Mastering Symfony2

SF2C2

SensioLabs

Chapter 1

Dependency Injection

Life without Dependency Injection

The Logger Class

```
class Logger
{
    public function log($message, $level = 'INFO')
    {
        $formatter = new XmlFormatter();

        $log = $formatter->format($message, $level);

        $this->writeLog($log);
    }
}
```

Problems?

- Only supports XML formatted logs
- Hardcoded XmlFormatter dependency
- Not flexible
- Unit testing becomes harder

One step further to Dependency Injection

Removing the Dependency

```
class Logger
{
    private $formatter;

    function __construct(XmlFormatter $formatter)
    {
        $this->formatter = $formatter;
    }
}
```

Removing the Dependency

```
class Logger
{
    private $formatter;

    public function log($message, $level = 'INFO')
    {
        $log = $this->formatter->format($message, $level);

        $this->writeLog($log);
    }
}
```

Removing the Dependency

```
$logger = new Logger(new XmlFormatter());

$logger->log('An error to log');

/**
 * <log>
 *     <message>An error to log</message>
 * </log>
 */
```

Improving Flexibility

```
class Logger
{
    private $formatter;

    function __construct(Formatter $formatter)
    {
        $this->formatter = $formatter;
    }
}
```

Improving Flexibility

```
$logger = new Logger(new JsonFormatter());

$logger->log('An error to log');

/**
 * { message: "An error to log" }
 */
```

Dependency Injection

What's Dependency Injection?

« Dependency Injection is where components are given their dependencies through their constructors, methods, or directly into fields. »

http://picocontainer.org/injection.html

Constructor Injection

```
class Logger
{
    private $formatter;

    function __construct(Formatter $formatter)
    {
        $this->formatter = $formatter;
    }
}
```

Setter/Method Injection

```
class Logger
{
    private $formatter;

    function setFormatter(Formatter $formatter)
    {
        $this->formatter = $formatter;
    }
}
```

Interface Injection

```
class Logger
{
    private $formatter;

    function setFormatter(FormatterInterface $format)
    {
        $this->formatter = $format;
    }
}
```

Introduced Issue

Code becomes more decoupled and testable but objects construction and initialization become also more complex…

The Service Container

Goals

The Service Container is simply a PHP object that manages the instantiation of services.

The Container in the Cache

```
class appDevDebugProjectContainer extends Container
{
    protected function getLoggerService()
    {
        $instance = new \Symfony\Bridge\Monolog\Logger('app');

        $this->services['logger'] = $instance;

        $instance->pushHandler($this->get('monolog.handler.main'));
        $instance->pushHandler($this->get('monolog.handler.debug'));

        return $instance;
    }
}
```

