
VISPA

Release 2.0.0

Sep 15, 2016

1	Introduction	1
1.1	What is VISPA?	1
1.2	What VISPA is NOT?	1
1.3	Contents	1
1.3.1	Installation	1
1.3.2	VISPA License (GPL v2)	2
2	Controller	11
3	Workspace	13
4	Client	15
5	User Guide	17
5.1	User Management	17
5.1.1	Groups	17
5.1.2	Projects	17
5.1.3	Workgroups	17
5.1.4	Roles and Permissions	17
5.1.5	Example	18
5.1.6	Global permissions	19
6	Deployment Guide	21
6.1	Optimized Static Resource	21
6.2	User Management	21
6.3	Upgrade the Database	22
6.3.1	Manual Upgrade	22
7	Developer Guide	25
7.1	Extensions	25
7.1.1	Overview	25
7.1.2	Tutorial	25
7.1.3	Server	25
7.1.4	Signals	25
7.1.5	Client	26
7.1.6	Workspace	26
7.1.7	User Management	26
7.2	Create Database Migration Script	27

8	API Reference	29
8.1	JavaScript API	29
8.1.1	VISPA	29
8.1.2	Workspace	29
8.1.3	vue	32
8.1.4	common	37
8.1.5	utils	42
8.1.6	extension	43
8.1.7	ui	45
8.1.8	mixins	51
8.1.9	jsondata	59
8.1.10	filehandler	60
8.1.11	module	61
8.1.12	prefs	61
8.1.13	views	61
8.2	Python API	69
8.2.1	vispa package	69
	Python Module Index	123

Introduction

1.1 What is VISPA?

1.2 What VISPA is NOT?

1.3 Contents

1.3.1 Installation

Prerequisites

Download Stable Versions

Development versions

Prerequisites

Download Stable Versions

Using *pip* or *easy_install*

Using *pip*:

```
$ pip install vispa
```

or with *easy_install*:

```
$ easy_install vispa
```

It is recommended to use *pip* instead of *easy_install*. If you want to download and install VISPA for yourself proceed to the next instructions depending on your platform.

Unix/Mac

You may download the most current version from [PyPI](#)

For other releases, browse our [download index](#).

- Unzip/untar the files
- Enter the directory created by the file extraction.
- Type “python setup.py install” to install the VISPA module

Windows

TODO

Next Steps

TODO

Development versions

TODO

1.3.2 VISPA License (GPL v2)

GNU GENERAL PUBLIC LICENSE Version 2, June 1991

Copyright (C) 1989, 1991 Free Software Foundation, Inc.,
51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA
Everyone is permitted to copy and distribute verbatim copies
of this license document, but changing it is not allowed.

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change free software--to make sure the software is free for all its users. This General Public License applies to most of the Free Software Foundation's software and to any other program whose authors commit to using it. (Some other Free Software Foundation software is covered by the GNU Lesser General Public License instead.) You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs; and that you know you can do these things.

To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the software, or if you modify it.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must give the recipients all the rights that

you have. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

We protect your rights with two steps: (1) copyright the software, and (2) offer you this license which gives you legal permission to copy, distribute and/or modify the software.

Also, for each author's protection and ours, we want to make certain that everyone understands that there is no warranty for this free software. If the software is modified by someone else and passed on, we want its recipients to know that what they have is not the original, so that any problems introduced by others will not reflect on the original authors' reputations.

Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that redistributors of a free program will individually obtain patent licenses, in effect making the program proprietary. To prevent this, we have made it clear that any patent must be licensed for everyone's free use or not licensed at all.

The precise terms and conditions for copying, distribution and modification follow.

GNU GENERAL PUBLIC LICENSE TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0. This License applies to any program or other work which contains a notice placed by the copyright holder saying it may be distributed under the terms of this General Public License. The "Program", below, refers to any such program or work, and a "work based on the Program" means either the Program or any derivative work under copyright law: that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language. (Hereinafter, translation is included without limitation in the term "modification".) Each licensee is addressed as "you".

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running the Program is not restricted, and the output from the Program is covered only if its contents constitute a work based on the Program (independent of having been made by running the Program). Whether that is true depends on what the Program does.

1. You may copy and distribute verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the Program a copy of this License along with the Program.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of Section 1

above, provided that you also meet all of these conditions:

- a) You must cause the modified files to carry prominent notices stating that you changed the files and the date of any change.
- b) You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.
- c) If the modified program normally reads commands interactively when run, you must cause it, when started running for such interactive use in the most ordinary way, to print or display an announcement including an appropriate copyright notice and a notice that there is no warranty (or else, saying that you provide a warranty) and that users may redistribute the program under these conditions, and telling the user how to view a copy of this License. (Exception: if the Program itself is interactive but does not normally print such an announcement, your work based on the Program is not required to print an announcement.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Program.

In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may copy and distribute the Program (or a work based on it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following:

- a) Accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,
- b) Accompany it with a written offer, valid for at least three years, to give any third party, for a charge no more than your cost of physically performing source distribution, a complete machine-readable copy of the corresponding source code, to be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,
- c) Accompany it with the information you received as to the offer to distribute corresponding source code. (This alternative is

allowed only for noncommercial distribution and only if you received the program in object code or executable form with such an offer, in accord with Subsection b above.)

The source code for a work means the preferred form of the work for making modifications to it. For an executable work, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the executable. However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

If distribution of executable or object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place counts as distribution of the source code, even though third parties are not compelled to copy the source along with the object code.

4. You may not copy, modify, sublicense, or distribute the Program except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense or distribute the Program is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

5. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Program or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Program (or any work based on the Program), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Program or works based on it.

6. Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License.

7. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all. For example, if a patent license would not permit royalty-free redistribution of the Program by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Program.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system, which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

8. If the distribution and/or use of the Program is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Program under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

9. The Free Software Foundation may publish revised and/or new versions of the General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of this License, you may choose any version ever published by the Free Software Foundation.

