MagentoLive

UK | 2015

Magento 2 Code Customizations

MagentoLive

UK | 2015

Anton Kril

Architect, Magento 2

MagentoLive UK | 2015

Agenda

- Service Contracts
- Presentation Layer
- Extension Points

Service Contracts

Extensions Compatibility Challenges

How to make sure that two extensions will be compatible in a new version?

How to implement extension in the way that it will keep backward compatibility but can evolve?

How to understand what functionality of the extension is stable and what is not?

Stable APIs

- Backward Compatible:
  - Classes or Interfaces are not removed
  - Methods of the classes keeps signature between versions
  - Interfaces neither changes existing methods nor add new ones

- Explicit Interfaces
  - No generic data types as “mixed”, “object” or “array”

Few ways to make promises in Magento 2

Semantic Versioning of the modules makes dependencies between modules explicit

```
{
    "name": "magento/module-catalog-inventory",
    "require": {
        "magento/module-customer": "0.74.0-beta2"
    },
    "type": "magento2-module"
}
```

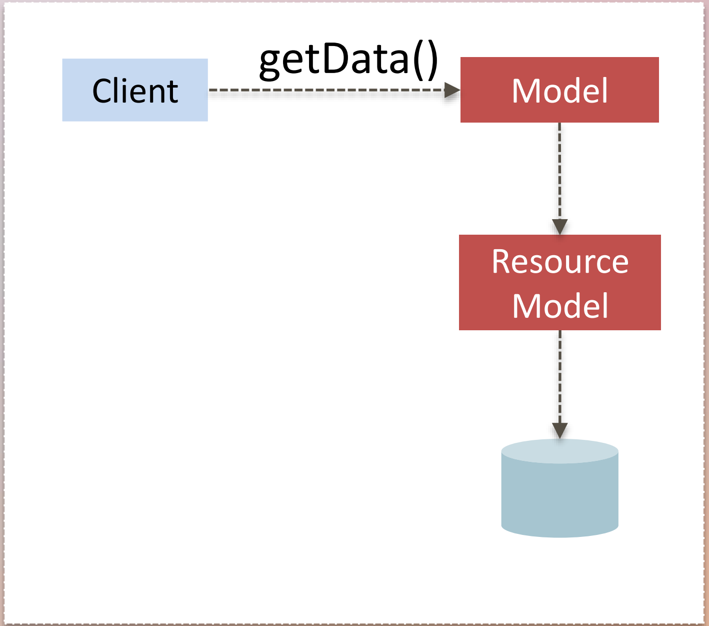
@api annotation identifies subset of the methods with the stable APIs

Enforced by tools and static tests

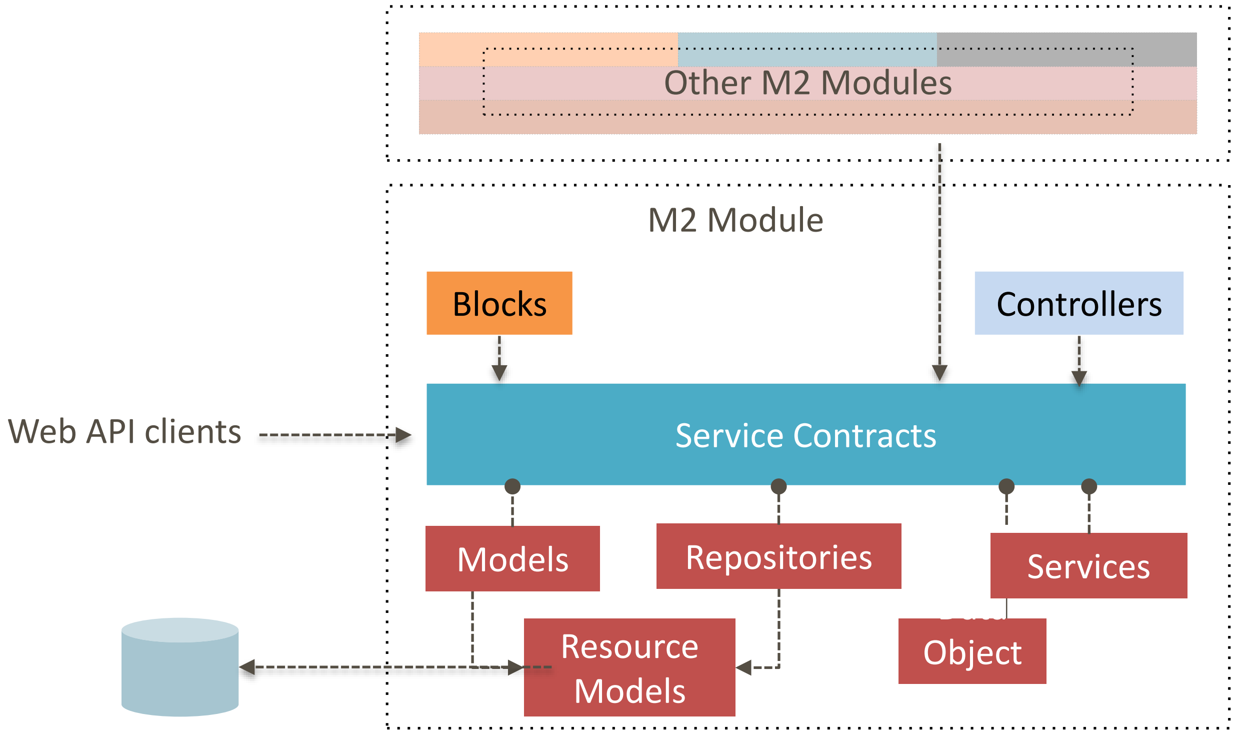
```
/**
 * @api
 */
interface AuthorizationInterface
{
    /**
     * Check current user permission on resource and privilege
     *
     * @param   string $resource
     * @param   string $privilege
     * @return  boolean
     */
    public function isAllowed($resource, $privilege = null);
}
```

