Web Controllers

Controllers

Controllers need to provide extensibility, much like `Model` (orm.html#odoo.models.Model), but can't use the same mechanism as the pre-requisites (a database with loaded modules) may not be available yet (e.g. no database created, or no database selected).

Controllers thus provide their own extension mechanism, separate from that of models:

Controllers are created by inheriting (https://docs.python.org/3/tutorial/classes.html#tut-inheritance) from `Controller`. Routes are defined through methods decorated with `route()`:

```
class MyController(odoo.http.Controller):
    @route('/some_url', auth='public')
    def handler(self):
        return stuff()
```

To *override* a controller, inherit (https://docs.python.org/3/tutorial/classes.html#tut-inheritance) from its class and override relevant methods, re-exposing them if necessary:

```
class Extension(MyController):
    @route()
    def handler(self):
        do_before()
        return super(Extension, self).handler()
```

> decorating with `route()` is necessary to keep the method (and route) visible: if the method is redefined without decorating, it will be "unpublished"
>
> the decorators of all methods are combined, if the overriding method's decorator has no argument all previous ones will be kept, any provided argument will override previously defined ones e.g.:
>
> ```
> class Restrict(MyController):
>     @route(auth='user')
>     def handler(self):
>         return super(Restrict, self).handler()
> ```
>
> will change `/some_url` from public authentication to user (requiring a log-in)

API

Routing

`odoo.http.route(`*`route=None`*`, `*`**kw`*`)`
(https://github.com/odoo/odoo/blob/14.0/odoo/http.py#L431)

Decorator marking the decorated method as being a handler for requests. The method must be part of a subclass of `Controller` .

Parameters:

- **`route`** – string or array. The route part that will determine which http requests will match the decorated method. Can be a single string or an array of strings. See werkzeug's routing documentation for the format of route expression (http://werkzeug.pocoo.org/docs/routing/ (http://werkzeug.pocoo.org/docs/routing/)).
- **`type`** – The type of request, can be `'http'` or `'json'` .
- **`auth`** –
  The type of authentication method, can on of the following:
  `user` : The user must be authenticated and the current request will perform using the rights of the user.

  `public` : The user may or may not be authenticated. If she isn't, the current request will perform using the shared Public user.

  `none` : The method is always active, even if there is no database. Mainly used by the framework and authentication modules. There request code will not have any facilities to access the database nor have any configuration indicating the current database nor the current user.
- **`methods`** – A sequence of http methods this route applies to. If not specified, all methods are allowed.
- **`cors`** – The Access-Control-Allow-Origin cors directive value.
- **`csrf`** (**`bool`** (https://docs.python.org/3/library/functions.html#bool)) –
  Whether CSRF protection should be enabled for the route.
  Defaults to `True` . See CSRF Protection for more.

⚠ CSRF Protection

New in version 9.0.
Odoo implements token-based **CSRF protection (https://en.wikipedia.org/wiki/CSRF)**.

CSRF protection is enabled by default and applies to *UNSAFE* HTTP methods as defined by **RFC 7231 (https://tools.ietf.org/html/rfc7231.html)** (all methods other than `GET` , `HEAD` , `TRACE` and `OPTIONS`).

CSRF protection is implemented by checking requests using unsafe methods for a value called `csrf_token` as part of the request's form data. That value is removed from the form as part of the validation and does not have to be taken in account by your own form processing.

When adding a new controller for an unsafe method (mostly POST for e.g. forms):

- if the form is generated in Python, a csrf token is available via `request.csrf_token() <odoo.http.WebRequest.csrf_token()` , the `request` object is available by default in QWeb (python) templates, it may have to be added explicitly if you are not using QWeb.
- if the form is generated in Javascript, the CSRF token is added by default to the QWeb (js) rendering context as `csrf_token` and is otherwise available as `csrf_token` on the `web.core` module:

  ```
  require('web.core').csrf_token
  ```
- if the endpoint can be called by external parties (not from Odoo) as e.g. it is a REST API or a **webhook (https://en.wikipedia.org/wiki/Webhook)**, CSRF protection must be disabled on the endpoint. If possible, you may want to implement other methods of request validation (to ensure it is not called by an unrelated third-party).

Request

The request object is automatically set on `odoo.http.request` at the start of the request

class **`odoo.http.WebRequest(`*`httprequest`*`)`** [(https://github.com/odoo/odoo/blob/14.0/odoo/http.py#L177)](https://github.com/odoo/odoo/blob/14.0/odoo/http.py#L177)

Parent class for all Odoo Web request types, mostly deals with initialization and setup of the request object (the dispatching itself has to be handled by the subclasses)

Parameters:

httprequest (`werkzeug.wrappers.BaseRequest` (https://werkzeug.palletsprojects.com/en/1.0.x/wrappers/#werkzeug.wrappers.BaseRequest)) – a wrapped werkzeug Request object

`httprequest`

the original `werkzeug.wrappers.Request` (https://werkzeug.palletsprojects.com/en/1.0.x/wrappers/#werkzeug.wrappers.Request) object provided to the request

`params`

`Mapping` of request parameters, not generally useful as they're provided directly to the handler method as keyword arguments

property **`cr`**

`Cursor` initialized for the current method call.

Accessing the cursor when the current request uses the `none` authentication will raise an exception.

property **`context`**

`Mapping` of context values for the current request

property **`env`**

The `Environment` bound to current request.

`session`

`OpenERPSession` holding the HTTP session data for the current http session

property **`registry`**

The registry to the database linked to this request. Can be `None` if the current request uses the `none` authentication.

Deprecated since version 8.0: use `env` (orm.html#odoo.models.env)

***property* db**

The database linked to this request. Can be `None` if the current request uses the `none` authentication.

csrf_token(*time_limit=None*) (https://github.com/odoo/odoo/blob/14.0/odoo/http.py#L387)

Generates and returns a CSRF token for the current session

Parameters:

time_limit (`int | None`) – the CSRF token validity period (in seconds), or `None` for the token to be valid as long as the current user session is (the default)

Returns:

ASCII token string

***class* odoo.http.HttpRequest(**args*) (https://github.com/odoo/odoo/blob/14.0/odoo/http.py#L710)**

Handler for the `http` request type.

matched routing parameters, query string parameters, form (http://www.w3.org/TR/html401/interact/forms.html#h-17.13.4.2) parameters and files are passed to the handler method as keyword arguments.

In case of name conflict, routing parameters have priority.

The handler method's result can be:

a falsy value, in which case the HTTP response will be an HTTP 204 (http://tools.ietf.org/html/rfc7231#section-6.3.5) (No Content)

a werkzeug Response object, which is returned as-is

a `str` or `unicode` , will be wrapped in a Response object and interpreted as HTML

make_response(*data*, *headers=None*, *cookies=None*) (https://github.com/odoo/odoo/blob/14.0/odoo/http.py#L811)

Helper for non-HTML responses, or HTML responses with custom response headers or cookies.

While handlers can just return the HTML markup of a page they want to send as a string if non-HTML data is returned they need to create a complete response object, or the returned data will not be correctly interpreted by the clients.

Parameters:

- **`data`** (**`basestring`**) – response body
- **`headers`** (**`[(name, value)]`**) – HTTP headers to set on the response
- **`cookies`** (**`collections.Mapping`**) – cookies to set on the client

`not_found(`*`description=None`*`)` **(https://github.com/odoo/odoo/blob/14.0/odoo/http.py#L849)**

Shortcut for a HTTP 404 (http://tools.ietf.org/html/rfc7231#section-6.5.4) (Not Found) response

`render(`*`template`*`, `*`qcontext=None`*`, `*`lazy=True`*`, **`*`kw`*`)` **(https://github.com/odoo/odoo/blob/14.0/odoo/http.py#L831)**

Lazy render of a QWeb template.

The actual rendering of the given template will occur at then end of the dispatching. Meanwhile, the template and/or qcontext can be altered or even replaced by a static response.

Parameters:

- **`template`** (**`basestring`**) – template to render
- **`qcontext`** (**`dict`** (https://docs.python.org/3/library/stdtypes.html#dict)) – Rendering context to use
- **`lazy`** (**`bool`** (https://docs.python.org/3/library/functions.html#bool)) – whether the template rendering should be deferred until the last possible moment
- **`kw`** – forwarded to werkzeug's Response object

`class` **`odoo.http.JsonRequest(*`*`args`*`)`** **(https://github.com/odoo/odoo/blob/14.0/odoo/http.py#L552)**

Request handler for JSON-RPC 2 (http://www.jsonrpc.org/specification) over HTTP

- **`method`** is ignored
- **`params`** must be a JSON object (not an array) and is passed as keyword arguments to the handler method
- the handler method's result is returned as JSON-RPC **`result`** and wrapped in the JSON-RPC Response (http://www.jsonrpc.org/specification#response_object)

Sucessful request:

```
--> {"jsonrpc": "2.0",
     "method": "call",
     "params": {"context": {},
                "arg1": "val1" },
     "id": null}

<-- {"jsonrpc": "2.0",
     "result": { "res1": "val1" },
     "id": null}
```

Request producing a error:

```
--> {"jsonrpc": "2.0",
     "method": "call",
     "params": {"context": {},
                "arg1": "val1" },
     "id": null}

<-- {"jsonrpc": "2.0",
     "error": {"code": 1,
               "message": "End user error message.",
               "data": {"code": "codestring",
                        "debug": "traceback" } },
     "id": null}
```

Response

class **odoo.http.Response(**args*, ***kw*) (https://github.com/odoo/odoo/blob/14.0/odoo/http.py#L1185)**

Response object passed through controller route chain.

In addition to the `werkzeug.wrappers.Response` (https://werkzeug.palletsprojects.com/en/1.0.x/wrappers/#werkzeug.wrappers.Response) parameters, this class's constructor can take the following additional parameters for QWeb Lazy Rendering.

Parameters:

- **`template`** (**`basestring`**) – template to render
- **`qcontext`** (**`dict`** (https://docs.python.org/3/library/stdtypes.html#dict)) – Rendering context to use
- **`uid`** (**`int`** (https://docs.python.org/3/library/functions.html#int)) – User id to use for the ir.ui.view render call, **`None`** to use the request's user (the default)

these attributes are available as parameters on the Response object and can be altered at any time before rendering

Also exposes all the attributes and methods of `werkzeug.wrappers.Response` (https://werkzeug.palletsprojects.com/en/1.0.x/wrappers/#werkzeug.wrappers.Response).

render() (https://github.com/odoo/odoo/blob/14.0/odoo/http.py#L1230)

Renders the Response’s template, returns the result

`flatten()` [(https://github.com/odoo/odoo/blob/14.0/odoo/http.py#L1237)](https://github.com/odoo/odoo/blob/14.0/odoo/http.py#L1237)

Forces the rendering of the response’s template, sets the result as response body and unsets **`template`**