instantiation

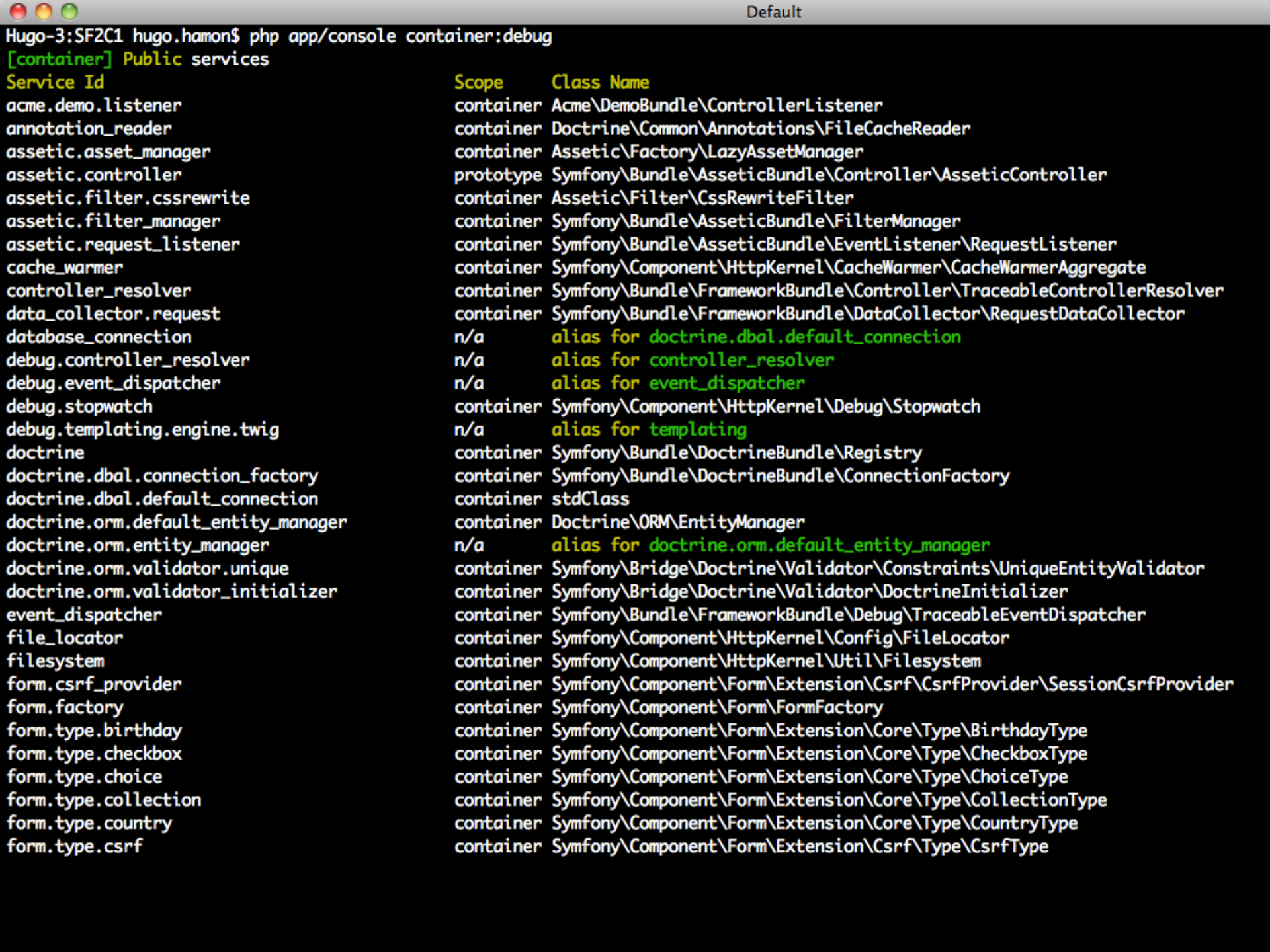
Initialization

Requesting a Service

```
$logger = $container->get('logger');
```

Requesting a Configuration Parameter

```
$container->getParameter('database_host');
```