Magento 1.x Domain Level API

- Model is an entry point to the Module
- Interface implicitly defined via the database schema
- No single place for the business rules
  They can reside in:
  - Controllers
  - Models
  - Helpers
  - Templates



Service Contracts



Service Contracts Interfaces

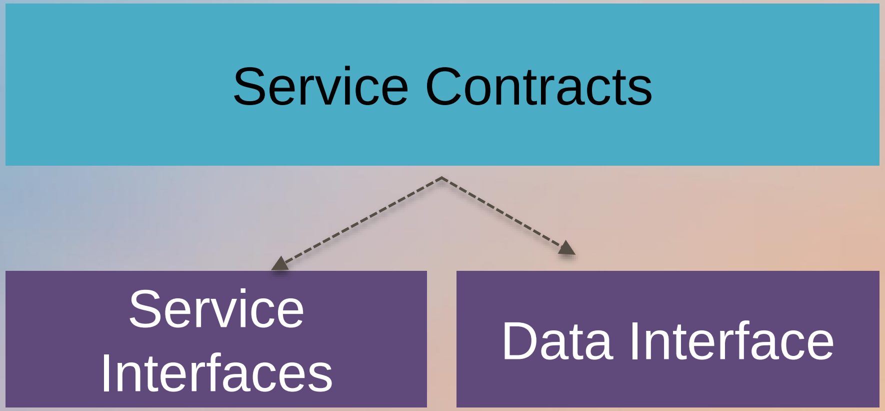
They are just PHP Interfaces

Service interfaces

- Defines business operations
- Examples: load, delete, save, change password, etc.

