Services in Joomla 4

Allon Moritz

J and Beyond 13. May 2018

DIGITALPEAK

About Me

Allon Moritz

@digitpeak / @laoneo

Founder Digital Peak GmbH

Doing Joomla extensions since 2007

Joomla 4 Working group

Team Lead Media Manager

GsoC Mentor

Agenda

- Concept of “Inversion of Control”
- Dependency injection
- DI Container in Joomla 4
- Services in core
- Booting an extension
- Override services in the container
- Backwards compatibility
- Questions

Concept of “Inversion of Control”

Concept of “Inversion of Control”

Inversion of control is used to increase modularity of the program and make it extensible

Wikipedia (https://en.wikipedia.org/wiki/Inversion_of_control)

Also known as the Hollywood Principle - "Don't call us, we'll call you"

Martin Fowler (https://martinfowler.com/bliki/InversionOfControl.html)

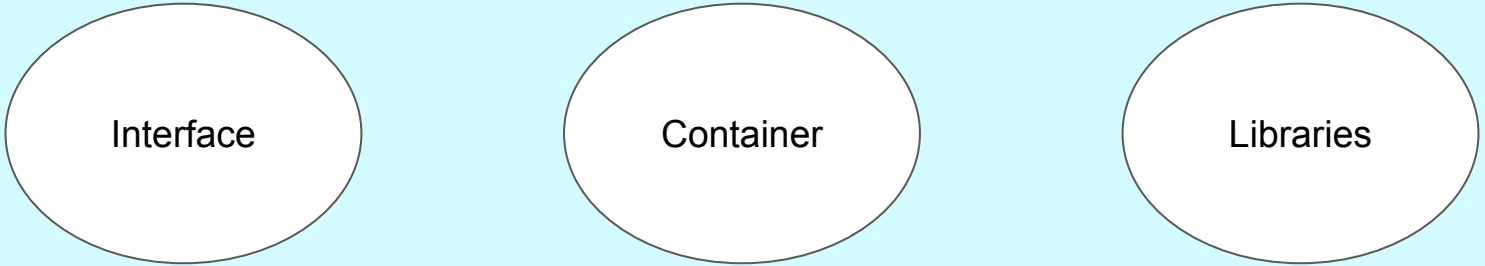
Dependency injection

Dependency injection

Dependency Injection is probably one of the most dead simple design pattern I know. But it is also one of the most difficult one to explain well.

Fabien Potencier (http://fabien.potencier.org/what-is-dependency-injection.html)

Dependency injection



Dependency injection

Dependency injected

```
1.  class AwesomeModel {
2.      private $logger = null;
3.
4.      public function __construct(LoggerInterface $logger) {
5.          $this->logger = $logger;
6.      }
7.
8.      public function doSomeWork() {
9.          $this->logger->log('Start working');
10.         // Do some work
11.         $this->logger->log('Finished working');
12.     }
13. }
```

Hardcoded dependency

```
1.  class AwesomeModel {
2.      private $logger = null;
3.
4.      public function __construct() {
5.          $this->logger = new FileLogger();
6.      }
7.
8.      public function doSomeWork() {
9.          $this->logger->log('Start working');
10.         // Do some work
11.         $this->logger->log('Finished working');
12.     }
13. }
```

Dependency injection

- Interface is the contract
- Constructor injection means a mandatory dependency
- Setter injection means an optional dependency
- Container manages the objects

DI Container in Joomla 4

DIGITALPEAK

DI Container in Joomla 4

- Imported from the framework
- PSR-11 compatible
- Core container
  - Manages the applications
  - Manages common global services like the JDocument service
- Extension container
  - Manages the extension dependencies
- Services are overridable

DI Container in Joomla 4

Set objects

$container->set($key, $resource);

Get objects

$container->get($key);

