Pyramid Web Framework Cheat Sheet

A quick reference guide to the Pyramid web framework for Python, covering configuration, views, routing, and more.

Core Concepts & Configuration

Basic Setup

Installation:

```
pip install pyramid
```

Project Structure (minimal):

```
myproject/
├── __init__.py
├── views.py
└── models.py
```

Configuration:

Using a `Configurator` to set up the application.

```
from pyramid.config import Configurator
from pyramid.response import Response

def hello_world(request):
    return Response('Hello World!')

with Configurator() as config:
    config.add_route('home', '/')
    config.add_view(hello_world, route_name='home')
    app = config.make_wsgi_app()
```

Configuration Options

`config.include(module)`	Include a configuration snippet from another module.
`config.scan()`	Scan a package for configuration directives (e.g., views).
`config.set_request_property(lambda request: ..., name='...')`	Adds a property to the request object.
`config.add_static_view(name='static', path='myproject:static')`	Serve static files from a directory.

Routes & Views

Routes

Routes map URLs to view callables.

Example:

```
config.add_route('blog_entry', '/blog/{id}')
```

Route parameters are available in the `request.matchdict`.

Example:

```
def blog_view(request):
    entry_id = request.matchdict['id']
    # ...
```

You can use `pyramid.url.route_url` to generate URLs based on routes.

```
from pyramid.url import route_url

def my_view(request):
    url = route_url('blog_entry', request, id=123)
    return Response(f'URL: {url}')
```

Views

Views are callables that handle requests.

Example:

```
from pyramid.view import view_config
from pyramid.response import Response

@view_config(route_name='home', renderer='templates/mytemplate.pt')
def my_view(request):
    return {'project':'MyProject'}
```

Views can return a `Response` object directly or a dictionary for rendering with a template.

```
from pyramid.response import Response

def my_view(request):
    return Response('Hello!')
```

View lookups are based on:

- `route_name`
- `view_name`
- `context` (the object the view is being rendered for)
- `request_method` (GET, POST, etc.)
- `accept` (Accept header)

View Predicates

`request_method`	Matches a specific HTTP method (e.g., `'POST'`)
`accept`	Matches the `Accept` header (e.g., `'application/json'`)
`xhr`	Matches if the request is an AJAX request (`True` or `False`)
`custom_predicate`	Use custom predicate function

Templates & Static Assets

Template Rendering

Pyramid supports various templating languages, including:

- Chameleon
- Jinja2
- Mako

Configure a renderer in the `view_config` or using `config.add_view`.

```
@view_config(route_name='home', renderer='templates/mytemplate.pt')
def my_view(request):
    return {'project':'MyProject'}
```

Chameleon:

```
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:tal="http://xml.zope.org/namespaces/tal">
<head>
  <title tal:content="project" />
</head>
<body>
  <h1>Welcome to <span tal:content="project" />!</h1>
</body>
</html>
```

Jinja2:

```
<!DOCTYPE html>
<html>
<head>
    <title>{{ project }}</title>
</head>
<body>
    <h1>Welcome to {{ project }}!</h1>
</body>
</html>
```

Static Assets

Serving static assets (CSS, JavaScript, images) using `config.add_static_view`.

```
config.add_static_view(name='static', path='myproject:static')
```

Accessing static assets in templates:

```
<link rel="stylesheet"
href="${request.static_url('myproject:static/style.css')}" type="text/css"
/>
```

Make sure to create the static directory.

Request & Response Objects

Request Object

The `request` object provides access to incoming request data.

- `request.params`: GET/POST parameters
- `request.matchdict`: Route parameters
- `request.POST`: POST data
- `request.GET`: GET data
- `request.cookies`: Cookies
- `request.session`: Session object

Example:

```
def my_view(request):
    name = request.params.get('name', 'Guest')
    return Response(f'Hello, {name}!')
```

Response Object

The `Response` object is used to return data to the client.

- `Response(body='...')`: Set the response body
- `Response(status=200)`: Set the HTTP status code
- `Response(content_type='text/html')`: Set the Content-Type header

Example:

```
from pyramid.response import Response

def my_view(request):
    response = Response('Hello, World!')
    response.status = '200 OK'
    response.content_type = 'text/plain'
    return response
```

You can also set cookies on the response object:

```
response.set_cookie('mycookie',
value='somevalue', max_age=3600)
```

Sessions

Pyramid provides session support.

To enable it, configure a session factory (e.g., `SignedCookieSessionFactory`):

```
from pyramid.config import Configurator
from pyramid.session import
SignedCookieSessionFactory

my_session_factory =
SignedCookieSessionFactory('itsaseekreet')

with
Configurator(session_factory=my_session_factor
y) as config:
    # ...
```

Accessing the session:

```
def my_view(request):
    session = request.session
    session['foo'] = 'bar'
    return Response('Session updated!')
```