

Scraping de contidos e integración en Drupal8

Tomás Vilariño Fidalgo

Enxeñeiro en Informática (UVigo)

Técnico Especialista en Desenvolvemento Web (UJI)



vifito



vifito



@vifito

Sobre esta charla

DRUPAL DAY



**SANTIAGO DE
COMPOSTELA
2016**
12 DE NOVIEMBRE

**UNIVERSIDADE
SANTIAGO DE
COMPOSTELA**

**CAMPUS VIDA
15782 SANTIAGO DE
COMPOSTELA**

DRUPALDAY.ES



Contexto

Migración web USC.es

Sitio con miles de páginas

Exportar páginas "estáticas"

Compleja exportación datos

Manter a mesma estrutura navegação (SEO)

Primeira idea

Utilizar solucións de exportación do CMS actual (OpenCMS), ou acceso directo á base de datos

Problemas

- Descoñecemento da estrutura da base de datos, elevada curva de aprendizaxe
- O actual CMS emprega un sistema de ficheiros en base de datos (VFS)
- Outros sitios "satélite" que compre integrar (Xornal, Campus Terra, ...)

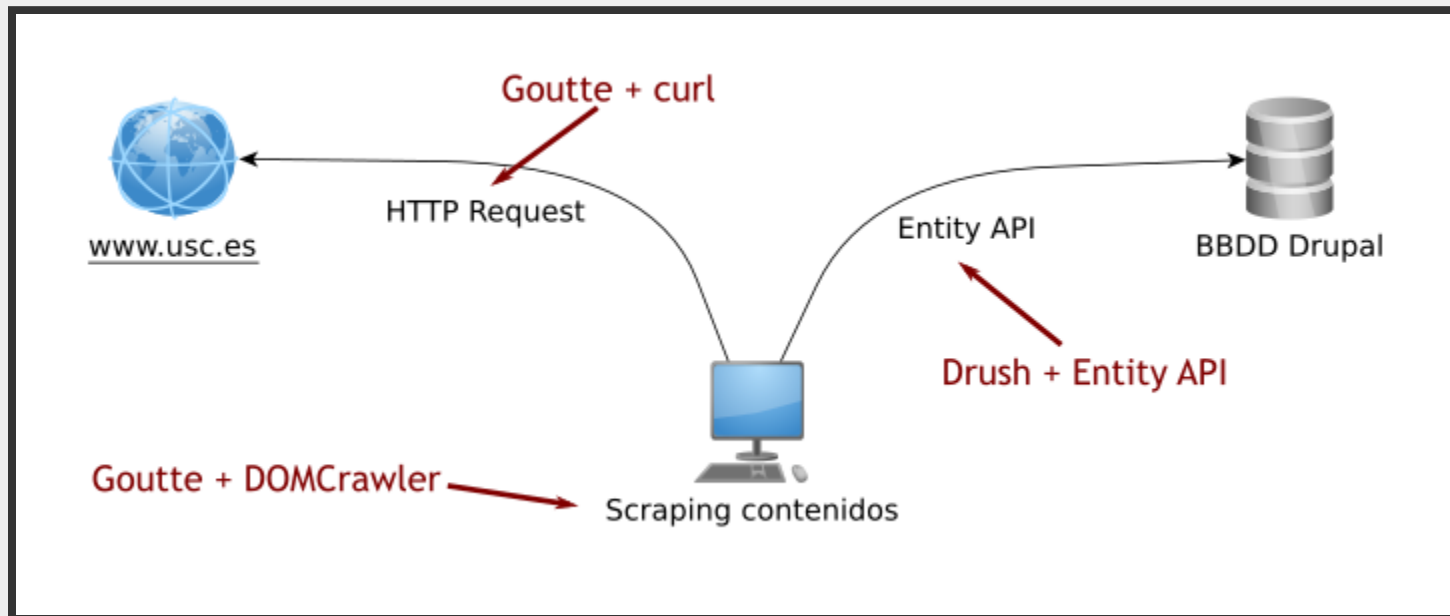
Segunda idea

Acceder directamente a través da web e extraer os contidos (scraping).

OLLO! falamos de contidos con pouca frecuencia de actualización (case "estáticos"). O resto de contidos son integrados vía servizos web.

Proba de concepto I

Empregar **Goutte** para recuperar e extraer contidos.
Insertar en Drupal a través dun comando **Drush**
empregando Entity API.
Todo implementado en **PHP** con componentes
alternativos **Guzzle** e **DOMCrawler**.



Proba de concepto I (inconvintes)

A idea funciona para unha páxina pero presenta inconvintes para rastrexar todo un sitio.

Faise necesario implementar a maiores:

- Rastrexo de ligazóns e xestión de duplicados
- Xestión da cola de peticións pendentes
- Almacenamento dos contidos extraídos
- Procesamento multifío
- Xestión jobs: poder parar e reanudar o traballo
- Estatísticas, tratamento imaxes, ficheiros, xestión de redireccións, erros 500...

Proba de concepto II

Precisamos algo máis que unha API de scraping de páxinas.