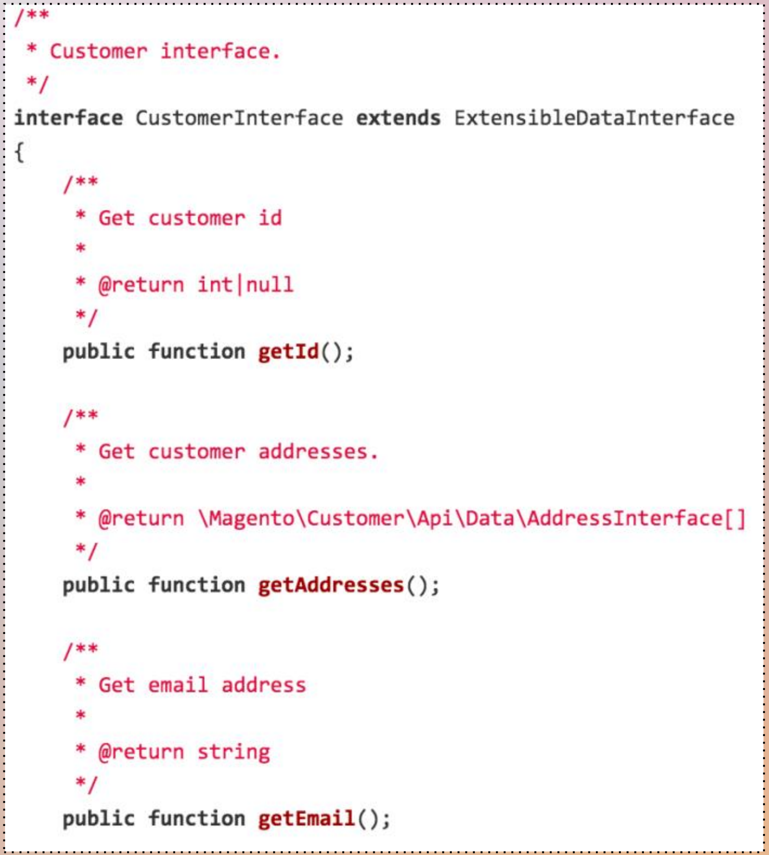
Data interfaces

- Defines data structures, used as input and output types of the business operations
- Examples: Customer, Product, Region, Currency, etc.



More on Data Interfaces

- Has just setters and getters to describe a data
- Reusable across different Service interfaces
- Encapsulates all the data needed to process service request
- Can be Serialized
  - Annotations are used to extract the data



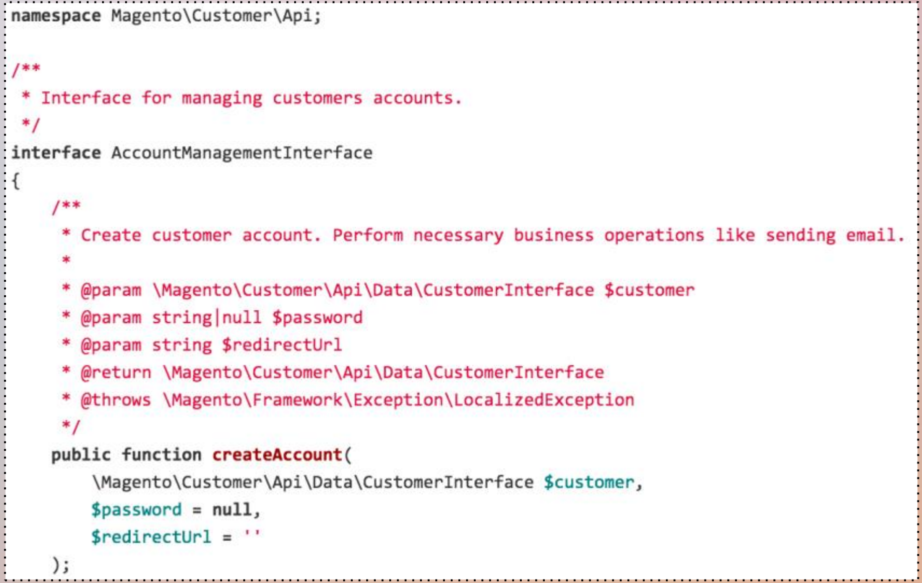
```
/**
 * Customer interface.
 */
interface CustomerInterface extends ExtensibleDataInterface
{
    /**
     * Get customer id
     *
     * @return int|null
     */
    public function getId();

    /**
     * Get customer addresses.
     *
     * @return \Magento\Customer\Api\Data\AddressInterface[]
     */
    public function getAddresses();

    /**
     * Get email address
     *
     * @return string
     */
    public function getEmail();
```

More on Service Interfaces

- Defines **public operations** supported by the module
- Methods **are independent and stateless**.
  - Invocation of one method should not affect the result of another
- Methods combined in interface by **cohesion** principle
- **Annotated** with types information



```
namespace Magento\Customer\Api;

/**
 * Interface for managing customers accounts.
 */
interface AccountManagementInterface
{
    /**
     * Create customer account. Perform necessary business operations like sending email.
     *
     * @param \Magento\Customer\Api\Data\CustomerInterface $customer
     * @param string|null $password
     * @param string $redirectUrl
     * @return \Magento\Customer\Api\Data\CustomerInterface
     * @throws \Magento\Framework\Exception\LocalizedException
     */
    public function createAccount(
        \Magento\Customer\Api\Data\CustomerInterface $customer,
        $password = null,
        $redirectUrl = ''
    );
```

