	(disk cache)	1 ms
registry.min.js	200	script	0c07c30....min.js:120	(disk cache)	1 ms
buttons.min.js	200	script	0c07c30....min.js:120	(disk cache)	1 ms

239 requests

239 / 436 requests | 6.5 KB / 276 KB transferred | 4.5 MB / 8. MB resources | Finish: 3.0 min | DOMContentLoaded: 1.50 s | Load: 8.07 s

After:



Console | Network | Elements | Sources | Memory | Performance | Application | Security | Audits | Snowplow Debugger

Preserve log | Disable cache | Online

Filter | Hide data URLs | All | XHR | JS | CSS | Img | Media | Font | Doc | WS | Manifest | Other

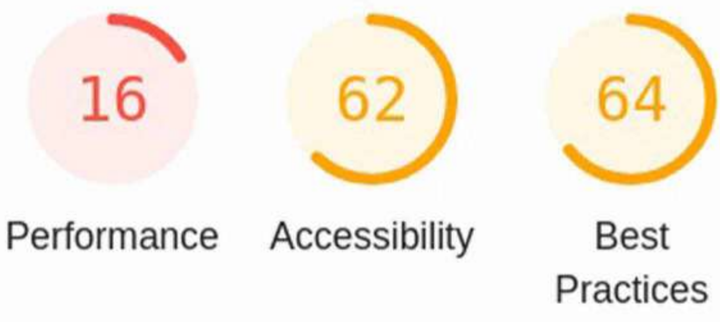
Name	Status	Type	Initiator	Size	Time
RC5b1002a5e4024c8aa71bdd3a4b337211-source.min.js	200	script	launch-EN44f5422....min.js:28	(ServiceWorker)	75 ms
b518182bad?a=274242596&v=1198.fe6ec20&to=Y1xUNUQCW...79&fcp=579&...	200	script	nr-1198.min.js:1	(ServiceWorker)	305 ms
s73372025484035?AQB=1&ndh=1&pf=1&callback=s_c_il[1...mcorgid=E3095B1...	200	script	AppMeasurement.min.js:2	(ServiceWorker)	5.39 s
ds.min.js	200	script	d03da4e....js:1895	(ServiceWorker)	8 ms
index.js	200	script	d03da4e....js:1895	(ServiceWorker)	19 ms
sp.js	200	script	ds.min.js:1	(ServiceWorker)	30 ms

20 / 245 requests | 0 B / 86.1 KB transferred | 4.9 MB / 23.0 MB resources | Finish: 3.2 min | DOMContentLoaded: 929 ms | Load: 5.53 s

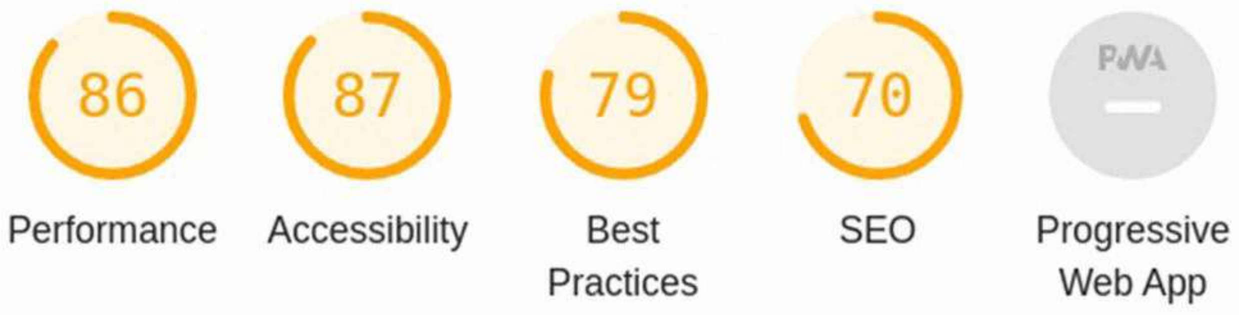
4 b. Performance using Light house:

Before configuring magepack, site performance score was 16 but now it enhances to 86. For your reference, we've attached screenshot.

Before:



After:



5. Conclusion:

Magepack offers a seamless solution for webpage speed optimization and improving the page performance of Magento based ecommerce stores, in the process. Magepack achieves this by applying the Advanced Javascript bundling methods to reduce the number of server requests made by the webpage. So, if you are struggling with slow loading Magento ecommerce store, then it is advisable to use Magepack to optimize your page load speed , as explained here and improve your speed score.

