:mod:`xmlrpc.server` --- Basic XML-RPC servers

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]xmlrpc.server.rst, line 1); *backlink*

Unknown interpreted text role "mod".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]xmlrpc.server.rst, line 4)

Unknown directive type "module".

```
.. module:: xmlrpc.server
   :synopsis: Basic XML-RPC server implementations.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]xmlrpc.server.rst, line 7)

Unknown directive type "moduleauthor".

```
.. moduleauthor:: Brian Quinlan <brianq@activestate.com>
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]xmlrpc.server.rst, line 8)

Unknown directive type "sectionauthor".

```
.. sectionauthor:: Fred L. Drake, Jr. <fdrake@acm.org>
```

Source code: :source:`Lib/xmlrpc/server.py`

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]xmlrpc.server.rst, line 10); *backlink*

Unknown interpreted text role "source".

The :mod:`xmlrpc.server` module provides a basic server framework for XML-RPC servers written in Python. Servers can either be free standing, using :class:`SimpleXMLRPCServer`, or embedded in a CGI environment, using :class:`CGIXMLRPCRequestHandler`.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]xmlrpc.server.rst, line 14); *backlink*

Unknown interpreted text role "mod".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]xmlrpc.server.rst, line 14); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]xmlrpc.server.rst, line 14); *backlink*

Unknown interpreted text role "class".

Warning

The :mod:`xmlrpc.server` module is not secure against maliciously constructed data. If you need to parse untrusted or unauthenticated data see :ref:`xml-vulnerabilities`.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc]

[library]xmlrpc.server.rst, line 22); *backlink*

Unknown interpreted text role "mod".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]xmlrpc.server.rst, line 22); *backlink*

Unknown interpreted text role "ref".

Create a new server instance. This class provides methods for registration of functions that can be called by the XML-RPC protocol. The *requestHandler* parameter should be a factory for request handler instances; it defaults to :class:`SimpleXMLRPCRequestHandler`. The *addr* and *requestHandler* parameters are passed to the :class:`socketserver.TCPServer` constructor. If *logRequests* is true (the default), requests will be logged; setting this parameter to false will turn off logging. The *allow_none* and *encoding* parameters are passed on to :mod:`xmlrpc.client` and control the XML-RPC responses that will be returned from the server. The *bind_and_activate* parameter controls whether :meth:`server_bind` and :meth:`server_activate` are called immediately by the constructor; it defaults to true. Setting it to false allows code to manipulate the *allow_reuse_address* class variable before the address is bound. The *use_builtin_types* parameter is passed to the :func:`~xmlrpc.client.loads` function and controls which types are processed when date/times values or binary data are received; it defaults to false.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]xmlrpc.server.rst, line 31); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]xmlrpc.server.rst, line 31); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]xmlrpc.server.rst, line 31); *backlink*

Unknown interpreted text role "mod".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]xmlrpc.server.rst, line 31); *backlink*

Unknown interpreted text role "meth".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]xmlrpc.server.rst, line 31); *backlink*

Unknown interpreted text role "meth".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]xmlrpc.server.rst, line 31); *backlink*

Unknown interpreted text role "func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]xmlrpc.server.rst, line 47)

Unknown directive type "versionchanged".

```
.. versionchanged:: 3.3
   The *use_builtin_types* flag was added.
```

Create a new instance to handle XML-RPC requests in a CGI environment. The *allow_none* and *encoding* parameters are passed on to :mod:`xmlrpc.client` and control the XML-RPC responses that will be returned from the server. The *use_builtin_types* parameter is passed to the :func:`~xmlrpc.client.loads` function and controls which types are processed when date/times values or binary data are received; it defaults to false.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]xmlrpc.server.rst, line 54); *backlink*

Unknown interpreted text role "mod".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]xmlrpc.server.rst, line 54); *backlink*

Unknown interpreted text role "func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]xmlrpc.server.rst, line 61)

Unknown directive type "versionchanged".

```
.. versionchanged:: 3.3
   The *use_builtin_types* flag was added.
```

Create a new request handler instance. This request handler supports `POST` requests and modifies logging so that the *logRequests* parameter to the :class:`SimpleXMLRPCServer` constructor parameter is honored.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]xmlrpc.server.rst, line 67); *backlink*

Unknown interpreted text role "class".