MagentoLive UK | 2015

Service Contracts Resources

- **Magento 2 Developer Guide**

  http://devdocs.magento.com/guides/v1.0/extension-dev-guide/service-contracts/service-contracts.html

- **Alan Kent's Blog**

  https://alankent.wordpress.com/2014/10/31/magento-2-service-contract-patterns/

- **Presentation by Eugene Tulika**

  https://youtu.be/mi55hminjic?t=1303

Presentation Layer

MagentoLive UK | 2015

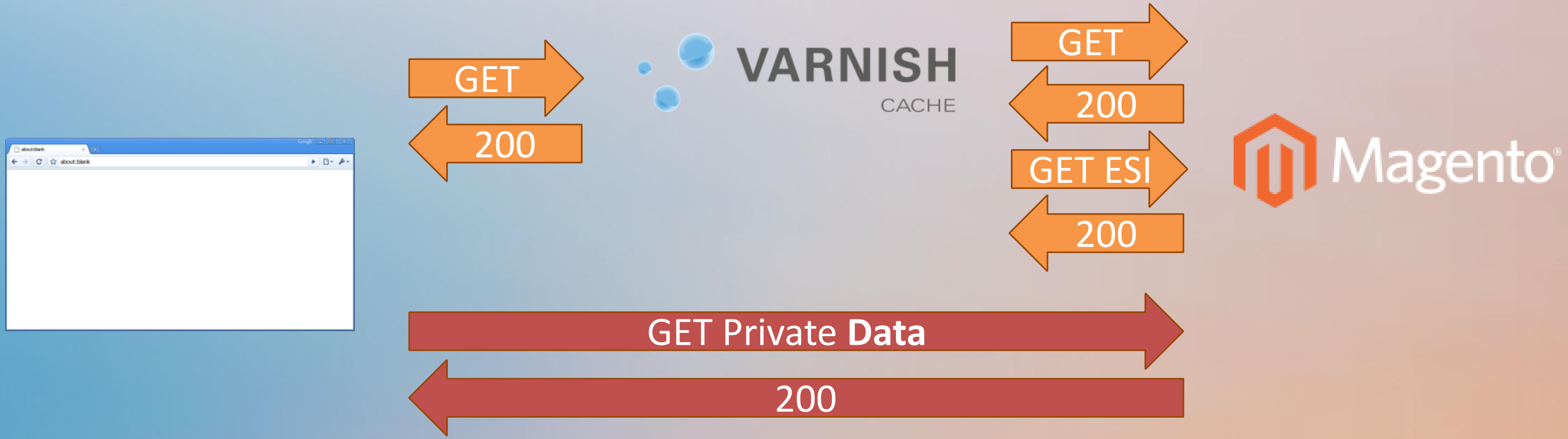
WebApi Declaration

Magento/Catalog/etc/webapi.xml

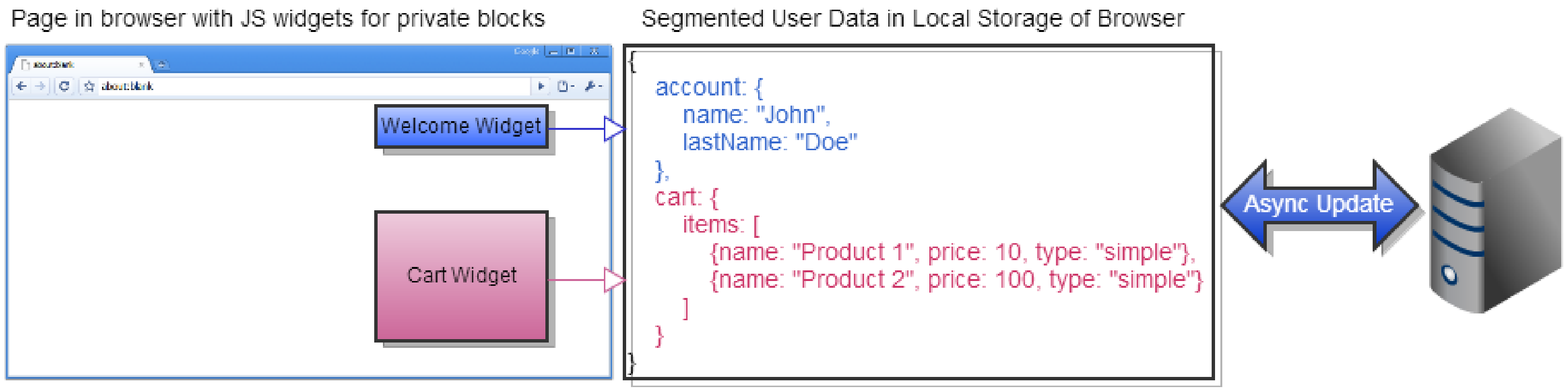
```
<routes xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <route url="/V1/products" method="POST">
    <service
          class="Magento\Catalog\Api\ProductRepositoryInterface"
          method="save"/>
    <resources>
        <resource ref="Magento_Catalog::products" />
    </resources>
  </route>
</routes>
```