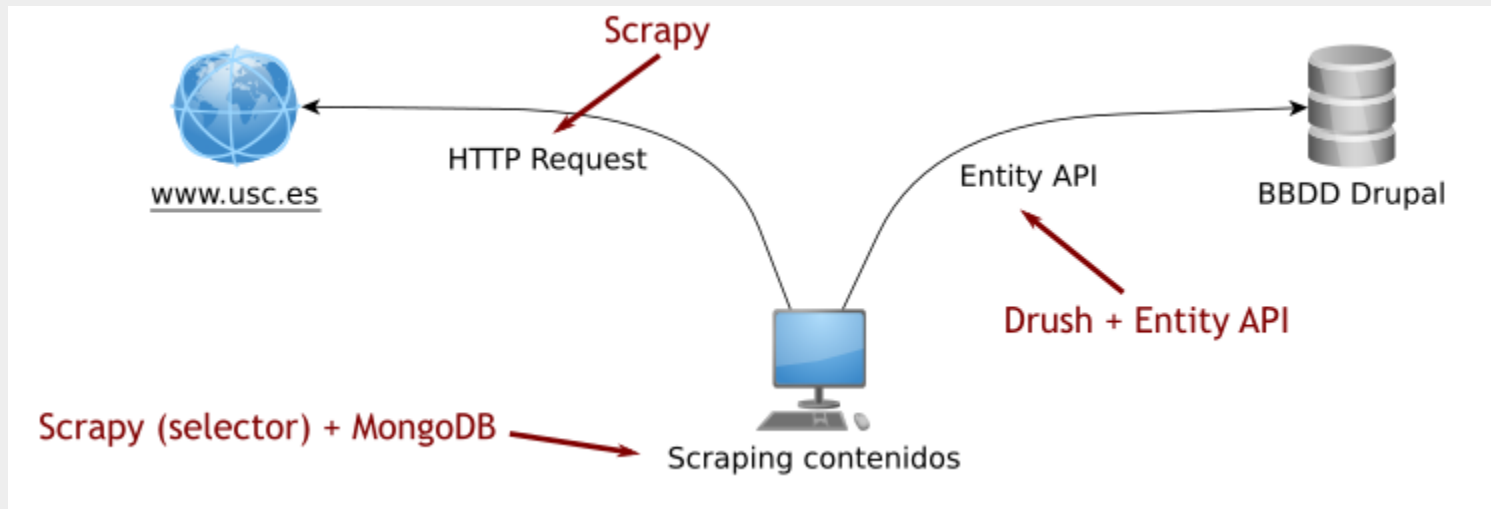
- Scraping, extracción de contidos
- Crawling, bot para seguir a navegación
- Storing, almacenar os contidos
- Plumbing, outras tarefas de «fontanería»

Proba de concepto II

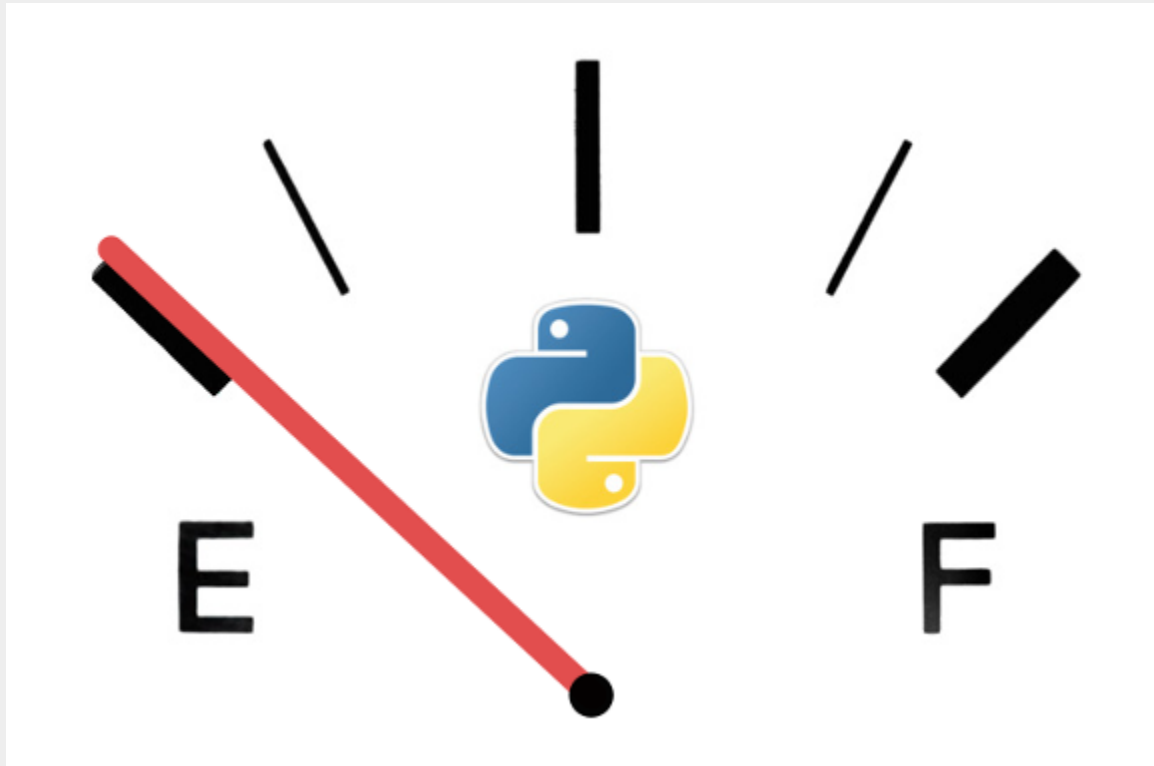
Atopamos un framework máis completo centrado na programación de arañas. Empregamos o framework Scrapy e obtemos contidos en formato JSON. Posteriormente con Drush integrámoslos en Drupal.



Scrapy



Antes de continuar



O meu nível de Python é baixo (tirando a nulo), porén nunha tarde estás programando arañas.