10. If you wish to incorporate parts of the Program into other free programs whose distribution conditions are different, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY

11. BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE

PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

12. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS

How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively convey the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

```
<one line to give the program's name and a brief idea of what it does.>
Copyright (C) <year> <name of author>
```

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA.

Also add information on how to contact you by electronic and paper mail.

If the program is interactive, make it output a short notice like this when it starts in an interactive mode:

```
Gnomovision version 69, Copyright (C) year name of author
Gnomovision comes with ABSOLUTELY NO WARRANTY; for details type `show w'.
This is free software, and you are welcome to redistribute it
under certain conditions; type `show c' for details.
```

The hypothetical commands `show w' and `show c' should show the appropriate parts of the General Public License. Of course, the commands you use may be called something other than `show w' and `show c'; they could even be mouse-clicks or menu items--whatever suits your program.

You should also get your employer (if you work as a programmer) or your school, if any, to sign a "copyright disclaimer" for the program, if necessary. Here is a sample; alter the names:

```
Yoyodyne, Inc., hereby disclaims all copyright interest in the program
`Gnomovision' (which makes passes at compilers) written by James Hacker.
```

```
<signature of Ty Coon>, 1 April 1989
Ty Coon, President of Vice
```

This General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Lesser General Public License instead of this License.

The starting point for each extension is a class extending `AbstractExtension`:

```
class vispa.server.AbstractExtension (server)
    Base class for Extensions

    add_controller (controller)
        Mount a CherryPy controller using the extension name for path.

        Parameters controller – filename relative to extension directory

    add_workspace_directory (directory='workspace')
        Add files to be transferred to the worker.

        Parameters directory – directory relative to extension directory

    clear_workspace_instance (name, key=None, user=None, workspace=None, db=None)

    config ()

    create_topic (topic='', view_id=None)

    dependencies ()
        Return a list of Extension names this Extension depends on.

    get_workspace_instance (name, key=None, user=None, workspace=None, db=None, **kwargs)

    name ()
        Return the name of the Extension. This name is used as part of the URL.

    setup ()
        Setup the extension.
```

A minimal example looks like this:

```
from vispa.server import AbstractExtension

class MyExtension (AbstractExtension):

    def name(self):
        return "myext"

    def dependencies(self):
        return []

    def setup(self):
        pass
```

Directories from which extensions are loaded:

- vispa/extensions
- \$(var_dir)/extensions
- global packages starting with *vispa_*

In our example, the code above could be placed in a `__init__.py` file in a global package/directory named *vispa_myext-1.0*. By default VISPA loads all extensions it finds, but extensions listed in *vispa.ini*, section *extensions*, option *ignore* will be ignored:

```
[extensions]
ignore = myext
```

Controller

AbstractController

```
class vispa.controller.AbstractController (mount_static=True)
```

```
    cache (workspace_id, key)
```

```
    convert (value, flag)
```

```
    get (key, *args, **kwargs)
```

```
    mount_static (path=None, url='static')
```

```
    release ()
```

```
    release_database ()
```

```
    release_session ()
```

```
    set_cache (workspace_id, item, key=None)
```

add_controller

every exposed function is visible

parameters need to be present or 404, or default

cherrypy.tools.ajax()

return values, string, objects

available cherrypy.request variables

Workspace

```
self.get_workspace_instance()
```

Client

TODO

5.1 User Management

VISPA's user management is designed for collaborative work on different scales. It provides content management as well as a server-side permission system.

5.1.1 Groups

First of all, it is useful to cluster users together to treat them as one unit. This is done by groups, which are just collections of users as well as other groups. Users can join and leave groups following three different stages of privacy:

- public: open for everyone
- protected: protected by password
- private: new users must be confirmed

Groups are organized by selected users, called managers, who can add and remove members as well as changing the groups name, privacy, password etc.

5.1.2 Projects

Projects are the points, where users and groups are connected to content. Therefore, a project consists of several users and groups as well as a collection of items. In order to control the access to the content, a project assigns permissions via roles to each member (see below). Similar to groups, projects are organized by selected users, called managers.

5.1.3 Workgroups

Workgroups are a lightweight version of projects. They have also content in the form of items but only users are members, no groups. Furthermore, there is no explicit handling of permissions because every user has full access to the workgroups content. There are also managers, which organize the membership of the users.

5.1.4 Roles and Permissions

Permissions are descriptive rights, which allow certain actions of the user. They are collected into roles, which users and groups can fulfill in context of a project. Permissions are abstract rights like 'read items of project' and not

concrete ones like ‘read item X of project Y’. Thus, permissions are usable in every project but get their full context not before being assigned to a certain user or group inside one.

5.1.5 Example

In order to bring light to the user managements concept, here is an example of how it can be used. Let’s assume there is a lecture ‘theoretical physics 101’, which wants to use VISPA. First of all, we look at the people, who belong to the lecture. There we have Professor Einstein, several graduates for the tutorials and many students (amongst others Peter, Paul and Mary). Thus, to make it simpler, we have two groups: * ‘theoretical physics 101 - tutors’ * ‘theoretical physics 101 - students’ which consist of the corresponding people. The tutors group should be private, because only a few dedicated persons belong to them while the students group could be public or protected by a password, which is given to the students during the first lecture. Furthermore, we need a project: ‘theoretical physics 101’. Of course, the lecture has some content e.g. slides or homework sheets and solutions, which can be added to the project as items. So far, we have:

- ‘theoretical physics 101’
 - members:
 - * user: Professor Einstein
 - * group: ‘theoretical physics 101 - tutors’
 - ...
 - * group: ‘theoretical physics 101 - students’
 - Peter
 - Paul
 - Mary
 - ...
 - content:
 - * lecture slides
 - * homework sheets
 - * homework solutions

Now, we need to control, who can do what. Therefore, we use different roles. These could be:

- role 1
 - read slides
 - read sheets
- role 2
 - read slides
 - read sheets
 - read solutions
- role 3
 - read slides
 - read sheets
 - read solutions

- write solution

Thus, we can assign the roles as follows:

- Professor Einstein:
 - role 3
- ‘theoretical physics 101 - tutors’:
 - role 2
- ‘theoretical physics 101 - students’:
 - role 1

Thus, the access to the different resources is controlled. Additionally, Peter, Paul and Mary want to work on their homework together. Therefore, they can create their own workgroup, which consists only of them and can be utilized e.g. to exchange some files.