http://devdocs.magento.com/guides/v1.0/get-started/bk-get-started-api.html

Page Cache


GET
200
VARNISH
CACHE
GET
200
GET ESI
200
Magento
GET Private **Data**
200
about:blank

Private Data Segments



Page in browser with JS widgets for private blocks
Segmented User Data in Local Storage of Browser
about:blank
Welcome Widget
Cart Widget
Async Update

```
{
  account: {
    name: "John",
    lastName: "Doe"
  },
  cart: {
    items: [
      {name: "Product 1", price: 10, type: "simple"},
      {name: "Product 2", price: 100, type: "simple"}
    ]
  }
}
```

UserData Section Declaration

My/Module/CustomerData/Segment.php

```php
namespace My\Module\CustomerData;

use Magento\Customer\CustomerData\SectionSourceInterface;

class Segment implements SectionSourceInterface
{
    public function getSectionData() {
        return [
            'last_post' => $this->blogService->getLastPostId(),
            'posts_count' => $this->blogService->getNumberOfPosts(),
        ];
    }
}
```

UserData Section Declaration

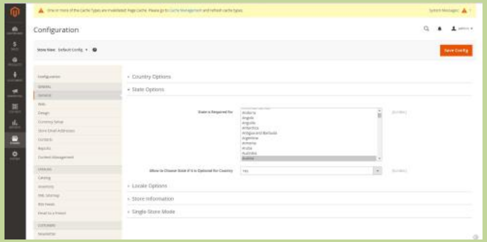
My/Module/etc/di.xml

```
<config xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <type name="Magento\Customer\CustomerData\SectionPoolInterface">
        <arguments>
            <argument name="sectionSourceMap" xsi:type="array">
                <item name="{{your_segment_name}}" xsi:type="string">
                    My\Module\CustomerData\Segment
                </item>
            </argument>
        </arguments>
    </type>
</config>
```

Extension Points

Customization points

Admin Configuration



One or more of the Cache Types are invalidated: Page Cache. Please go to Cache Management and refresh cache types. System Messages: 1

Configuration

Store View: Default Config

Save Config

Configuration

GENERAL

General

Web

Design

Currency Setup

Store Email Addresses

Contacts

Reports

Content Management

CATALOG

Catalog

Inventory

XML Sitemap

RSS Feeds

Email to a Friend

CUSTOMERS

Newsletter

Country Options

State Options

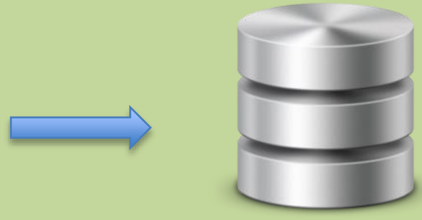
State is Required for: Andorra, Angola, Anguilla, Antarctica, Antigua and Barbuda, Argentina, Armenia, Aruba, Australia, Austria [GLOBAL]

Allow to Choose State if It is Optional for Country: Yes [GLOBAL]