Instalación framework scraping (Scrapy)

Instalación de scrapy empregando pip

```
$ sudo pip install scrapy
```

Empregamos o xestor de paquetes de Python [pip](#) 

pypi v8.1.2

Requisitos previos (Ubuntu)

```
$ sudo apt-get install python-dev python-pip libxml2-dev \  
libxslt1-dev zlib1g-dev libffi-dev libssl-dev
```

Probamos a instalación de scrapy

```
vifito@arkham:~/PycharmProjects/sitepoint-scrapy$ scrapy
Scrapy 1.1.1 - no active project

Usage:
  scrapy <command> [options] [args]

Available commands:
  bench          Run quick benchmark test
  commands
  fetch          Fetch a URL using the Scrapy downloader
  genspider      Generate new spider using pre-defined templates
  runspider      Run a self-contained spider (without creating a project)
  settings       Get settings values
  shell          Interactive scraping console
  startproject   Create new project
  version        Print Scrapy version
  view           Open URL in browser, as seen by Scrapy

[ more ]        More commands available when run from project directory
```

Iniciamos proxecto de scrapy

Creamos un proxecto de scrapy coa opción
startproject

```
$ scrapy startproject gdgourense
```

Estrutura do proxecto

```
.
├── scrapy.cfg      → Ficheiro de scrapy (nome do proxecto)
└── gdgourense
    ├── __init__.py
    ├── items.py    → Clase coa definición dos ítems a extraer
    ├── pipelines.py → Clase que procesa, valida, almacena, ... ítems
    ├── settings.py → Configuración de opcións do proxecto
    └── spiders
        └── __init__.py
```

Tipos de arañas

Dende líña de comandos podemos crear o "esquelete" da araña coa opción `genspider`. Un proxecto pode conter varias arañas.

Scrapy vén con varios modelos para implementar arañas.

```
$ scrapy genspider --list
Available templates:
  basic
  crawl
  csvfeed
  xmlfeed
```


Clases Python

basic

A clase herda de `scrapy.Spider`, é a implementación máis sinxela.

crawl

A clase herda de `scrapy.CrawlSpider`. Por medio dunhas regras busca ligazóns e vai enchendo a cola.

xmlfeed

A clase herda de `XMLFeedSpider`. Deseñada para parsear feeds XML.

csvfeed

A clase herda de `CSVFeedSpider`. Similar a `XMLFeedSpider` para documentos CSV.

Crear unha araña

```
$ scrapy genspider -t crawl sitepoint www.sitepoint.com
```

Código da clase xerada

OLLO! O código xerado pode variar en función da versión de scrapy instalada.

```
# imports ...
class SitepointSpider(CrawlSpider):
    name = 'sitepoint'
    allowed_domains = ['www.sitepoint.com']
    start_urls = ['http://www.sitepoint.com/']

    rules = (
        Rule(LinkExtractor(allow=r'Items/'), callback='parse_item'),
    )

    def parse_item(self, response):
        i = {}
        #i['domain'] = response.xpath('//input[@id="sid"]/@value').extract()
        #i['name'] = response.xpath('//div[@id="name"]').extract()
        #i['desc'] = response.xpath('//div[@id="description"]').extract()
        return i
```

Definimos as nosas regras

```
rules = (  
    Rule(  
        LinkExtractor(  
            # xpathBasePath é unha expresión regular por liña de comandos  
            restrict_xpaths=xpathBasePath,  
            deny=[r'/(en|es)/', r'(calMonth=|calYear=) ']  
        ),  
        callback='process_item',  
        follow=True  
    ),  
)
```

Ítems a extraer (items.py)

Definimos a clase PaxinasItem cos campos que imos a recuperar.

```
11 class PaxinasItem(scrapy.Item):
12     # define the fields for your item here like:
13
14     url = scrapy.Field()
15     base_path = scrapy.Field()
16     title = scrapy.Field()
17     lang = scrapy.Field()
18     description = scrapy.Field()
19     keywords = scrapy.Field()
20     body = scrapy.Field()
21     headers = scrapy.Field()
22
23     titulo = scrapy.Field()
24     contidos = scrapy.Field()
25
26     image_alts = scrapy.Field()
27     image_urls = scrapy.Field()
28     images = scrapy.Field()
29
30     file_urls = scrapy.Field()
31     files = scrapy.Field()
32
```

ETIQUETA	NOME DE SISTEMA	TIPO DE CAMPO
Body	body	Text (formatted, long, with summary)
Descrición	field_fe_description	Text (plain, long)
Ficheiros	field_fe_file	Ficheiro
Imaxe	field_fe_image	Imaxe
Keywords	field_fe_keywords	Entity reference
Sección	field_fe_section	Entity reference

Selectores

CSS (response.css)	XPath (response.xpath)
title::text	//title/text()
base::attr(href)	//base/@href
a[href*=image]::attr(href)	//a[contains(@href, "image")]/@href

También con expresiones regulares `response.re()`.

.grid_11.logo.grid_4.busca.separador

.grid_15.separador

[Inicio](#) » [Información xeral](#) » [Visitas ao patrimonio](#)

#contidos.grid_12.alpha.grid_3.omega.blq-navegacion-lateral

Visitas guiadas ao Patrimonio Histórico da USC

.centrado



scrapy shell http://...

virito@arkham: ~/public_html/dev/tools/scrapy/paxinas

virito@arkham: ~/public_html/dev/tools/scrapy/paxinas 95x11

```
drupalday$ scrapy shell http://www.usc.es/gl/info_xeral/visitaspatrimonio/
```

Implementamos parse_item

Empregamos scrapy shell para hacer pruebas dos selectores

```
class PaxinasSpider(CrawlSpider):
    # ....
    def parse_item(self, response):
        # Comprobación: 'text/html' not in response.headers['Content-Type']

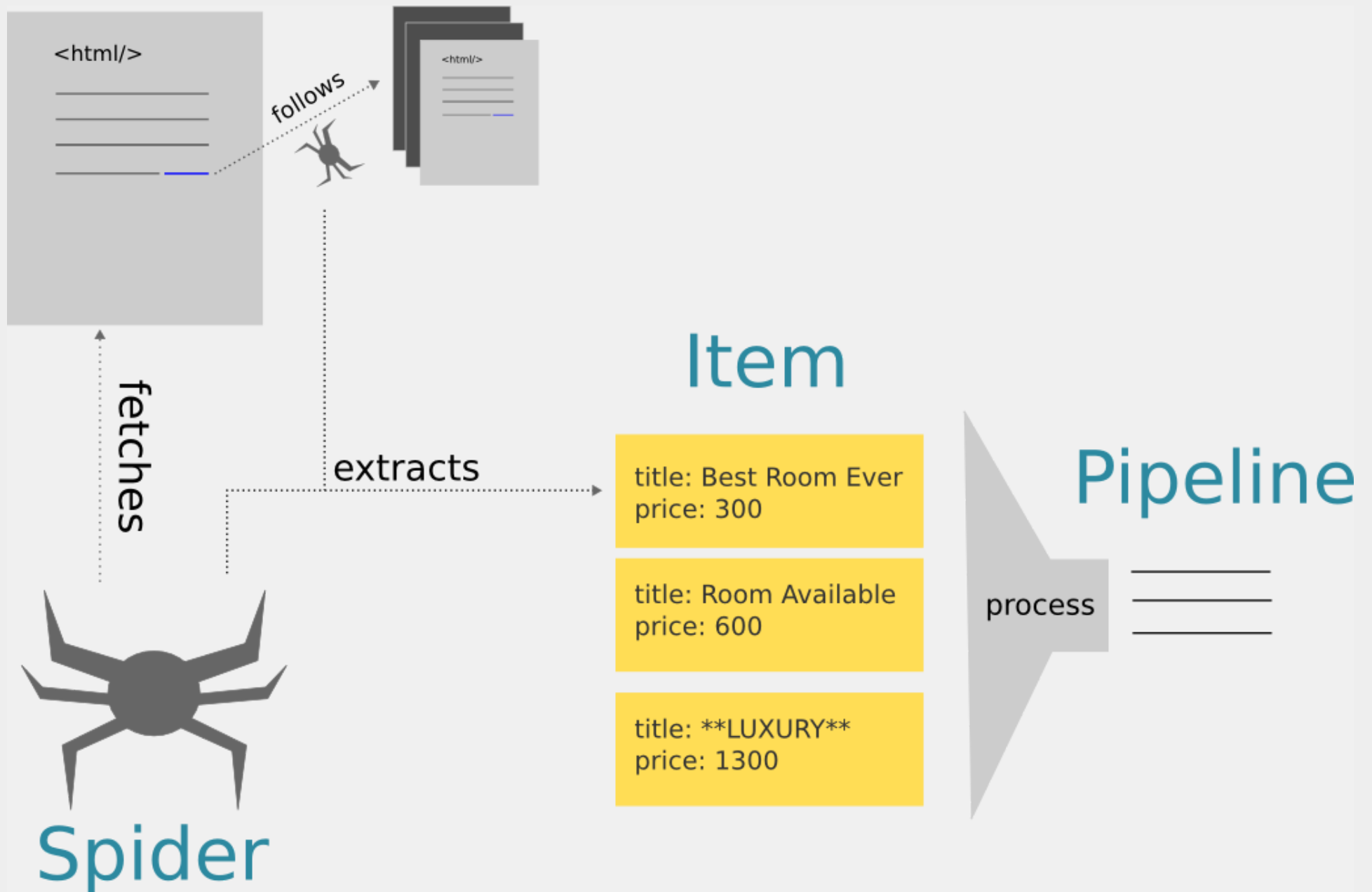
        item = PaxinasItem()
        item['url'] = response.url
        item['title'] = response.xpath('//title/text()').extract_first()
        item['body'] = response.body
        item['keywords'] = response.xpath('//meta[@name="keywords"]/@content')
                               .extract_first()
        # Búsqueda de imaxes e ficheiros
        # ....
        item['image_urls'] = images
        item['files_urls'] = files
        return item
```


Configurar pipelines

Precisamos configurar `ITEM_PIPELINES` para dispoñer do comportamento automático dos campos `image_urls` e `file_urls`.

```
ITEM_PIPELINES = {  
    'scrapy.pipelines.images.ImagesPipeline' : 1,  
    'scrapy.pipelines.files.FilesPipeline' : 1,  
}  
  
dir = os.path.dirname(os.path.abspath(__file__))  
IMAGES_STORE = dir + '/../assets/images'  
FILES_STORE  = dir + '/../assets/files'
```

Funcionamento dun pipeline



Executamos crawler

Lanzamos para que garde en formato json (-o xerencia.json), con parámetros adicionais (-a), incluimos a opción de parar e reanudar o crawler (-s JOBDIR) e gardamos un log.

```
scrapy crawl paxinas -o xerencia.json \  
-a start_url='http://www.usc.es/gl/gobierno/xerencia/' \  
-a base_path='/gobierno/xerencia/' \  
-s JOBDIR=crawls/paxinas -1 \  
--logfile=xerencia.log
```

Mapeo de campos

Temos que mapear os campos do documento JSON aos campos do tipo de contido.

```
11 class PaxinasItem(scrapy.Item):
12     # define the fields for your item here like:
13
14     url = scrapy.Field()
15     base_path = scrapy.Field()
16     title = scrapy.Field()
17     lang = scrapy.Field()
18     description = scrapy.Field()
19     keywords = scrapy.Field()
20     body = scrapy.Field()
21     headers = scrapy.Field()
22
23     titulo = scrapy.Field()
24     contidos = scrapy.Field()
25
26     image_alts = scrapy.Field()
27     image_urls = scrapy.Field()
28     images = scrapy.Field()
29
30     file_urls = scrapy.Field()
31     files = scrapy.Field()
32
```