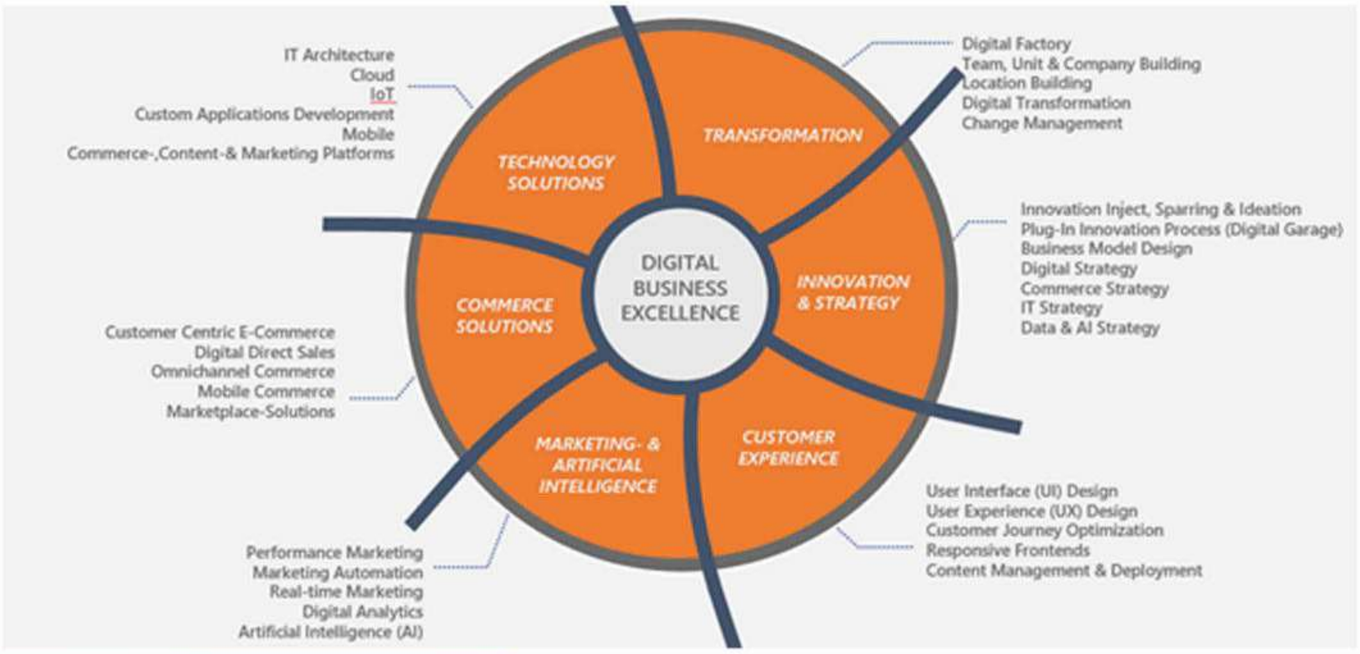
6. About the Author

Megha is a Senior Associate engineer at Embitel. A Magento 2 Certified developer, she brings along over 5 years of experience in coding and web development. Megha is known for her passion and enthusiasm for taking up challenging and complex coding problems and tackling them with innovative solutions. Apart from coding she enjoys travelling and cooking.

Embitel Technologies

// diconium
diconium digital solutions GmbH
Stuttgart, Germany
Started: 1995
Team Size: 700+
Platforms: Hybris, IBM, Intershop, Adobe
Top rated by New Media
embitel
delivered with passion
Embitel Technologies, Bangalore, India
Started: 2006
Team Size: 410+
Platforms: Magento, AEM, Hybris, Pimcore
Business Units: E-Com, IoT, PES
Our Presence: India | USA | UK | Middle East

Our Service Portfolio



DIGITAL BUSINESS EXCELLENCE
TECHNOLOGY SOLUTIONS
TRANSFORMATION
INNOVATION & STRATEGY
CUSTOMER EXPERIENCE
MARKETING- & ARTIFICIAL INTELLIGENCE
COMMERCE SOLUTIONS
IT Architecture
Cloud
IoT
Custom Applications Development
Mobile
Commerce-,Content-& Marketing Platforms
Digital Factory
Team, Unit & Company Building
Location Building
Digital Transformation
Change Management
Innovation Inject, Sparring & Ideation
Plug-In Innovation Process (Digital Garage)
Business Model Design
Digital Strategy
Commerce Strategy
IT Strategy
Data & AI Strategy
User Interface (UI) Design
User Experience (UX) Design
Customer Journey Optimization
Responsive Frontends
Content Management & Deployment
Performance Marketing
Marketing Automation
Real-time Marketing
Digital Analytics
Artificial Intelligence (AI)
Customer Centric E-Commerce
Digital Direct Sales
Omnichannel Commerce
Mobile Commerce
Marketplace-Solutions