SimpleXMLRPCServer Objects

The :class:`SimpleXMLRPCServer` class is based on :class:`socketserver.TCPServer` and provides a means of creating simple, stand alone XML-RPC servers.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]xmlrpc.server.rst, line 77); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]xmlrpc.server.rst, line 77); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]xmlrpc.server.rst, line 82)

Unknown directive type "method".

```
.. method:: SimpleXMLRPCServer.register_function(function=None, name=None)

   Register a function that can respond to XML-RPC requests.  If *name* is given,
   it will be the method name associated with *function*, otherwise
   ``function.__name__`` will be used.  *name* is a string, and may contain
   characters not legal in Python identifiers, including the period character.

   This method can also be used as a decorator.  When used as a decorator,
   *name* can only be given as a keyword argument to register *function* under
   *name*.  If no *name* is given, ``function.__name__`` will be used.

   .. versionchanged:: 3.7
      :meth:`register_function` can be used as a decorator.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]xmlrpc.server.rst, line 97)

Unknown directive type "method".

```
.. method:: SimpleXMLRPCServer.register_instance(instance, allow_dotted_names=False)
```

```
Register an object which is used to expose method names which have not been
registered using :meth:`register_function`.  If *instance* contains a
:meth:`_dispatch` method, it is called with the requested method name and the
parameters from the request.  Its API is ``def _dispatch(self, method, params)``
(note that *params* does not represent a variable argument list).  If it calls
an underlying function to perform its task, that function is called as
``func(*params)``, expanding the parameter list. The return value from
:meth:`_dispatch` is returned to the client as the result.  If *instance* does
not have a :meth:`_dispatch` method, it is searched for an attribute matching
the name of the requested method.

If the optional *allow_dotted_names* argument is true and the instance does not
have a :meth:`_dispatch` method, then if the requested method name contains
periods, each component of the method name is searched for individually, with
the effect that a simple hierarchical search is performed.  The value found from
this search is then called with the parameters from the request, and the return
value is passed back to the client.

.. warning::

   Enabling the *allow_dotted_names* option allows intruders to access your
   module's global variables and may allow intruders to execute arbitrary code on
   your machine.  Only use this option on a secure, closed network.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]xmlrpc.server.rst, line 124)

Unknown directive type "method".

```
.. method:: SimpleXMLRPCServer.register_introspection_functions()

   Registers the XML-RPC introspection functions ``system.listMethods``,
   ``system.methodHelp`` and ``system.methodSignature``.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]xmlrpc.server.rst, line 130)

Unknown directive type "method".

```
.. method:: SimpleXMLRPCServer.register_multicall_functions()

   Registers the XML-RPC multicall function system.multicall.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]xmlrpc.server.rst, line 135)

Unknown directive type "attribute".

```
.. attribute:: SimpleXMLRPCRequestHandler.rpc_paths

   An attribute value that must be a tuple listing valid path portions of the URL
   for receiving XML-RPC requests.  Requests posted to other paths will result in a
   404 "no such page" HTTP error.  If this tuple is empty, all paths will be
   considered valid. The default value is ``('/', '/RPC2')``.
```

SimpleXMLRPCServer Example

Server code:

```
from xmlrpc.server import SimpleXMLRPCServer
from xmlrpc.server import SimpleXMLRPCRequestHandler

# Restrict to a particular path.
class RequestHandler(SimpleXMLRPCRequestHandler):
    rpc_paths = ('/RPC2',)

# Create server
with SimpleXMLRPCServer(('localhost', 8000),
                        requestHandler=RequestHandler) as server:
    server.register_introspection_functions()
```

```
    # Register pow() function; this will use the value of
    # pow.__name__ as the name, which is just 'pow'.
    server.register_function(pow)

    # Register a function under a different name
    def adder_function(x, y):
        return x + y
    server.register_function(adder_function, 'add')

    # Register an instance; all the methods of the instance are
    # published as XML-RPC methods (in this case, just 'mul').
    class MyFuncs:
        def mul(self, x, y):
            return x * y

    server.register_instance(MyFuncs())

    # Run the server's main loop
    server.serve_forever()
```

The following client code will call the methods made available by the preceding server:

```
import xmlrpc.client

s = xmlrpc.client.ServerProxy('http://localhost:8000')
print(s.pow(2,3))  # Returns 2**3 = 8
print(s.add(2,3))  # Returns 5
print(s.mul(5,2))  # Returns 5*2 = 10

# Print list of available methods
print(s.system.listMethods())
```

:meth:`register_function` can also be used as a decorator. The previous server example can register functions in a decorator way:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]xmlrpc.server.rst, line 194); *backlink*

Unknown interpreted text role "meth".

```
from xmlrpc.server import SimpleXMLRPCServer
from xmlrpc.server import SimpleXMLRPCRequestHandler

class RequestHandler(SimpleXMLRPCRequestHandler):
    rpc_paths = ('/RPC2',)

with SimpleXMLRPCServer(('localhost', 8000),
                        requestHandler=RequestHandler) as server:
    server.register_introspection_functions()

    # Register pow() function; this will use the value of
    # pow.__name__ as the name, which is just 'pow'.
    server.register_function(pow)

    # Register a function under a different name, using
    # register_function as a decorator. *name* can only be given
    # as a keyword argument.
    @server.register_function(name='add')
    def adder_function(x, y):
        return x + y

    # Register a function under function.__name__.
    @server.register_function
    def mul(x, y):
        return x * y

    server.serve_forever()
```

The following example included in the :file:`Lib/xmlrpc/server.py` module shows a server allowing dotted names and registering a multicall function.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]xmlrpc.server.rst, line 225); *backlink*

Unknown interpreted text role "file".

Warning

Enabling the *allow_dotted_names* option allows intruders to access your module's global variables and may allow intruders to execute arbitrary code on your machine. Only use this example only within a secure, closed network.

```
import datetime

class ExampleService:
    def getData(self):
        return '42'

    class currentTime:
        @staticmethod
        def getCurrentTime():
            return datetime.datetime.now()

with SimpleXMLRPCServer(("localhost", 8000)) as server:
    server.register_function(pow)
    server.register_function(lambda x,y: x+y, 'add')
    server.register_instance(ExampleService(), allow_dotted_names=True)
    server.register_multicall_functions()
    print('Serving XML-RPC on localhost port 8000')
    try:
        server.serve_forever()
    except KeyboardInterrupt:
        print("\nKeyboard interrupt received, exiting.")
        sys.exit(0)
```

This ExampleService demo can be invoked from the command line:

```
python -m xmlrpc.server
```

The client that interacts with the above server is included in *Lib/xmlrpc/client.py*:

```
server = ServerProxy("http://localhost:8000")

try:
    print(server.currentTime.getCurrentTime())
except Error as v:
    print("ERROR", v)

multi = MultiCall(server)
multi.getData()
multi.pow(2,9)
multi.add(1,2)
try:
    for response in multi():
        print(response)
except Error as v:
    print("ERROR", v)
```

This client which interacts with the demo XMLRPC server can be invoked as:

```
python -m xmlrpc.client
```

CGIXMLRPCRequestHandler

The :class:`CGIXMLRPCRequestHandler` class can be used to handle XML-RPC requests sent to Python CGI scripts.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]xmlrpc.server.rst, line 292); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]xmlrpc.server.rst, line 296)

Unknown directive type "method".

```
.. method:: CGIXMLRPCRequestHandler.register_function(function=None, name=None)

   Register a function that can respond to XML-RPC requests.  If *name* is given,
   it will be the method name associated with *function*, otherwise
   ``function.__name__`` will be used.  *name* is a string, and may contain
   characters not legal in Python identifiers, including the period character.

   This method can also be used as a decorator.  When used as a decorator,
   *name* can only be given as a keyword argument to register *function* under
   *name*.  If no *name* is given, ``function.__name__`` will be used.
```

```
      .. versionchanged:: 3.7
         :meth:`register_function` can be used as a decorator.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]xmlrpc.server.rst, line 311)

Unknown directive type "method".

```
   .. method:: CGIXMLRPCRequestHandler.register_instance(instance)

      Register an object which is used to expose method names  which have not been
      registered using :meth:`register_function`. If  instance contains a
      :meth:`_dispatch` method, it is called with the  requested method name and the
      parameters from the  request; the return value is returned to the client as the
      result. If instance does not have a :meth:`_dispatch` method, it is searched
      for an attribute matching the name of the requested method; if  the requested
      method name contains periods, each  component of the method name is searched for
      individually,  with the effect that a simple hierarchical search is performed.
      The value found from this search is then called with the  parameters from the
      request, and the return value is passed  back to the client.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]xmlrpc.server.rst, line 325)

Unknown directive type "method".

```
   .. method:: CGIXMLRPCRequestHandler.register_introspection_functions()

      Register the XML-RPC introspection functions  ``system.listMethods``,
      ``system.methodHelp`` and  ``system.methodSignature``.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]xmlrpc.server.rst, line 331)

Unknown directive type "method".

```
   .. method:: CGIXMLRPCRequestHandler.register_multicall_functions()

      Register the XML-RPC multicall function ``system.multicall``.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]xmlrpc.server.rst, line 336)

Unknown directive type "method".

```
   .. method:: CGIXMLRPCRequestHandler.handle_request(request_text=None)

      Handle an XML-RPC request. If *request_text* is given, it should be the POST
      data provided by the HTTP server,  otherwise the contents of stdin will be used.
```

Example:

```
class MyFuncs:
    def mul(self, x, y):
        return x * y


handler = CGIXMLRPCRequestHandler()
handler.register_function(pow)
handler.register_function(lambda x,y: x+y, 'add')
handler.register_introspection_functions()
handler.register_instance(MyFuncs())
handler.handle_request()
```

Documenting XMLRPC server

These classes extend the above classes to serve HTML documentation in response to HTTP GET requests. Servers can either be free standing, using :class:`DocXMLRPCServer`, or embedded in a CGI environment, using :class:`DocCGIXMLRPCRequestHandler`.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]xmlrpc.server.rst, line 359); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]xmlrpc.server.rst, line 359); *backlink*

Unknown interpreted text role "class".

Create a new server instance. All parameters have the same meaning as for :class:`SimpleXMLRPCServer`; *requestHandler* defaults to :class:`DocXMLRPCRequestHandler`.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]xmlrpc.server.rst, line 369); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]xmlrpc.server.rst, line 369); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]xmlrpc.server.rst, line 373)

Unknown directive type "versionchanged".

```
.. versionchanged:: 3.3
   The *use_builtin_types* flag was added.
```

Create a new instance to handle XML-RPC requests in a CGI environment.

Create a new request handler instance. This request handler supports XML-RPC POST requests, documentation GET requests, and modifies logging so that the *logRequests* parameter to the :class:`DocXMLRPCServer` constructor parameter is honored.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]xmlrpc.server.rst, line 384); *backlink*

Unknown interpreted text role "class".

DocXMLRPCServer Objects

The :class:`DocXMLRPCServer` class is derived from :class:`SimpleXMLRPCServer` and provides a means of creating self-documenting, stand alone XML-RPC servers. HTTP POST requests are handled as XML-RPC method calls. HTTP GET requests are handled by generating pydoc-style HTML documentation. This allows a server to provide its own web-based documentation.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]xmlrpc.server.rst, line 395); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]xmlrpc.server.rst, line 395); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]xmlrpc.server.rst, line 402)

Unknown directive type "method".

.. method:: DocXMLRPCServer.set_server_title(server_title)

   Set the title used in the generated HTML documentation. This title will be used
   inside the HTML "title" element.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]xmlrpc.server.rst, line 408)

Unknown directive type "method".

.. method:: DocXMLRPCServer.set_server_name(server_name)

   Set the name used in the generated HTML documentation. This name will appear at
   the top of the generated documentation inside a "h1" element.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]xmlrpc.server.rst, line 414)

Unknown directive type "method".

.. method:: DocXMLRPCServer.set_server_documentation(server_documentation)

   Set the description used in the generated HTML documentation. This description
   will appear as a paragraph, below the server name, in the documentation.

DocCGIXMLRPCRequestHandler

The :class:`DocCGIXMLRPCRequestHandler` class is derived from :class:`CGIXMLRPCRequestHandler` and provides a means of creating self-documenting, XML-RPC CGI scripts. HTTP POST requests are handled as XML-RPC method calls. HTTP GET requests are handled by generating pydoc-style HTML documentation. This allows a server to provide its own web-based documentation.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]xmlrpc.server.rst, line 423); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]xmlrpc.server.rst, line 423); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]xmlrpc.server.rst, line 430)

Unknown directive type "method".

.. method:: DocCGIXMLRPCRequestHandler.set_server_title(server_title)

   Set the title used in the generated HTML documentation. This title will be used
   inside the HTML "title" element.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]xmlrpc.server.rst, line 436)

Unknown directive type "method".

.. method:: DocCGIXMLRPCRequestHandler.set_server_name(server_name)

   Set the name used in the generated HTML documentation. This name will appear at
   the top of the generated documentation inside a "h1" element.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]xmlrpc.server.rst, line 442)

Unknown directive type "method".

```
.. method:: DocCGIXMLRPCRequestHandler.set_server_documentation(server_documentation)

   Set the description used in the generated HTML documentation. This description
   will appear as a paragraph, below the server name, in the documentation.
```