ETIQUETA	NOME DE SISTEMA	TIPO DE CAMPO
Body	body	Text (formatted, long, with summary)
Descrición	field_fe_description	Text (plain, long)
Ficheiros	field_fe_file	Ficheiro
Imaxe	field_fe_image	Imaxe
Keywords	field_fe_keywords	Entity reference
Sección	field_fe_section	Entity reference

Estructura JSON

(versión reducida)

```
[{
  "title": "Servizo de Xestión Académica - USC",
  "url": "http://www.usc.es/gl/servizos/sxa",
  "keywords": "usc, universidade, santiago , compostela, ...",
  "contidos": "<p>...</p>",
  "image_urls": ["http://www.usc.es/gl/servizos/sxa/imaxes/coie.jpg"],
  "file_urls": ["http://www.usc.es/gl/servizos/sxa/4_Vacantes_Master.pdf"],
  "images": [{
    "url": "http://www.usc.es/gl/servizos/sxa/imaxes/coie.jpg",
    "path": "full/7eb49e6658ab0f2e7ed894027529dfaeebd9346a.jpg"
  }],
  "files": [{
    "url": "http://www.usc.es/gl/servizos/sxa//4_Vacantes_Master.pdf",
    "path": "full/67063aa5e3f072742ce4d2c066b973d1ceca64c8.pdf"
  }]
}, {...}]
```

Integración en Drupal

Crear un módulo Drupal

Ficheiro `module_name.drush.inc`, contén os comandos que poderemos executar con Drush.

```
/**
 * Implements hook_drush_command().
 */
function gdgourense_drush_command() {
  $items['import-pages'] = [
    'callback' => 'import_pages',
    'description' => 'Importar páxinas dun sitio (formato JSON).',
    'aliases' => array('ipag'),
    'arguments' => [
      'input' => 'Ficheiro JSON co contido das páxinas a cargar.', ],
    'options' => [
      'assets' => 'Ruta ao directorio de assets (files e images)',
      'menu_name' => 'Nome do menú a xerar',
    ],
  ];
}
```

Procesamento ficheiro JSON

Recorrer ítems do array json e ir inserindo o contido

```
$data = json_decode(file_get_contents($input), TRUE);  
/** @var \Drupal\Core\Path\AliasStorage $aliasStorage */  
$aliasStorage = \Drupal::service('path.alias_storage');  
  
foreach($data as $item) {  
    if (!$aliasStorage->aliasExists($link, $langcode)) {  
        $values = [  
            'type' => 'page_general',  
            'title' => $item['titulo'],  
            // ...  
            'body' => ['value' => $item['contidos'],],  
            'field_description' => $item['description'],  
        ];  
    }  
    // ... tratar imágenes y ficheros  
}
```


Ficheiros xestionados

Recorrer arrays de `images` e `files`, e damos de alta en Drupal (táboa: `file_managed`)

```
// Versión reducida: validacións omitidas, file_exists, ...
foreach($item['images'] as $img) {
    $path = 'public://' . $img['url'];
    $imagename = realpath($assets . '/images/' . $img['path']);
    $dir = dirname($path);
    if(!file_prepare_directory($dir)) {
        drupal_mkdir($dir, NULL, True);
    }
    $image_stream = file_get_contents($imagename);
    $file = file_save_data($image_stream, $path, FILE_EXISTS_REPLACE);
    $image_id[] = [
        'target_id' => $file->id(),
        'alt' => $alternativeText,
    ];
    // Remprazar os href no contido ($values['body'])
}
$values['field_image'] = $image_id; // array cos IDs das imáxes
```

Outros campos (taxonomías)

Crear taxonomías para o campo keywords

```
$terms = explode(',', $item['keywords']);
foreach($terms as $term) {
    $termId = $storage->getQuery()
        ->condition('vid', $keywords_vid)
        ->condition('name', $term)->execute();

    if (empty($termId)) {
        $termEntity = \Drupal\taxonomy\Entity\Term::create([
            'vid' => $keywords_vid,
            'name' => $term,
        ]);
        $termEntity->save();
        $termId = [$termEntity->id() => $termEntity->id()];
    }
    $tids[] = ['target_id' => current($termId)];
}
$values['field_keywords'] = $tids;
```

Integrar con menús I

Comprobar que existe un menú, noutro caso crealo

```
// $menu_name => nome do menú
$nids = \Drupal::entityQuery('menu')
  ->condition('id', $menu_name)->execute();

if (empty($nids)) { // Crear menú si no existe
  $menu_desc = t('Menú') . ' ' .
    ucwords(str_replace('menu-', '', $menu_name));
  /** @var \Drupal\system\Entity\Menu $menu */
  $menu = entity_create('menu', [
    'id'          => $menu_name,
    'label'       => $menu_desc,
    'description' => $menu_desc,
    'langcode'    => 'gl',
  ]);
  $menu->save();
}
```

Integrar con menús II

Crear ítems do menú

```
// Comprobar se xa existe
$nids = \Drupal::entityQuery('menu_link_content')
  ->condition('default_langcode', 1)->condition('title', $item['titulo'])
  ->condition('link.uri', 'internal:' . $link)
  ->condition('menu_name', $menu_name)->execute();

if (count($nids) === 0) {
  $menuData = [
    'title' => $item['titulo'],
    'menu_name' => $menu_name,
    // ...
    'link' => ['uri' => 'internal:' . $link],
  ];

  $menu_link_content = MenuLinkContent::create($menuData);
  $menu_link_content->save();
}
```

Outras alternativas

- Empregar o módulo **migrate** para cargar os contidos.
- Inserir dende código Python os contidos directamente na **base de datos**.
- Publicar os **servizos REST** de Drupal e dende código Python inserir os contidos empregando o API.