Locale Options

Store Information

Single-Store Mode



High-level configuration

config.xml

routes.xml

layout

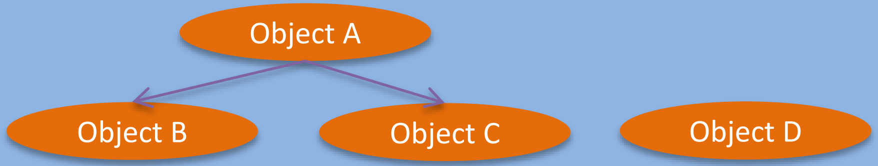
events.xml

routes.xml

webapi.xml

Low-level linking

di.xml



MagentoLive UK | 2015

Config Modification & Extension

Magento/Core/etc/config.xml

```
<config>
  <default>
    <design>
      <pagination>
        <list_allow_all>1</list_allow_all>
        <pagination_frame>10</pagination_frame>
        <step>10</step>
      </pagination>
    </design>
  </default>
</config>
```

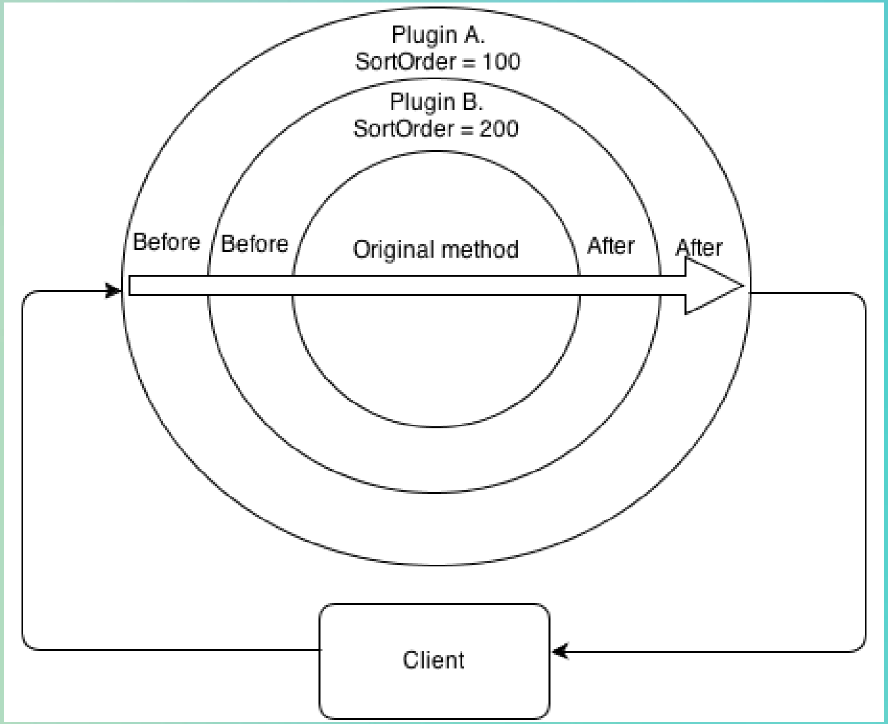
Code Extension & Modification

Class-rewrites
behavior modification (conflicting)

Events
behavior extension (non-conflicting)

Interception

- Ability to observe Public Methods
- Non-conflicting extension mechanism
- Also a method rewrite mechanism
- AOP



Pluginized Object

Lib/Magento/Framework/Url.php

```
namespace Magento\Framework;
class Url
{
    public function getUrl($routePath)
    {
      // Some url calculation
      return $calculatedUrl;
    }
}
```

Plugin

My/Module/Url/Plugin.php

```
namespace My\Module\Url;
class Plugin
{
    public function beforeGetUrl(\Magento\Url $subject, $routePath)
    {
        // Do something before url is built
        return '*/' . $routePath; // modify param
    }

    public function afterGetUrl(\Magento\Url $subject, $result)
    {
        return $result . '?someVar=3'; // modify return result
    }
}
```

Plugin Declaration

My/Module/etc/di.xml

```xml
<config>
  <type name="Magento\Framework\Url">
    <plugin name="my_plugin" type="My\Module\Url\Plugin"/>
  </type>
</config>
```

Magento 2 Class Rewrites

Non-granular

```
<preference for="Magento\Catalog\Model\ProductRepository“
            type=“My\Module\Model\ProductRepository" />
```

Granular

```
<type name="Magento\Eav\Model\Entity\Attribute\Config">
    <arguments>
        <argument name="reader" xsi:type="object">
           Magento\Eav\Model\Entity\Attribute\Config\Reader\Proxy
        </argument>
    </arguments>
</type>
```

Goals

- Reduced Upgrade efforts
- Streamlined customization process
- Higher code quality

Contacts

Anton Kril

akril@ebay.com

@antonkril