5.1.6 Global permissions

In addition to a concept as in the example above, VISPA is capable of managing global permissions. This is done with a global project, by default called ‘VISPA’. A global Permission is e.g. the right to use a certain extension.

Deployment Guide

6.1 Optimized Static Resource

VISPA uses require.js to load JavaScript, CSS and Template files. In normal or development mode each file is loaded dynamically using the AMD loader. Since the VISPA frontend consists of many files, this is quite some overhead. Require.js provides the means to merge and minify all the files using the r.js optimizer. The basic idea is to provide the optimized files instead of the original files, so not path or other flags need to be changed.

Note: The optimizer currently requires that all extensions are placed in the vispa/extensions folder. Symlinks are possible though.

The following steps are required to use optimized builds in VISPA:

1. Install the npm packages:

```
$ npm install
```

1. Add all extensions that shall be optimized in build.js. Add another module, like the filebrowser extension:

```
{
  name: 'extensions/file/static/js/extension',
  exclude: ['vispa/config', 'vispa/vispa']
},
```

1. Run the optimizer. All static files are copied to the build folder.

```
$ node_modules/.bin/r.js -o build.js
```

1. Set the build_path variable in the [web] section in the vispa.ini. Use a relative path to the vispa package folder or an absolute path.

```
[web]
build_path = ../build
```

6.2 User Management

The concept of VISPA's user management is explained in the User Guide. VISPA is able to automatically setup the user management when the server is ramped up. It is highly recommended to use the auto setup in order to use the full

capability of the user management. The auto setup can be configured via the `vispa.ini`. The `vispa.ini.sample` contains a full example of how such a setup should look:

```
[usermanagement]
# do auto setup?
autosetup = True
# the role/permission setup of the extensions can be accepted or not
accept_extensions = True
# name of the global project
global_project = VISPA
# default user and guest group
user_group = default_user
guest_group = default_guest
# three roles, e.g. with ascending rank
role_1 = Student
role_2 = Tutor
role_3 = Manager
# list of permissions that must exist
permissions = ["project.read_items", "project.create_items", "project.edit_items",
↪ "project.delete_items"]
# assignment of permissions to default roles. permissions must exists (use line above)
role_1_permissions = ["project.read_items"]
role_2_permissions = ["project.read_items", "project.create_items"]
role_3_permissions = ["project.read_items", "project.create_items", "project.edit_
↪ items", "project.delete_items"]
# assignment of roles to user and guest group
user_group_roles = [1]
guest_group_roles = []
```

The comments explain the different options. For a successful setup it is important, that all options are implemented. Otherwise, the automatic setup fails and nothing is committed to the database. It is remarkable, that the permissions “project.*_items” are hard coded into the user management controller and needed for the access to the items of a project.

6.3 Upgrade the Database

VISPA uses alembic to perform database migrations. Per default VISPA upgrades automatically to the latest version of the database schema (head). In a production setup this may not be desirable. To disable automatic database migration set `use_alembic` option in the `alembic` section in the `vispa.ini` file to `False`:

```
[alembic]
use_alembic = False
```

NOTE: In case of an SQLite database, alembic is never used by VISPA, because SQLite does not support all necessary database migration steps.

6.3.1 Manual Upgrade

To manually upgrade the database schema, you also need to set the `sqlalchemy_url` option in the `database` section:

```
[database]
sqlalchemy.url = sqlite:///var/db/vispa.db

[alembic]
```

```
use_alembic = False
script_location = vispa/models/alembic
```

Now you can use the alembic tool to migrate the database:

```
$ alembic -c conf/vispa.ini upgrade head
```

NOTE: The manual upgrade can also be performed on SQLite database, but it can cause errors due to the mentioned limitations of the SQLite dialect. In such a case, the database must be deleted and recreated without alembic, as done by VISPA.

Developer Guide

7.1 Extensions

Extensions add functionality to the the VISPA Platform. They provide new views in the browser and new functionality on the server.

7.1.1 Overview

Extensions contain code to be executed on the client browser, HTML, CSS and Javascript, on the VISPA Server, Python, and on the workspaces, also Python.

7.1.2 Tutorial

TODO

7.1.3 Server

The main entry point is the `AbstractExtension` class in an extension module. Extension modules are either activated in the config file, or by its name: `vispa_<name>`.

7.1.4 Signals

In VISPA important events are communicated by messages. Events concerning the webserver are communicated using the CherryPy bus. VISPA internal events are send through the VISPA bus. For example, each time a user is logged in, an event with the topic “user.login” is sent. The only parameter is the `model.User` object:

```
vispa.publish("user.login", user_object)
```

To receive the message, you need to subscribe to the topic:

```
vispa.subscribe("user.login", login_callback)
```

Where `login_callback` is a function which is called with the parameters provided to the publish call. In the case of “user.login”, the user object:

```
def login_callback(user): print "a new user logged in:", user.name
```

Currently the following topics are published by VISPA:

`exit`, `stop`, `bus.session_added`, `bus.session_removed`, `bus.all_user_sessions_removed`

7.1.5 Client

Folderstructure:

- myextension/
 - __init__.py
 - controller.py
 - extension.py
 - myextension.ini
 - workspace/
 - * ...
 - static/
 - * js/
 - myextension.js
 - myfile.js
 - ...
 - * css/
 - myextension.css/less
 - * templates/
 - ...