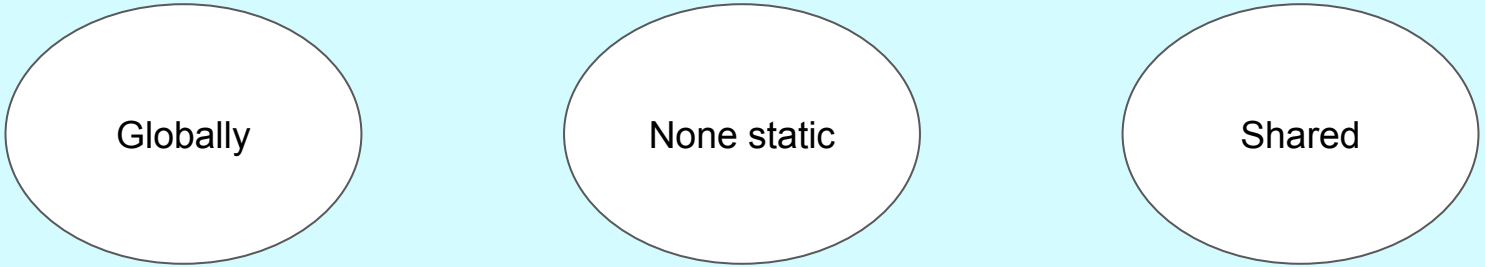
DI Container in Joomla 4

```php
class Form implements ServiceProviderInterface
{
    /**
     * Registers the service provider with a DI container.
     *
     * @param   Container  $container  The DI container.
     *
     * @return  void
     *
     * @since   4.0
     */
    public function register(Container $container)
    {
        $container->alias('form.factory', FormFactoryInterface::class)
            ->alias(FormFactory::class, FormFactoryInterface::class)
            ->share(
                FormFactoryInterface::class,
                function (Container $container)
                {
                    return new FormFactory;
                },
                true
            );
    }
}
```

https://github.com/joomla/joomla-cms/pull/16483

Services in core

Services in core



Services in core

Actual list of core services

1. new \Joomla\CMS\Service\Provider\Application
2. new \Joomla\CMS\Service\Provider\Authentication
3. new \Joomla\CMS\Service\Provider\Config
4. new \Joomla\CMS\Service\Provider\Console
5. new \Joomla\CMS\Service\Provider\Database
6. new \Joomla\CMS\Service\Provider\Dispatcher
7. new \Joomla\CMS\Service\Provider\Document
8. new \Joomla\CMS\Service\Provider\Form
9. new \Joomla\CMS\Service\Provider\Logger
10. new \Joomla\CMS\Service\Provider\Menu
11. new \Joomla\CMS\Service\Provider\Pathway
12. new \Joomla\CMS\Service\Provider\HTMLRegistry
13. new \Joomla\CMS\Service\Provider\Session
14. new \Joomla\CMS\Service\Provider\Toolbar

Services in core

How to access these services?

Injected, you don't care!

Give me an example.

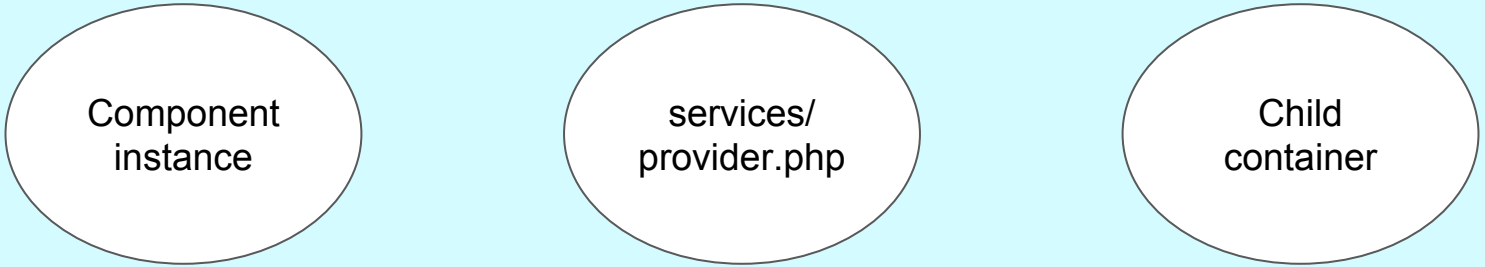
Have a look on the FormModel.
The FormFactory got injected through a setter.

How do I migrate from JForm::getInstance()?

Wait, I come to that later.

Booting an extension

Booting an extension



Booting an extension

$component = $app->bootComponent('com_foo');

```php
public function register(Container $container)
{
    $container->set(Categories::class, ['' => new Category]);
    $container->set(AssociationExtensionInterface::class, new AssociationsHelper);

    $container->registerServiceProvider(new MVCFactoryFactory('\\Joomla\\Component\\Content'));
    $container->registerServiceProvider(new DispatcherFactory('\\Joomla\\Component\\Content'));

    $container->set(
        ComponentInterface::class,
        function (Container $container)
        {
            $component = new ContentComponent;

            $component->setDispatcherFactory($container->get(DispatcherFactoryInterface::class));
            $component->setRegistry($container->get(Registry::class));
            $component->setMvcFactoryFactory($container->get(MVCFactoryFactoryInterface::class));
            $component->setCategories($container->get(Categories::class));
            $component->setAssociationExtension($container->get(AssociationExtensionInterface::class));

            return $component;
        }
    );
}
```

https://github.com/joomla/joomla-cms/pull/20217

Override services in the container

Override services in the container

Override services in the container

Global container overrides can be done in a system plugin:

```
1.  public function onBeforeExecute(BeforeExecuteEvent $event)
2.  {
3.      $event->getContainer()->share(
4.          FormFactoryInterface::class,
5.          function (Container $container)
6.          {
7.              return new MyAwesomeFormFactory;
8.          },
9.          true
10.     );
11. }
```

https://github.com/joomla/joomla-cms/pull/19842

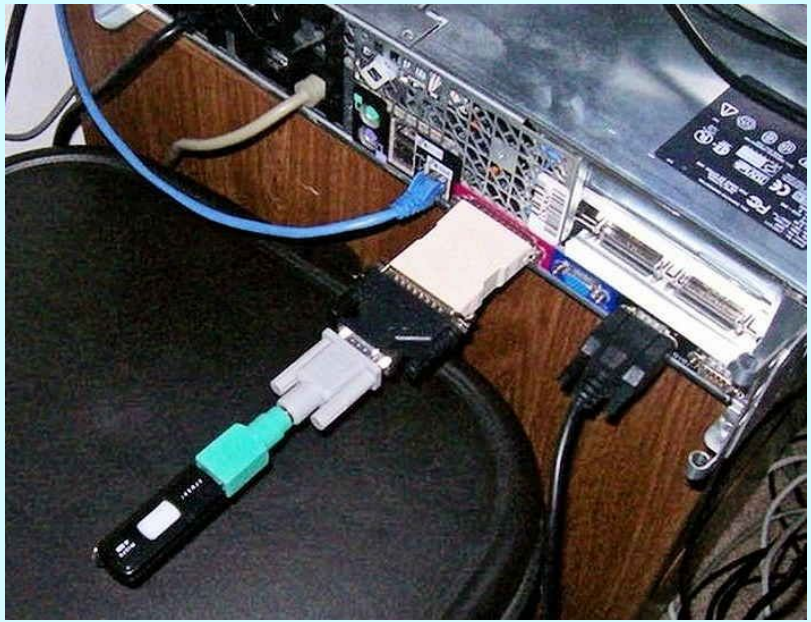
Override services in the container

Extension container overrides can be done in a system plugin:

```
1.  public function onAfterExtensionBoot(AfterExtensionBootEvent $event)
2.  {
3.          $event->getContainer()->share(
4.                  MVCFactoryFactoryInterface::class,
5.                  function (Container $container)
6.                  {
7.                          return new MyAwesomeMVCFactoryFactory;
8.                  },
9.                  true
10.         );
11. }
```

https://github.com/joomla/joomla-cms/pull/19841

Backwards compatibility



Backwards compatibility

- Old code still works, got deprecated for version 5
- It proxies to the container service

```
2192 +      * @since   11.1
2193 +      * @deprecated  5.0 Use the FormFactory service from the container
2194        * @throws  \InvalidArgumentException if no data provided.
2195        * @throws  \RuntimeException if the form could not be loaded.
2196        */
     @@ -2209,21 +2210,21 @@ public static function getInstance($name, $data = null, $options = array(), $rep
2210                      }
2211
2212                      // Instantiate the form.
     -                  $forms[$name] = new static($name, $options);
2213 +                  $forms[$name] = \JFactory::getContainer()->get(FormFactoryInterface::class)->createForm($name, $options);
```

https://github.com/joomla/joomla-cms/pull/16483

Questions?

Slides on https://joomla.digital-peak.com/blog/219-j-and-beyond-2018

Thank you and enjoy the day :-)