Default
Hugo-3:SF2C1 hugo.hamon$ php app/console container:debug
[container] Public services
Service Id Scope Class Name
acme.demo.listener container Acme\DemoBundle\ControllerListener
annotation_reader container Doctrine\Common\Annotations\FileCacheReader
assetic.asset_manager container Assetic\Factory\LazyAssetManager
assetic.controller prototype Symfony\Bundle\AsseticBundle\Controller\AsseticController
assetic.filter.cssrewrite container Assetic\Filter\CssRewriteFilter
assetic.filter_manager container Symfony\Bundle\AsseticBundle\FilterManager
assetic.request_listener container Symfony\Bundle\AsseticBundle\EventListener\RequestListener
cache_warmer container Symfony\Component\HttpKernel\CacheWarmer\CacheWarmerAggregate
controller_resolver container Symfony\Bundle\FrameworkBundle\Controller\TraceableControllerResolver
data_collector.request container Symfony\Bundle\FrameworkBundle\DataCollector\RequestDataCollector
database_connection n/a alias for doctrine.dbal.default_connection
debug.controller_resolver n/a alias for controller_resolver
debug.event_dispatcher n/a alias for event_dispatcher
debug.stopwatch container Symfony\Component\HttpKernel\Debug\Stopwatch
debug.templating.engine.twig n/a alias for templating
doctrine container Symfony\Bundle\DoctrineBundle\Registry
doctrine.dbal.connection_factory container Symfony\Bundle\DoctrineBundle\ConnectionFactory
doctrine.dbal.default_connection container stdClass
doctrine.orm.default_entity_manager container Doctrine\ORM\EntityManager
doctrine.orm.entity_manager n/a alias for doctrine.orm.default_entity_manager
doctrine.orm.validator.unique container Symfony\Bridge\Doctrine\Validator\Constraints\UniqueEntityValidator
doctrine.orm.validator_initializer container Symfony\Bridge\Doctrine\Validator\DoctrineInitializer
event_dispatcher container Symfony\Bundle\FrameworkBundle\Debug\TraceableEventDispatcher
file_locator container Symfony\Component\HttpKernel\Config\FileLocator
filesystem container Symfony\Component\HttpKernel\Util\Filesystem
form.csrf_provider container Symfony\Component\Form\Extension\Csrf\CsrfProvider\SessionCsrfProvider
form.factory container Symfony\Component\Form\FormFactory
form.type.birthday container Symfony\Component\Form\Extension\Core\Type\BirthdayType
form.type.checkbox container Symfony\Component\Form\Extension\Core\Type\CheckboxType
form.type.choice container Symfony\Component\Form\Extension\Core\Type\ChoiceType
form.type.collection container Symfony\Component\Form\Extension\Core\Type\CollectionType
form.type.country container Symfony\Component\Form\Extension\Core\Type\CountryType
form.type.csrf container Symfony\Component\Form\Extension\Csrf\Type\CsrfType

Configuration

DIC Vocabulary

Name	Meaning
service	A global object managed by the DIC.
argument	A parameter given by the DIC to a service, via the constructor or a mutator method (setter).
parameter	A configuration value.

Service attributes

Name	Meaning
id	The service name
class	The class to instantiate
alias	An alias name for this service
public	False to make the service private and used internally
factory-service	A service that is used to instantiate the object
factory-method	A method to call to instantiate the object

Some services definition examples

```
<container>
    <services>

        <service id="sensio.logger" class="Sensio\Logger" />

        <service id="sensio.logger.xml_formatter"
                 alias="sensio.logger.formatter"
                 class="Sensio\Logger\Formatter\XmlFormatter"
                 public="false" />

    </services>
</container>
```

Service elements

Name	Meaning
argument	An argument to pass to a method (constructor or setter)
call	The method to call on the newly created object
configurator	A static method to call to initialize the object
tag	A tag to attach to the service definition
file	A file path to load before the service is created
property	Inject a dependency in a public property

Some services definition examples

```
<service id="sensio.logger" class="Sensio\Logger" >

    <!-- Constructor arguments -->
    <argument>/path/to/app/logs/app.log</argument>
    <argument type="service" id="sensio.logger.xml_formatter" />

    <!-- Methods to call -->
    <call method="setDefaultSeverity">
        <argument>DEBUG</argument>
    </call>

    <!-- Tags -->
    <tag name="tools.logger" />

</service>
```

Argument attributes

Name	Meaning
type	Define the nature of the argument. Allowed values: string, constant, collection or service
id	The service name
key	The argument key when dealing with the collection type
on-invalid	« ignore » will ignore the argument if it does not exist, otherwise an exception is raised.

Some services definition examples

```
<argument>Foo</argument>

<argument type="string">Foo</argument>

<argument type="constant">true</argument>

<argument type="constant">E_ALL</argument>

<argument type="constant">PDO::FETCH_NUM</argument>

<argument type="service" id="logger" on-invalid="ignore" />

<argument type="collection">
    <argument key="first" type="constant">true</argument>
</argument>
```

Registering the Configuration

From a YAML file

```yaml
imports:
    - { resource: parameters.yml }
    - { resource: security.yml }
    - { resource: services.xml }
```

From an XML file

```xml
<imports>
    <import resource= "parameters.yml" />
    <import resource= "security.yml" />
    <import resource= "services.xml" />
</imports>
```

SensioLabs Training Department

Address

92-98 Boulevard Victor Hugo

92 115 Clichy Cedex

France

Phone

+33 140 998 205

Email

training@sensiolabs.com