RequireJs usage: The namespace is “vispa/myextension/myfile”. Vendor modules are simply identified by their name, e.g. “jquery” or “ace”. There is no need to register the extension yourself.

7.1.6 Workspace

TODO

7.1.7 User Management

The concept of VISPA’s user management is explained in the User Guide. VISPA has an AJAX controller, which gives a simple and convenient access to the user management. The general syntax is:

```
ExtensionView.METHOD("/ajax/um/some_usermanagement_function", ...)
```

Thus, it is a simple AJAX request, e.g.:

```
ExtensionView.GET("/ajax/um/user_get_groups", {}, ...)
```

It is recommended to use the UM controller also on server side python code, because it automatically checks dependencies and permissions. Such a call looks like:

```
Server.controller.ajax.um.some_usermanagement_function(...)
```

For a full documentation of the user management controller look into the class reference.

The user management delivers a cherryypy tool, which can be used to check for global permissions. For further information look into the class reference.

The user management also delivers a simple interface for automatically setting up permissions and roles the extension needs. Therefore, the myextension.ini can be used (look above for folder structure). The myextension.ini is loaded on server ramp up and the extensions can define a set of permission it needs as well as an assignment of the permissions to three default roles. The myextension.ini itself is structured as follows:

```
[usermanagement]
# list of permissions that must exist
permissions = ["myextension.permission1", "myextension.permission2", "myextension.
↪permission3"]
# assignment of permissions to default roles. permissions must exists (use line above)
role_1_permissions = ["myextension.permission1"]
role_2_permissions = ["myextension.permission1", "myextension.permission2"]
role_3_permissions = ["myextension.permission1", "myextension.permission2",
↪"myextension.permission3"]
```

The prefix “myextensions” for the permissions is a namespace which should be used by convention.

7.2 Create Database Migration Script

```
alembic -c <path-to-vispa.ini> revision -m “commit message” --autogenerate
```

API Reference

TODO: Intro to API reference

8.1 JavaScript API

8.1.1 VISPA

TODO: introduction to VISPA

spawnInstance (*extName*, *viewName* [, *args* [, *viewArgs* [, *baseArgs*]]])

Spawn a new view from the given parameters.

Arguments

- **extName** (*String*) – The name of the extension that has the view.
- **viewName** (*String*) – The name of the view to open.
- **args** (*Object*) – The args to be provided to the view.
- **viewArgs** (*Object*) – The viewArgs to be provided to the view.
- **baseArgs** (*Object*) – The baseArgs to be provided to the view.

Returns The view that was opened. It is most likely not ready to use yet.

Return type `BaseView`

Throws **Error** – If either the extension or the view was not found.

8.1.2 Workspace

class **Workspace** (*config*)

Extends `Emitter`

Mixins `LoggerInterface`

The abstract representation of a workspace and its functions.

Warning: This class is not to be instantiated manually. Instead use its class-members functions.

Arguments

- **config** (*Object*) – The configuration of the workspace.

read-onlyid

Type Number

The id of the workspace.

read-onlyname

Type String

The name of the workspace.

read-onlystate

Type String

The connectivity state of the workspace. Possible values are: - disconnected - requestedConnect - connecting - connected - requestedDisonnect - disconnecting

Emits stateChanged If this value changed.

color

Type String

The assigned color of this workspace.

connect ([, *callback* [, *password*]])

Connect the workspace.

Callback args

1. **err** (*String*/*null*) – Will contain the error description or *null* if the connect was successfull. Consider using the *connected* event instead.

Arguments

- **password** (*String*) – May contain the password/SSH passphrase used for connecting.

disconnect ([, *callback*])

Disconnects the workspace.

Callback args

1. **err** (*String*/*null*) – Will contain the error description or *null* if the disconnect was successfull. Consider using the *disconnected* event instead.

edit (*newConfig* [, *callback* [, *connect*]])

Edit a current workspace. Will be disconnected if it is not already.

Arguments

- **newConfig** (*Object*) – The new config for the workspace.
- **newConfig.name** (*String*) – The new workspace name.
- **newConfig.host** (*String*) – The new host name.
- **newConfig.login** (*String*) – The new login user name.
- **newConfig.key** (*String*) – The new private key used for authentication.
- **newConfig.command** (*String*) – The new command.
- **connect** (*Bool*) – Connect when edit has finished.

Callback args

1. **err** (*String*/*null*) – Will contain the error description or *null*.

remove ([, *callback*])

Removes the workspace. A success implies the destruction of the instance and it must not be used thereafter.

callback *String*/*null* **err** Will contain the error description or *null* if the remove was successful

closeAllInstances ()

Forces all known open instances to be closed. This is synchronus.

initPrefs ()

Registers the preferences with *vispa.preferences*.

class-memberload (*callback*)

Loads the workspaces from the DB and registers for the socket event redistribution.

Callback args

1. **err** (*String*/*null*) – *null* on success else the error description.

class-membergetId (*workspace*, *fallback*)

Retrieves the workspace id if possible or return the fallback value.

Arguments

- **workspace** (*Workspace*/*Number*) – A Workspace instance or a valid workspace id.
- **fallback** (*any*) – The fallback value.

Returns The workspace if available or the fallback value.

Return type *Number*

class-memberbyId (*id*)

Retrieves the workspace by id.

Returns the workspace or *null* if there no workspace for the id given

Return type *Workspace*

class-memberbyName (*name*)

Retrieves the first workspace with the given name.

Returns the workspace or *null* if there no such workspace

Return type *Workspace*

current

Type *Workspace*/*null*

The current workspace. This is the workspace new vies might be bound to - it is not the workspace of the currently active view/thread.

read-only class-memberworkspaces

Type *Object*(*id* => *Workspace*)

The workspace directory (*id* => *workspace*). Safe against alteration.

workspaceCount

Type *Number*

The number of workspaces.

class-member**createNew** (*config* [, *callback*])

Create a new workspace and saves it.

Arguments

- **config** (*Object*) – A plain object holding the informations for the new Workspace. See *Workspace()* constructor argument.

Callback args

1. **err** (*String/null*) – The error information or *null* if it was a success.
2. **workspace** (*Workspace*) – The newly created workspace.

8.1.3 vue

vue/sidebar

class Toggle ()

Extends MenuItem

Arguments

- **object** (*config*) – The config of the Toggle.

A toggle. This class should not be instantiated directly. It extends *MenuItem()* to be a toggle with no submenu and eventually an img instead of an icon.

img

Type Bool

Determines, if the toggle is an image instead of an icon.

imgClass

Type String

The class the toggle gets in case of an img.

src

Type String

The html src in case of an img.

contentItem

Type Bool

Only content items will have an entry in the overflow menu as well as in the body.

id

Type String

The corresponding element in the body will get this id.

class Sidebar ()

Extends Vue

Arguments

- **object** (*config*) – The config of the MenuButton. See attributes for more details.

A side bar with a header of toggles and an overflow button menu as well as a body with where the toggles place their content.

itemClass

Type String

The class each toggle gets.

contentClass

Type String

The class a content in the body gets.

items

Type Object (toggle definitions)

The items are used in two (three) ways.

active

Type Bool

Determines, whether the sidebar gets the CSS class 'active' und thus, is expanded

size

Type Integer

The width of the sidebar in percent when active.

footer

Type Object

Contains the data for the footer: footer.items, footer.menu.iconClass, footer.menu.label, footer.menu.btnClass, footer.menu.items

vue/base

runSync (*func* [, *scope*])

Runs the function in synchronous mode.

Warning: Avoid using this at all cost!

Note: This will not flush the Vue internal queue, thus some action may actually not be synchronous because they are already queued (and we have not means to flush it manually).

Arguments

- **func** (*Function*) – The function to be run.
- **scopy** (*Object*) – The scope (*this*) to run the function on.

syncVueFrgs (*vueInstances*)

Moves a list of (single node) vue instance behind the fragment start node - if any.

Arguments

- **vueInstances** (*Array/Object*) – An collection of vue instances (usually obtained from \$refs).

compProxy (*base, keys*)

Generates a mixin of computed properties that delegate their access to a object contained withing the instance.

Note: Setting value that were previously unset is done in a manner that support their detection within the Vue framework.

Arguments

- **base** (*String*) – The name of the object that the access should be delegated to.
- **keys** (*Array[String]*) – The name of the properties to be delegated to the objetct.
- **keysFunc** (*Array[String]*) – The name of the properties to be delegated to the objetct, if these are functions they will be called for generation of dynamic properties.

Returns The object in Vue mixin style.

Return type Object (Vue mixin)

update (*target, source[, clean]*)

Recursively updates a *target* with data from *source*, with Vue data interactivity.

Arguments

- **Object** (*source*) – The object to be updated.
- **Object** – The object with the update template.
- **Bool** (*clean*) – Clean all key from the *target* that are not in source. Essentially a Vue aware deep copy.

mixin itemsAcc ()

Provides access to the instances that correspond with the configs with the same key.

get (*key*)

Retrieves a *MenuItem()* for the corresponding entry in *items*.

Arguments

- **key** (*String*) – The key of entry.

Returns The entry for the key.

Return type MenuItem

set (*key, config*)

Sets the config for for a given key. This is the only valid way to create a new *MenuEntry()*.

Arguments

- **key** (*String*) – The key of the entry.

del (*key*)

Removes an item for *items*. This is the only valid way to delete a *MenuEntry()*.

Arguments

- **key** (*String*) – The key of the entry.

read-only length**Type** NumberThe number of entries in *items*.**vue/menu****class MenuItem()****Extends** Vue**Mixins** VueBase.itemsAcc**Arguments**

- **object** (*config*) – The config using the same options as can be provided to sub-MenuItem's via `:js:attr:items()`.

An menu item. This class should not be instanciaded directly.

header**Type** BooleanWhether this item is a header. Default is *false*. If it is *true* *label* and *iconClass* will be used for its contnet.**divider****Type** BooleanWhether this item is a divider. Default is *false*.**disabled****Type** BooleanWhether this entry should be disabled. Default is *false*.**label****Type** String

The label of the entry. Must be provided.

iconClass**Type** String (icon class-name)

The class-name for the icon to use (if any). Default is "".

auxClass**Type** String (class-names)

Auxiliary class names for the menu item for additional styling.

position**Type** Number

Defines the ordering of the entries. They will be sorted in ascending order.

hidden

Type Boolean

Whether this item is hidden (not shown at all).

callback**Type** Function

This function will be called with *this* referring to the `MenuItem()`. If it returns *true* the menu will be kept open.

items**Type** Object (key => config)

Contains configuration of subitems by key. All the above mentioned attributes can be included in a config.

data**Type** Object

The original data object, this way auxiliary data provided can be accessed.

root**Type** MenuButton

The root instance containing the menu.

class `MenuButton()`

Extends Vue

Mixins VueBase.itemsAcc

Arguments

- **object** (*config*) – The config of the MenuButton. See attributes for more details.

A MenuButton with a Menu attached.

label**Type** String

The label of the button. Must be provided.

dropup**Type** Boolean

Whether this is a dropup menu. Default false.

iconClass**Type** String (icon class-name)

The class-name of the icon for the button (displayed before the label).

btnClass**Type** String (class-names)

The class-names to be included. Should be provided.

class**Type** String (class-names)

The class-names for the MenuButton wrapper. May be used to integrate it into a input group.

items

Type Object (menu definition)

See *MenuItem()*.

class MenuHeadless ()

Extends Vue

Mixins VueBase.itemsAcc

Arguments

- **object** (*config*) – The config of the MenuHeadless. See attributes for more details.

Just a headless menu, which is to be dynamically \$attacheTo a menu head, to provide the same menu in multiple locations.

items

Type Object (menu definition)

See *MenuItem()*.

objClone (obj)

Generates a proper (pseudo-)clone of an object describing a menu.

Arguments

- **obj** (*Object*) – The object describing the menu.

Returns The cloned menu describing object

Return type Object

spawnFromFile (viewClass, selectorArgs, callback)

Arguments

- **viewClass** (*BaseView*) – The view class to be spawned.
- **selectorArgs** (*Object*) – Additional arguments to be passed to the file selector.

Callback args

1. **path** (*String*) – The path argument of the fileSelector callback. If the callback does not return *undefined*, this value will be used as *args* while spawning the *view-Class*.

8.1.4 common

common/color

class Color (*val[, predefinedColors]*)

A class providing color conversion and parsing.

Note: This class is provided and documented by *Bootstrap Colorpicker* <<https://mjolnic.com/bootstrap-colorpicker/>>.

Arguments

- **val** (*String|Object*) – The color value, either as valid CSS color or object describing in the HSBA color scheme.
- **val.h** (*Number*) – HSBA hue.
- **val.s** (*Number*) – HSBA saturation.
- **val.b** (*Number*) – HSBA brightness.
- **val.a** (*Number*) – HSBA alpha.
- **predefinedColors** (*Object*) – Additional named color aliases.

common/dialog

class Dialog (*vueOpts*)

Extends Vue

Creates an simple extensible (modal) dialog.

Arguments

- **vueOpts** (*Object*) – The options given to the vue instance.
- **vueOpts.appendTo** (*HTMLElement*) – The element to append the dialog to. (optional)
- **vueOpts.data** (*Object*) – The configuration of the dialog. The keys in this object are the same as the properties on the instance.

rootClass

Type String

Default ""

An optional, additional CSS class on the very root node of the dialog.

containerClass

Type String

Default ""

An optional, additional CSS class on the container node of the dialog.

iconClass

Type String (CSS icon class)

Default ""

If given, this icon will be displayed in front of the title.

title

Type String

Default ""

The title of the dialog.

body

Type String (HTML)

Default ""

The body of the dialog. Is interpreted as HTML.

closable**Type** Boolean**Default** true

Whether the dialog can be closed by clicking a cross in its opper right corner.

backdrop**Type** Boolean**Default** true

Whether the background should be grayed out or not.

isShown**type** Boolean**default** false

Whether the dialog is currently shown. If set from the beginning, the inital animation is skipped.

onOpen**Type** Function**Default** null

If provided, this function will be called when opening the dialog.

onClose**Type** Function**Default** null

If provided, this function will be called when closing the dialog.

buttons**Type** Object**Default** { }

An object containing further object describing the button to be present in the footer.

button.*.label**Type** String

The label of the button.

button.*.iconClass**Type** String (CSS icon class)

The icon of the button.

button.*.class**Type** String (CSS class)

The CSS class of the button.

button.*.pos**Type** Number|String

The buttons are sorted by this value. (ascending).

`button.*.callback`

Type Function

The function to be called when this button is clicked. (*this* is the dialog instance). If it returns true, the dialog will stay open. If not provided it will close too.

focusButton

Type String

The button with this key will be initially focused.

class PromptDialog (*vueOpts*)

Extends Dialog

callback

Type Function

The callback that will be called with the result of the prompt.

value

Type String

Default ""

The current value of the prompt input. If provided from the beginning, this is the default value.

password

Type Boolean

Default False

Whether the prompted value should be masked (for passwords prompting).

placeholder

Type String

Default ""

The placeholder value of prompt input - only visible with no current input and no focus.

`Dialog.dialog` (*opts* [, *appendTo*])

Opens a dialog.

Arguments

- **opts** (*Object*) – The same object as *vueOpts.data* of *Dialog()*.
- **appendTo** (*HTMLNode*) – The same parameter as *vueOpts.appendTo* of *Dialog()*. If not provided, it will attempted to fill it with *this.\$messageContainer* (see *BaseView()*).

`Dialog.alert` (*message* [, *opts* [, *appendTo*]])

Shows an alert message.

Arguments

- **message** (*String*) – The message to be displayed
- **opts** (*Object*) – The same object as *vueOpts.data* of *Dialog()*. Will be merged over the opts generated by the other parameters.
- **opts.okLabel** (*String*) – An easy way to overwrite the default label “Ok” (other route is *opts.buttons.ok.label*).

- **appendTo** (*HTMLNode*) – The same parameter as *vueOpts.appendTo* of *Dialog()*. If not provided, it will attempted to fill it with *this.\$messageContainer* (see *BaseView()*).

`Dialog.confirm(message, callback[, opts[, appendTo]])`

Shows an alert message.

Arguments

- **message** (*String*) – The message to be displayed
- **opts** (*Object*) – The same object as *vueOpts.data* of *Dialog()*. Will be merged over the opts generated by the other parameters.
- **opts.yesLabel** (*String*) – An easy way to overwrite the default label “Yes” (other route is *opts.buttons.yes.label*).
- **opts.noLabel** (*String*) – An easy way to overwrite the default label “No” (other route is *opts.buttons.no.label*).
- **appendTo** (*HTMLNode*) – The same parameter as *vueOpts.appendTo* of *Dialog()*. If not provided, it will attempted to fill it with *this.\$messageContainer* (see *BaseView()*).

Callback args

1. **yesChosen** (*Boolean*) – The yes(-kind) optiong was chosedn.

`Dialog.prompt(message, callback[, opts[, appendTo]])`

Shows an alert message.

Arguments

- **message** (*String*) – The message to be displayed
- **opts** (*Object*) – The same object as *vueOpts.data* of *Dialog()*. Will be merged over the opts generated by the other parameters.
- **opts.cancelLabel** (*String*) – An easy way to overwrite the default label “Cancel” (other route is *opts.buttons.cancel.label*).
- **opts.okLabel** (*String*) – An easy way to overwrite the default label “Ok” (other route is *opts.buttons.ok.label*).
- **appendTo** (*HTMLNode*) – The same parameter as *vueOpts.appendTo* of *Dialog()*. If not provided, it will attempted to fill it with *this.\$messageContainer* (see *BaseView()*).

Callback args

1. **result** (*String/null*) – The result of the prompt.

common/socket

`class Socket(url[, options])`

text

Arguments

- **url** (*string*) – Url.
- **options** (*string*) – Options.

8.1.5 utils

format (*template*, ...)

Substitutes each occurrence of the curly brackets enclosed number *n* with the with the *n+1* additional argument.

Arguments

- **template** (*String*) – The template to use.
- ... (*String*) – The substitute.

Returns The processed string.

Return type String

Example format("foo {0}!", "bar") == "foo bar!"

uuid ([, *length*])

Generates a UUID consisting of upper and lower case characters and numbers at random.

Arguments

- **length** (*Number*) – The length of the requested UUID. Defaults to 10.

Returns The UUID.

Return type String

capitalize (*string*)

Capitalizes the first character of the given string with toUpperCase, hence is not respecting the locale specific case definitions.

Arguments

- **string** (*String*) – The input string.

Returns The capitalized string.

Return type String

callback (*callback*)

Used for callback type checking and defaulting.

Arguments

- **callback** (*Function*) – The callback to check. If it is not a function it will return

an empty function.

Returns A proper callback function, the provided one if it is valid.

Return type Function

ROP (*value*)

Return a property descriptor for a read-only property, which is still configurable and enumerable. It also contains the descriptor flag required for JClass property definitions.

Arguments

- **value** (*Any*) – The value of the property.

Returns A property descriptor that is JClass ready.

Return type Object

setROP (*object*, *name*, *value*)

Sets a read only (but still configurable and enumerable) property on the given object.

Arguments

- **object** (*Object*) – The object to set the property on.
- **name** (*String*) – The name of the property.
- **value** (*Any*) – The value to set the property to.

Returns The *value*.

Return type Any

asyncRequire (*path*)

Interfaces the a require calls into the async callback system.

Arguments

- **path** (*String*) – The path to require.

Returns A function that preforms the require call and accepts an callback(err, returnValue)

Return type Function

formatDate (*data*[, *options*])

Format s date object appropriately.

Arguments

- **date** (*Date*) – The date to format.
- **options** (*Object*) – Further options for the formatting. (none supported yet)

Returns The formatted date.

Return type String

shortcutPP (*definition*[, *os*])

Shortcut preprocessing

Arguments

- **definition** (*String*) – The shortcuts definition.
- **os** (*String*) – The OS for which the shortcuts should be preprocessed. If not given, the value from *vispa.device.os* will be used.

Returns The shortcut ready to be handles by the jQuery.Shortcuts plugin.

Return type String

8.1.6 extension

class Extension (*name*)

Extends Emitter

Mixins AjaxMixin

Mixins LoggerMixin

The basis of an extension.

read-onlyname

Type String

The name of the workspace.

addView (*viewClass*)

Adds a view class to the extension. A view can be party of only one extension.

Arguments

- **viewClass** (*JClass (BaseView)*) – The view class to add.

Throws

- **Error** – If *viewClass* is not a JClass extending BaseView.
- **Error** – If the *viewClass* is already assigned to another extension.
- **Error** – If a *viewClass* with the same name is already registered.

Returns The main menu entry of the class (*viewClass._members.mainMenuEntry*).

Return type Object|null

(*menuItems*)

Adds the given menu items to the main menu.

Arguments

- **menuItems** (*Array*) – The items to add.

addViews (*viewClasses*)

Adds multiple views to the extension. Calls *addView()* for each element of the provided Array.

Arguments

- **viewClasses** (*Array (JClass (BaseView))*) – An Array of view classes.

Throws Error – If *viewClasses* is not an Array.

getView (*viewName*)

Retrieves the view class for the view with this name.

Arguments

- **viewName** (*String*) – The name of the view to retrieve.

Returns The view class or *null* if there is none for the given name.

Return type BaseView

read-onlyviews

Type Object(name => JClass(BaseView))

The object that holds all the views by their name. Safe against manipulation.

closeAllInstances ()

Closes all know instances of this extension. Is synchronus.

class-memberload (*callback*)

Loades all extensions.

Callback args

1. **err** (*String|null*) – *null* on success else the error description.

class-memberbyName (*name*)

Retrieves the view class by its name.

Arguments

- **name** (*String*) – The new of the view class.

Returns The view class.

Return type JClass(BaseView)

read-only class-memberextensions

Type Object(name => Extension)

The object that holds all the extensions by their name. Safe against manipulation.

8.1.7 ui

ui/tabber

class Tabber ()

Extends UINodeCollection

This Class arranges its nodes in an tab bar to display only one at a time.

activeInstance

Type UINode|null

Default null

Holds the currently active instance of the *Tabber* ()

attach (*instance* [, *leftOf* [, *rightOf* [, *activeIndex*]]])

Attaches an instance.

Note: Instances are expected to have a \$tab attribute consisting of a JQuery object containing the DOM node representing its tab.

Arguments

- **instance** (*UINode*) – The instance to attach.
- **leftOf** (*UINode* | *Number*) – See UINodeCollection.attach().
- **rightOf** (*UINode* | *Number*) – See UINodeCollection.attach().
- **activeIndex** (*Number*) – The active index for the instance to attach. The higher the index the more recent this tab was shown. The currently shown tab has the highest index. If it is not a valid Number it will be set to the highest index. Negative numbers will be subtracted from the currently highest active index, but are clipped at 0.

replace (*oldInstance*, *newInstance*)

Replaces an instance with a new one.

Arguments

- **oldInstance** (*UINode*) – Instance to replace.
- **newInstance** (*UINode*) – Instance to take the place.

detach (*instance*)

Detaches a currently attached instance.

Arguments

- **instance** (*UINode*) – Instance to detach.

getNeighbor (*instance*, *dir*)

Get the neighboring instance left/right of the given one.

Arguments

- **instance** (*UINode*) – The instance whose neighbor is to be found.
- **dir** (*String*) – The direction to look for the neighbor. Valid values are “+” for left or “-” for right.

Returns The neighboring instance or undefined if there is no neighbor.

Return type *UINode*

class-member**validInstanceClass**

Value *MainView*

See *UINodeCollection()* for details.

ui/uinodecollection

class *UINodeCollection* ()

Extends *UINode*

Designed to hold *UINode()* s.

Warning: This is only a skeleton class. It is not to be instantiated!

attach (*instance*)

Attach instance to internal instance array.

Warning: This method does only implement some checks and does detach the instance if it is already attached.

Arguments

- **instance** (*UINode*) – The instance to attach.
- **leftOf** (*UINode/Number*) – The instance to attach will be left of this instance or the instance at this position.
- **rightOf** (*UINode/Number*) – If *leftOf* is *undefined* then this option is considered. The instance to attach will be right of this instance or the instance at this position.

Returns The position the instance will be placed at.

Return type *Number*

Note:

Both, *leftOf* and *rightOf* are interpreted this way:

- If it isn't a *Number* then interpret the argument as an instance. If it isn't an valid instance either, just default to the position at the very end.
- If it is a *Number* then interpret it as position in the list of instance, with 0 being the first. If it is negative then use it as an offset from the end. e.g. -2 is the second last position (clipped at the first position)

replace (*oldInstance*, *newInstance*)

Replaces a currently attached instance with another one.

Warning: This method does only implement some checks und does detach the newInstance if it is already attached

Arguments

- **oldInstance** (*UINode*) – Instance to replace.
- **newInstance** (*UINode*) – Instance to take the place.

Returns Index of the instance to replace in the internal instance array.

Return type Number

detach (*instance*)

Detaches a currently attached instance.

Warning: This method does not implement anything, it is to be extended.

Arguments

- **instance** (*UINode*) – Instance to detach.

detachAll ()

Detaches all instance (one by one).

Returns Count of instances detached.

Return type Number

length

Type Integer

The number of current instances this collection has.

class-member validInstanceClass

Value *UINode*

Only instances of this Class will be accepted by *attach()* and *replace()*. Classes extending *UINodeCollection()* are expected to define it appropriately.

ui/split**class Split** (*instances* [, *sizes*])**Extends** `UINodeCollection`

This class provides the ability to arrange the attached nodes next to each other in the set direction. The size of each can be changed with the draggable bars divideing the nodes.

Arguments

- **instances** (*Array* (`UINode`)) – An array of instances to attach.
- **sizes** (*Array* (`Number`)) – An array of the sizes for the instances to attach.

direction**Type** `String`**Default** `“horizontal”`

This lower-case string sets the direction along which the nodes are arranged. Valid values are “horizontal” or “vertical”. For setting this value it is enough to provide the first letter: “h” or “v”.

Throws Error If attempting to set an invalid direction or set the direction while more than 1 `UINode()` is attached.

attach (*instance* [, *leftOf* [, *rightOf*]])

This function attaches an `UINode` instance.

Arguments

- **instance** (`UINode`) – The instance to attach.
- **leftOf** (`UINode` / `Number`) – See `UINodeCollection.attach()`.
- **rightOf** (`UINode` / `Number`) – See `UINodeCollection.attach()`.

replace (*oldInstance*, *newInstance*)

Replaces an instance with a new one.

Arguments

- **oldInstance** (`UINode`) – Instance to replace.
- **newInstance** (`UINode`) – Instance to take the place.

detach (*instance*)

Detaches a currently attached instance.

Arguments

- **instance** (`UINode`) – Instance to detach.

resizeEnabled**Type** `Boolean`**Default** `true`

Whether resizing is enabled.

ui/uinode**class UINode ()****Extends** Emitter

A class with an assigned DOM node representing it.

master**Type** UINode**Default** null**Throws Error** If attempting to set the master to an invalid type/class.

The designated master for this node.

parent**Type** UINode**Default** null**Throws Error** If attempting to set the parent to an invalid type/class.

The designated parent for this node.

\$node**Type** JQuery**Default** null**Throws Error** If attempting to set the \$node to an invalid type/class.

The JQuery object containing the DOM element (exactly 1) representing this node.

ui/viewmangager**class ViewManager** (*container, listenerTarget*)**Extends** UINodeCollectionThis window manager allows for multiple abitrary arranged *MainView()* by using *Split()* and *Tabber()*.**Arguments**

- **container** (*JQuery*) – The JQuery container DOM node in which all content will be rendered.
- **listenerTarget** (*Emitter*) – The Emitter whichi is watched for focus/blur and close events to take action upon.

read-onlyscope**Type** String

An UUID. This is internally needed to prevent the attached UINodes from escapeing when they are dragged around.

read-onlyinstances**Type** Array(MainView)

An array of all attached instances.

read-onlyvisiableInstances

Type `Array(MainView)`

An array of all currently visible instances (shown).

read-only `focusInstance`

Type `MainView`

Default `undefined`

The currently focused instance or undefined if none is focused.

isManaged (*instance*)

Test whether the instance is attached.

Arguments

- **instance** (`MainView`) – The instance to test.

Returns Is the instance attached?

Return type `Bool`

isVisible (*instance*)

Test whether the instance is visible.

Arguments

- **instance** (`MainView`) – The instance to test.

Returns Is the instance visible?

Return type `Bool`

isFocused (*instance*)

Test whether the instance is focused.

Arguments

- **instance** (`MainView`) – The instance to test.

Returns Has the instance focus?

Return type `Bool`

attach (*instance* [, *viewArgs*])

Attaches an instance.

Arguments

- **instance** (`MainView`) – The instance to attach.
- **viewArgs** (*Object*) – An object holding further options for the placing of the instance.
- **viewArgs.replace** (`MainView`) – