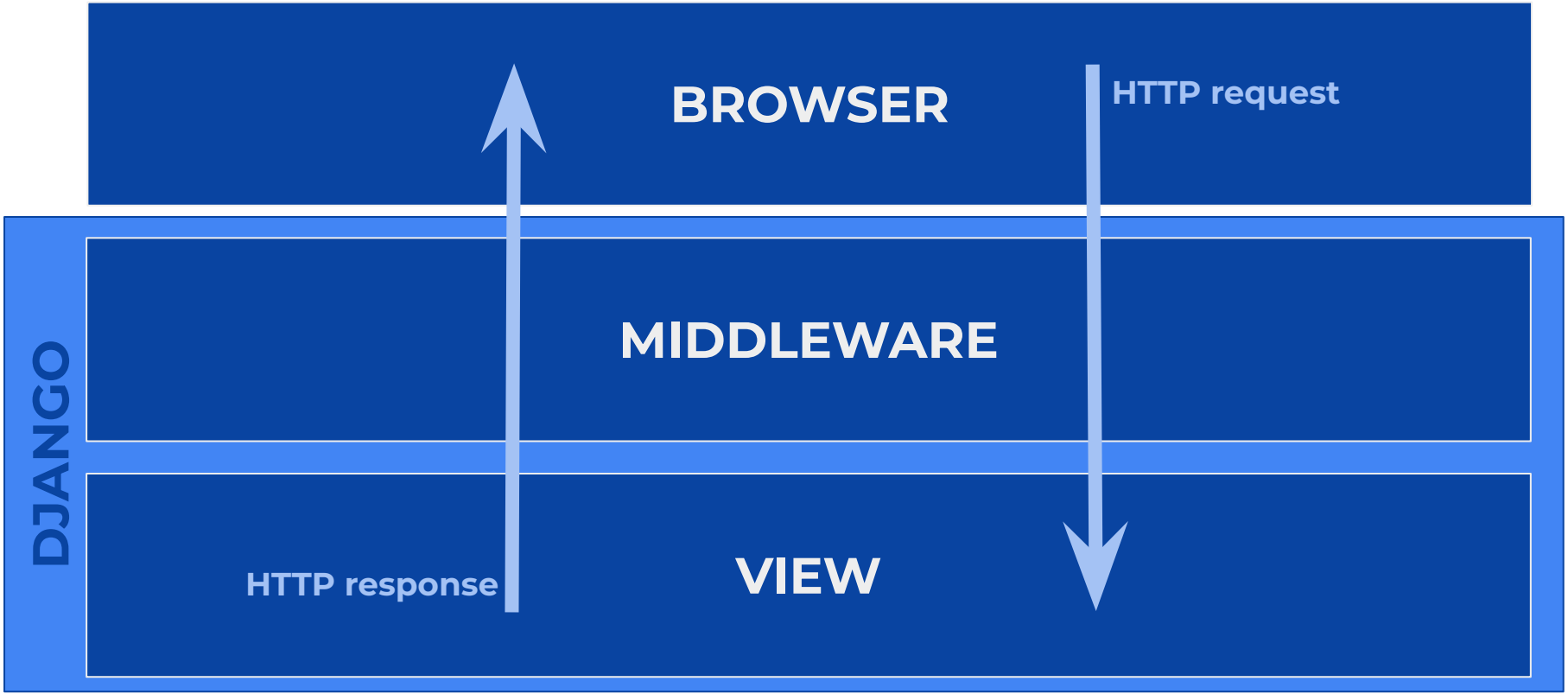
Digital Career Institute

Python Course - Django Advanced Features

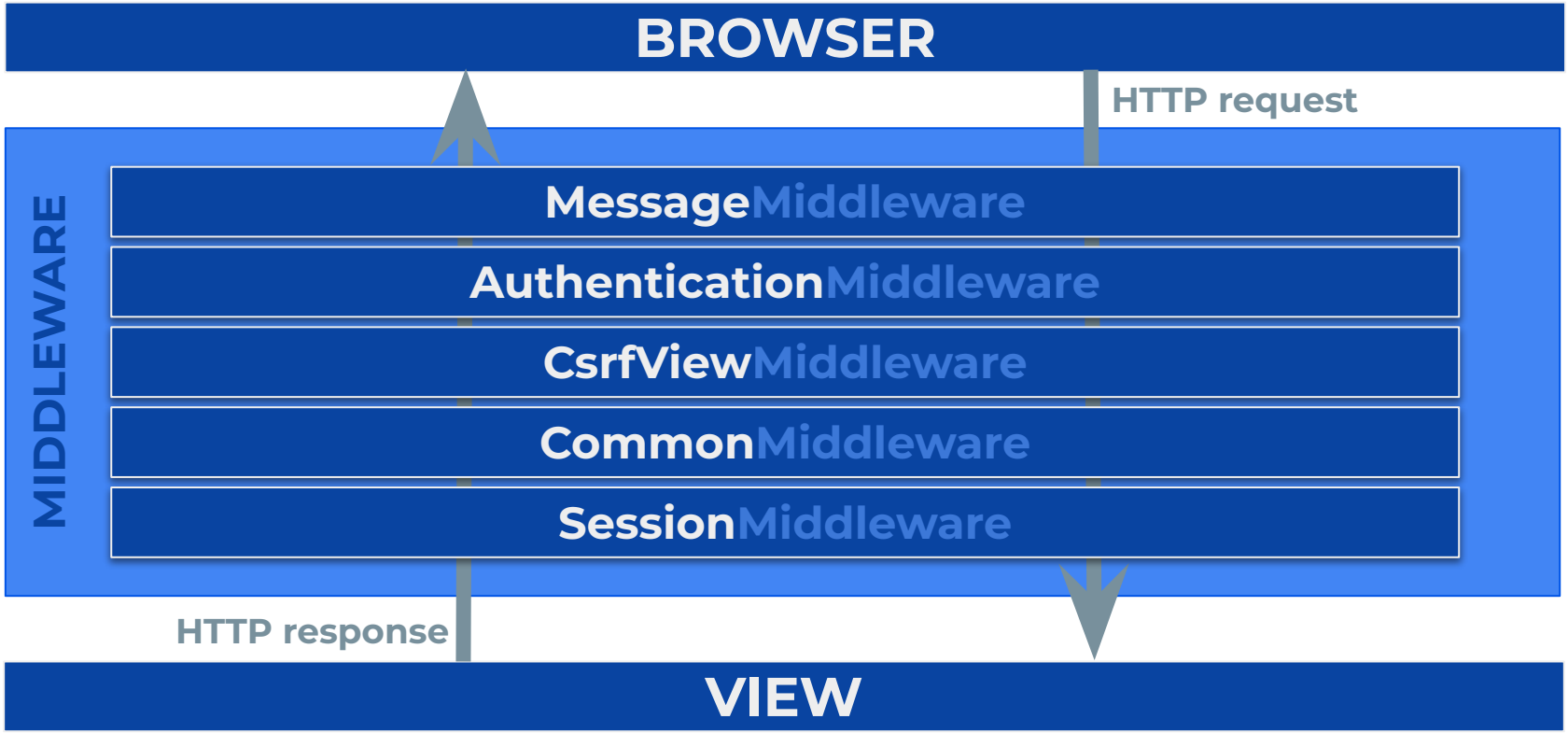
Digital Career Institute

Middleware

What are Middleware



How Middleware Work

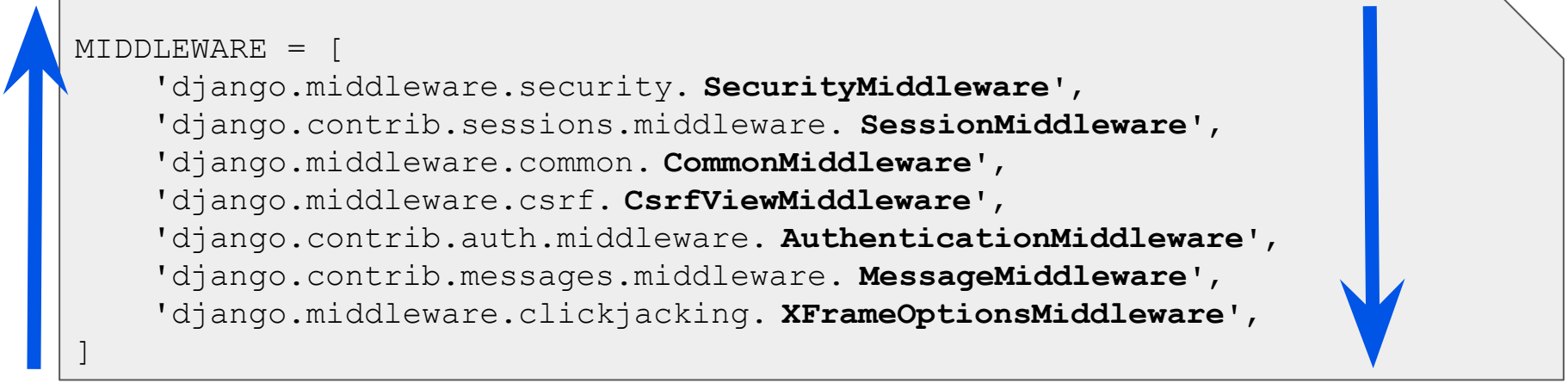


Activating Middleware

Middleware can be activated with the constant **`settings.MIDDLEWARE`**.

hello/settings.py

```
MIDDLEWARE = [
    'django.middleware.security. SecurityMiddleware',
    'django.contrib.sessions.middleware. SessionMiddleware',
    'django.middleware.common. CommonMiddleware',
    'django.middleware.csrf. CsrfViewMiddleware',
    'django.contrib.auth.middleware. AuthenticationMiddleware',
    'django.contrib.messages.middleware. MessageMiddleware',
    'django.middleware.clickjacking. XFrameOptionsMiddleware',
]
```



Built-in Middleware

SecurityMiddleware provides various layers of security.

hello/settings.py

```
MIDDLEWARE = [
    'django.middleware.security. SecurityMiddleware',
]
SECURE_CONTENT_TYPE_NOSNIFF = True
SECURE_HSTS_INCLUDE_SUBDOMAINS = False