BANNED



User-Agent aleatorio

Creamos un middleware que escollerá aleatoriamente dunha lista un user-agent distinto en cada petición.

```
# ./scrapy/gdgourense/settings.py
USER_AGENT_LIST = [
    'Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/535.7 (KHTML, ...' ,
    'Mozilla/5.0 (Windows NT 6.2; Win64; x64; rv:16.0) Gecko/16.0 ...' ,
    'Mozilla/5.0 (Macintosh; Intel Mac OS X 10_7_3) AppleWebKit/534...'
]

# ./scrapy/gdgourense/middlewares.py
#...
class RandomUserAgentMiddleware(object):
    def process_request(self, request, spider):
        ua = random.choice(settings.get('USER_AGENT_LIST'))
        if ua:
            request.headers.setdefault('User-Agent', ua)
```

Empregar Proxies

No caso que nos bloqueen por IP a nosa máquina podemos botar man dos proxies.

```
# ./scrapy/gdgourense/settings.py
#HTTP_PROXY = 'http://127.0.0.1:8123' # Polipo proxy service
HTTP_PROXY = 'http://127.0.0.1:8118' # Privoxy

# ./scrapy/gdgourense/middlewares.py
#...
class ProxyMiddleware(object):
    def process_request(self, request, spider):
        request.meta['proxy'] = settings.get('HTTP_PROXY')
```

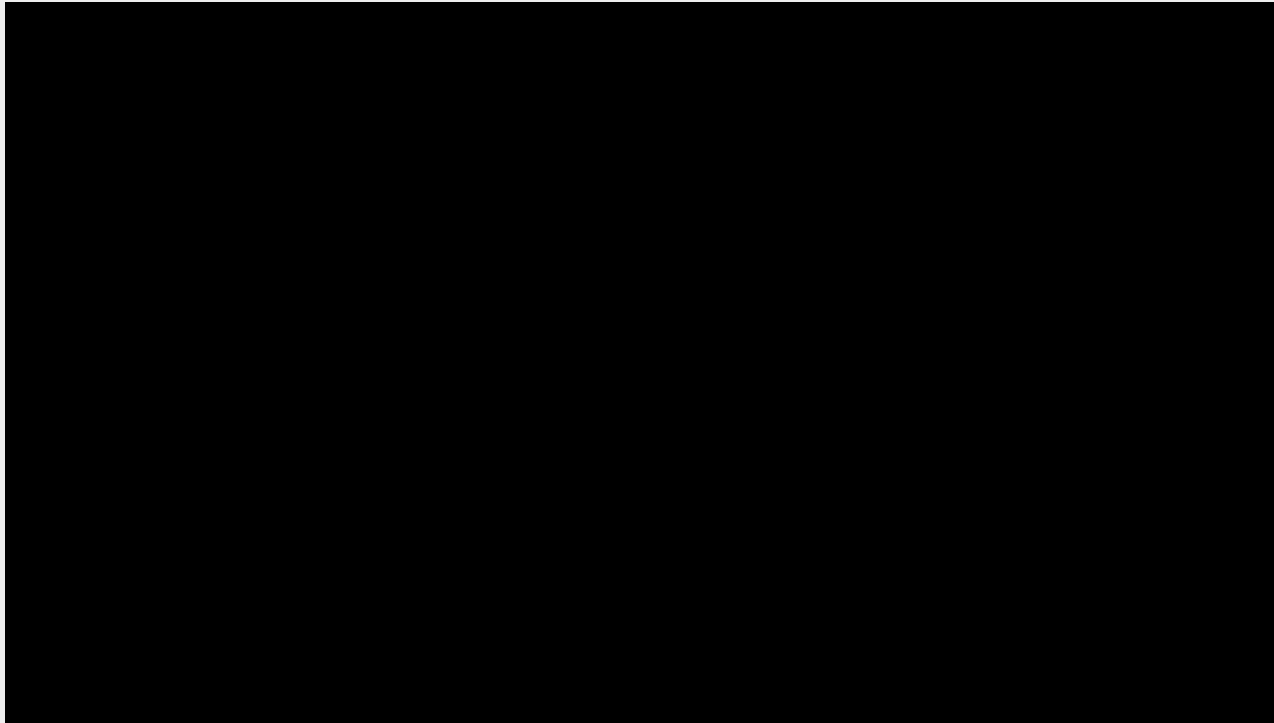

Privoxy e Tor

Podemos configurar Scrapy para realice peticións a través de Tor e Privoxy.

```
sudo apt-get install tor privoxy  
# sudo apt-get install polipo
```

Configurar Tor e Privoxy [↗](#)

Solución en scrapinghub



Almacenamiento alternativo

Gardar os contidos en MongoDB para realizar consultas e xerar un JSON máis personalizado.

```
import pymongo
# import ...
class PaxinasPipeline(object):
    def __init__(self):
        connection = pymongo.MongoClient(
            settings['MONGODB_SERVER'],
            settings['MONGODB_PORT']
        )
        db = connection[settings['MONGODB_DB']]
        self.collection = db[settings['MONGODB_COLLECTION']]

    def process_item(self, item, spider):
        for data in item:
            if not data:
                raise DropItem("Missing data!")
        self.collection.update({'url': item['url']}, dict(item), upsert=True)
        return item
```

Configuramos pipeline MongoDB

```
ITEM_PIPELINES = {  
    'paxinas.pipelines.PaxinasPipeline': 1,  
    # ... imaxes, ficheiros, ...  
}
```

```
MONGODB_SERVER = "localhost"  
MONGODB_PORT = 27017  
MONGODB_DB = "gdgourense"  
MONGODB_COLLECTION = "paxinas"
```

Que pasa coas single-page application (SPA)?



Splash


Splash é un **servizo de renderizado javascript**.

Algunhas das súas características:

- Recuperar o resultado en HTML, ou pantallazo
- Non cargar imaxes ou empregar regras de Adblock Plus para renderizar máis rápido
- Executar Javascript no contexto da páxina
- Scripting en Lua
- Obter captura HAR (HTTP Archive)

Tamén se pode combinar Scrapy con PhamtonJS ou Selenium.

Splash e Scrapy

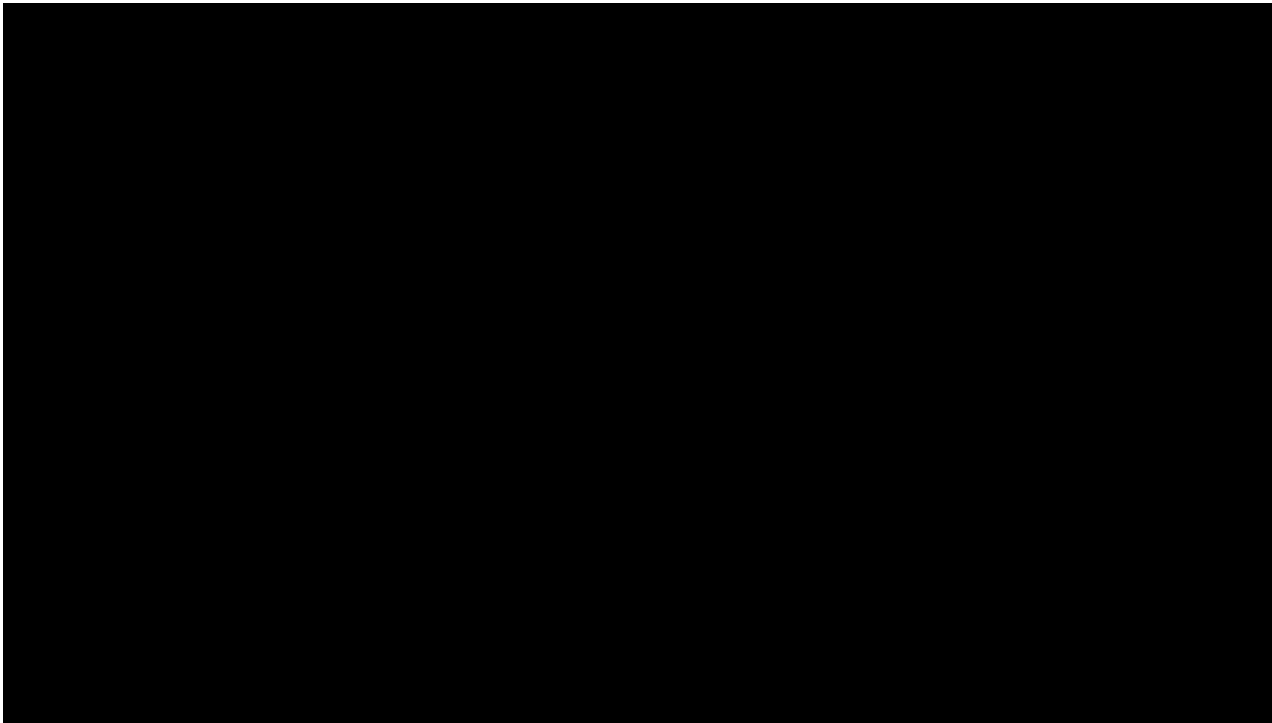
Empregamos o servico de Splash a través dun **middleware** . As páxinas *descárganse e interprétanse* con Splash e pásanse a Scrapy.

```
# ./scrapy/gdgourense/settings.py
SPLASH_URL = 'http://192.168.59.103:8050'
DOWNLOADER_MIDDLEWARES = {
    'scrapy_splash.SplashCookiesMiddleware' : 723,
    'scrapy_splash.SplashMiddleware' : 725,
    'scrapy.downloadermiddlewares.httpcompression.HttpCompressionMiddleware'
}
```



Portia

Portia, proxecto que permite extraer de forma visual.




```
docker pull scrapinghub/portia
docker run -i -t --rm -p 9001:9001 --name portia portia

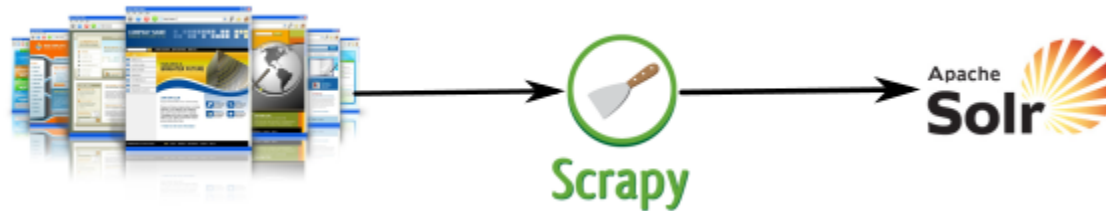
# Abrir no navegador http://127.0.0.1:9001/static/index.html
```

Portia2code

Converter o proxecto de Portia dentro de arañas Scrapy.
Só proxectos con Portia 2 soportados.

Introducing Portia2Code (Blogue Scrapinghub) [↗](#)

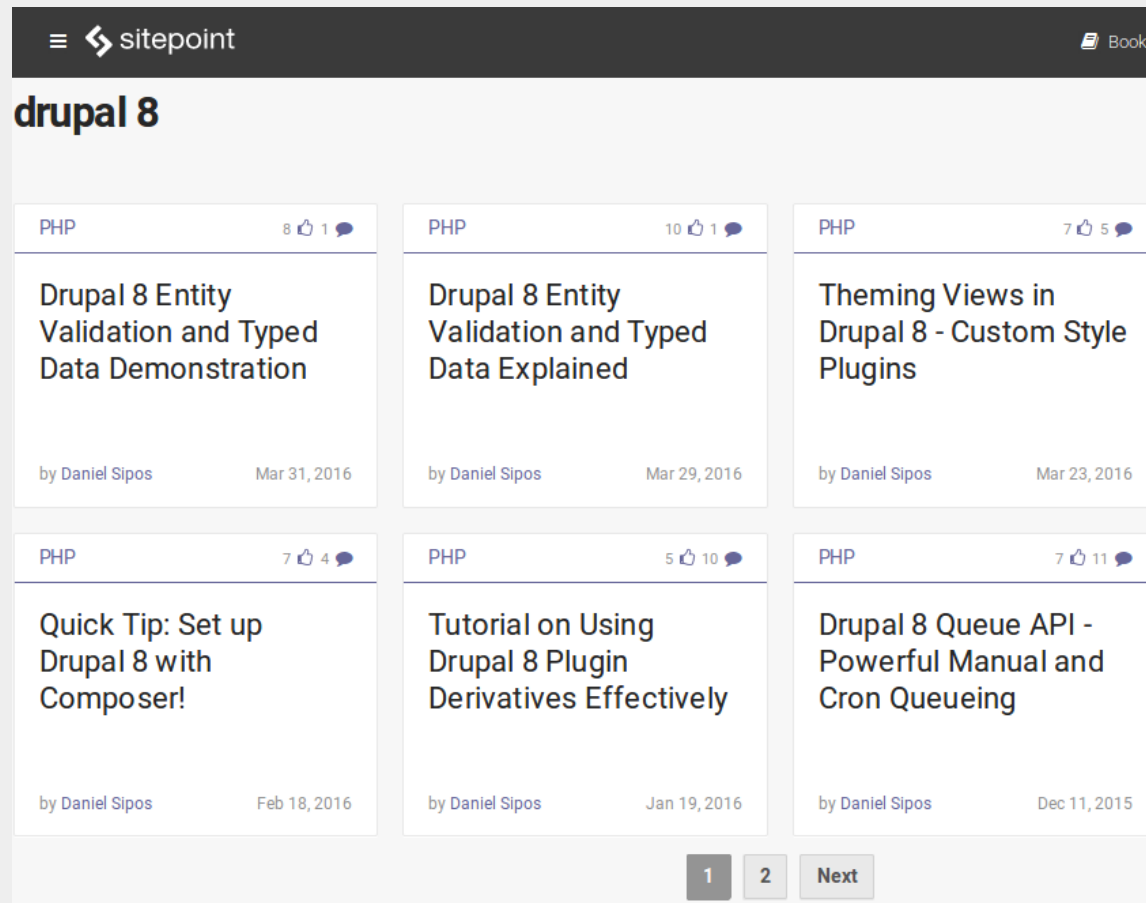
Indexar conteúdos



Pipeline de Scrapy para Solr [↗](#)

DEMO

A demo inclúe **unha araña con Scrapy** que recupera os artigos de **Drupal 8 de Sitepoint** [↗](#) e posterior integración nunha instalación limpa de Drupal.



Probando a shell

```
scrapy shell https://www.sitepoint.com/tag/drupal-8/
```

drupal8

PHP

8 1

h1.article_title | 258x90

Drupal 8 Entity
Validation and Typed
Data Demonstration

by Daniel Sipos

Mar 31, 2016

PHP

10 1

Drupal 8 Entity
Validation and Typed
Data Explained

by Daniel Sipos

Mar 29, 2016

Console Sources Network Performance Memory Application Security Audits

```
"article-list_item tile post-tile">
class="article article--micro category-php " data-disqus-id="https://www.sitepoint.com/drupal-8-entity-validation-
class="article_category"> </header>
on c
▼ <h1 class="article_title">
href = <https://www.sitepoint.com/drupal-8-entity-validation-demonstration/>Drupal 8 Entity Validatio
>
class="contributor article_contributor">...</div>
class="article_meta-data">...</div>
.on>
e>
```

Show me your code



Esta presentación está disponible en:
vifito.github.io/scrappy-for-drupal/ 

E no repositorio: github.com/vifito/scrappy-for-drupal 

```
github.com/vifito/scrappy-for-drupal
├── demo
│   ├── module
│   │   └── gdgourense
│   └── scrapy
│       └── gdgourense
```

Contido da Demo

Un proxecto Scrapy cunha araña de tipo Basic e un **módulo de Drupal** cun comando de Drush para inserir os artigos.

No directorio scrapy executar `launch.sh` ou:

```
$ scrapy crawl sitepoint -o drupal.json --logfile=drupal.log
```

Copiar o ficheiro `drupal.json` e o directorio `assets` á raíz dunha instalación drupal e executar:

```
$ drush gdgourense-import drupal.json --assets=./assets
```


Mans a obra

A imaxe wadmiraal/drupal contén unha instalación de drupal con drush, drupal console, mailhog, ...

```
cd scrapy-for-drupal/demo
# Arrincar un contedor
docker run --name gdgouense -d \
  -v `pwd`/module:/var/www/modules/custom \
  -v `pwd`/scrapy:/var/www/scrapy \
  -p 8080:80 wadmiraal/drupal

# Lanzamos Scrapy para cargar contidos
cd scrapy
./launch.sh

# Acceso ao contedor e executar drush
docker exec -it gdgouense bash
cd /var/www/scrapy
drush -y en gdgouense file_entity
drush gdgouense-import `pwd`/drupal.json --assets=`pwd`/assets
```

Referencias Scrapy

- Páxina de Scrapy
- Proxectos de Scrapinghub en Github (scrapy, portia, portia2code, splash, ...)
- Documentación de Splash
- Scrapy empregando TOR e alternando User-agent
- Indexando sitios con Scrapy e Solr
- Proxectos de Portia2code dentro de scrapy
- Unit 5: Scraping JavaScript based pages
- Tutoriais de Scrapy

Referencias Drupal 8

- Páxina oficial de Drupal
- Drush
- Drupal Console
- Expoñer entidades como API REST

Grazas pola atención