SECURE_HSTS_PRELOAD = False
SECURE_HSTS_SECONDS = 0
SECURE_REDIRECT_EXEMPT = []
SECURE_REFERRER_POLICY = "same-origin"
SECURE_SSL_HOST = None
SECURE_SSL_REDIRECT = False
```

SecurityMiddleware

The **nosniff** header forces the browser to use the content-type indicated.

hello/settings.py

```
MIDDLEWARE = [
    'django.middleware.security.SecurityMiddleware',
]
SECURE_CONTENT_TYPE_NOSNIFF = True
```

Some browsers try to guess the content-type of the response.

This is helpful for improperly configured servers, but it can also pose a security risk.

SecurityMiddleware

The **HTTP Strict Transport Security** header instructs the browser to refuse any insecure connection.

hello/settings.py

```
MIDDLEWARE = [
    'django.middleware.security.SecurityMiddleware',
]
SECURE_HSTS_INCLUDE_SUBDOMAINS = False
SECURE_HSTS_PRELOAD = False
SECURE_HSTS_SECONDS = 0
```

Refusing to even emit a request to an insecure source will prevent man-in-the-middle attacks.

SecurityMiddleware

The **Referrer** header tells the server where the request came from.

hello/settings.py

```
MIDDLEWARE = [
    'django.middleware.security.SecurityMiddleware',
]
SECURE_REFERRER_POLICY = "same-origin"
```

The referrer con be used to restrict the access to some assets (to only requests coming from the same server, for instance).

The referrer can be indicated with a full URL or with the origin alone (the host).

SecurityMiddleware

When getting a request to an insecure resource (HTTP) Django can be instructed to redirect to the secure version (HTTPS).

hello/settings.py

```
MIDDLEWARE = [
    'django.middleware.security.SecurityMiddleware',
]
SECURE_SSL_REDIRECT = False
SECURE_SSL_HOST = None
SECURE_REDIRECT_EXEMPT = []
```

If the name of the host is different (for instance, different subdomain), it can be indicated in **`SECURE_SSL_HOST`**.

Regular expressions can be used in **`SECURE_REDIRECT_EXEMPT`** to define exceptions to the redirection.

SessionMiddleware

hello/settings.py

```
MIDDLEWARE = [
    'django.contrib.sessions.middleware. SessionMiddleware',
]
```

`SessionMiddleware` provides the features to enable session support.

CommonMiddleware

CommonMiddleware can be used to control our unique URLs.

hello/settings.py

```
MIDDLEWARE = [
    'django.middleware.common. CommonMiddleware',
]
APPEND_SLASH = True
PREPEND_WWW = False
```

The **settings.py** options **APPEND_SLASH** and **PREPEND_WWW** can be used to standardize and provide single points of entry for each content.

../shop/browse → **APPEND_SLASH** → ../shop/browse/

https://domain.com/ → **PREPEND_WWW** → https://www.domain.com/

CsrfViewMiddleware

hello/settings.py

```
MIDDLEWARE = [
    'django.middleware.csrf.CsrfViewMiddleware',
]
```

`CsrfViewMiddleware` provides protection against **Cross Site Request Forgery**.

AuthenticationMiddleware

`AuthenticationMiddleware` adds the **`user`** attribute to the **`request`** object in views.

hello/settings.py

```
MIDDLEWARE = [
    'django.contrib.auth.middleware. AuthenticationMiddleware',
]
```

Django comes with a built-in user system.

MessageMiddleware

hello/settings.py

```
MIDDLEWARE = [
    'django.contrib.messages.middleware. MessageMiddleware',
]
```

`MessageMiddleware` provides features to manage flash messages.

XFrameOptionsMiddleware

`XFrameOptionsMiddleware` helps prevent attacks using **`iframe`** HTML elements.

hello/settings.py

```
MIDDLEWARE = [
    'django.middleware.clickjacking. XFrameOptionsMiddleware',
]
X_FRAME_OPTIONS = "DENY"
```

Using **`X_FRAME_OPTIONS = "SAMEORIGIN"`** will let the page load in an iframe if the origin of the parent page is the same.

Cache Middleware

hello/settings.py

```
MIDDLEWARE = [
    'django.middleware.cache.UpdateCacheMiddleware',
    ...
    'django.middleware.cache.FetchFromCacheMiddleware',
]
CACHE_MIDDLEWARE_SECONDS = 0
```

`UpdateCacheMiddleware` and **`FetchFromCacheMiddleware`** provide the ability to cache each page in the site.

Custom Middleware

Middleware can be defined with closure functions.

hello/middleware.py

```
def simple_middleware(get_response):
    # One-time configuration and initialization.
    def middleware(request):
        # Code to be executed for each request before
        # the view (and later middleware) are called.
        response = get_response(request)
        # Code to be executed for each request/response after
        # the view is called.
        return response
    return middleware
```

Custom Middleware

The **get_response** callable is the next middleware in the sequence, or the view itself if this is the last middleware.

hello/middleware.py

```
def first_middleware(get_response):
    def middleware(request):
        response = get_response(request)
        return response
    return middleware

def last_middleware(get_response):
    def middleware(request):
        response = get_response(request)
        return response
    return middleware
```

This instruction will call the next middleware.

This instruction will call the view if this is the last middleware in the sequence.

Class-Based Custom Middleware

Middleware can also be defined with classes.

hello/middleware.py

```
class SimpleMiddleware:
    def __init__(self, get_response):
        self.get_response = get_response
        # One-time configuration and initialization.
    def __call__(self, request):
        # Code to be executed for each request before
        # the view (and later middleware) are called.
        response = self.get_response(request)
        # Code to be executed for each request/response after
        # the view is called.
        return response
```

Class-Based Custom Middleware

Class-based middleware have additional hookups.

hello/middleware.py

```
class SimpleMiddleware:
    ...
    def process_view(self, request, view_func, view_args, view_kwargs):
        """Called right before calling the view.
        If this method returns an HttpResponse, it will not call
        the view nor execute any other process_view. Only if this
        returns None, it will keep executing the rest of the calls.
        """
        return HttpResponse("Prevent view execution")
```

Class-Based Custom Middleware

Class-based middleware have additional hookups.

hello/middleware.py

```
class SimpleMiddleware:
    ...
    def process_exception(self, request, exception):
        """Called right after calling the view, if it produced an
        exception.
        If this method returns an HttpResponse, it will not process the
        rest of the middleware. If no middleware returns a response, it
        will use the default exception handling.
        """
        return HttpResponse("Override view response")
```

Class-Based Custom Middleware

Class-based middleware have additional hookups.

hello/middleware.py

```
class SimpleMiddleware:
    ...
    def process_template_response(self, request, response):
        """Called right after calling the view, if the response is a
        template.
        This method must return a TemplateResponse, or an object that
        implements a render() method that returns an HttpResponse.
        """
        # response is a TemplateResponse, or equivalent.
        return response
```

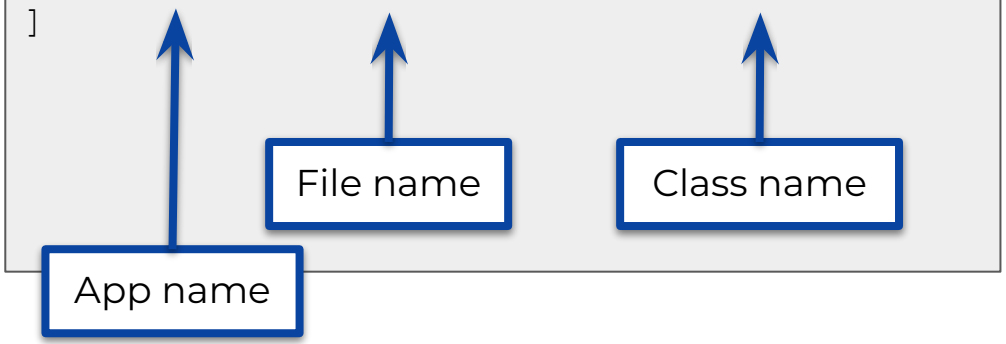
Middleware: Directory Tree

Middleware can be placed anywhere in our directory tree.

```
+hello
  + hello
    - middleware.py
  + shop
    - middleware.py
    - cool_middle.py
```

hello/settings.py

```
MIDDLEWARE = [
    'hello.middleware.SimpleMiddleware',
    'shop.middleware.ShopMiddleware',
    'shop.cool_middle.CoolMiddleware',
]
```



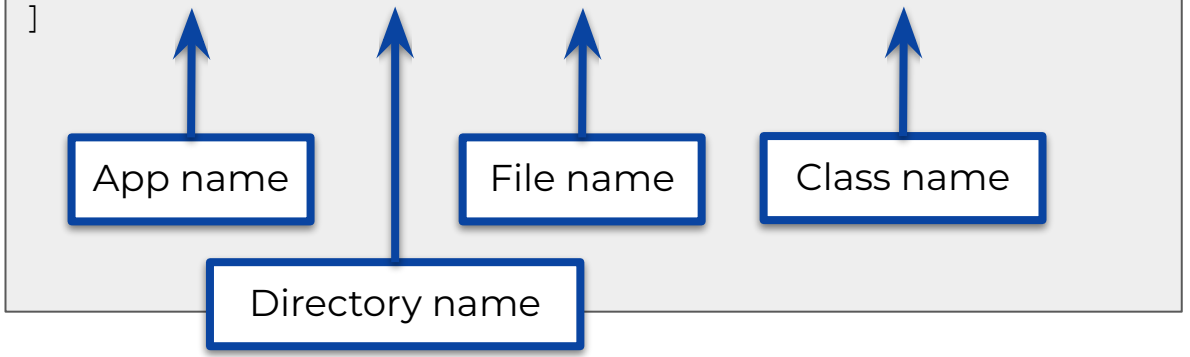
Middleware: Directory Tree

Middleware can be placed anywhere in our directory tree.

```
+hello
  + hello
    + middleware
      - middle1.py
      - middle2.py
```

hello/settings.py

```
MIDDLEWARE = [
    'hello.middleware.middle1.SimpleMiddleware',
    'hello.middleware.middle2.SimpleMiddleware',
]
```



Activating Custom Middleware

The location of our middleware in the **`settings.MIDDLEWARE`** sequence will depend on what our middleware needs.

hello/settings.py

```
MIDDLEWARE = [
    'django.middleware.security.SecurityMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.common.CommonMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
    'django.middleware.clickjacking.XFrameOptionsMiddleware',
    'hello.middleware.SimpleMiddleware',
]
```

Contact Details
DCI Digital Career Institute gGmbH

Digital Career Institute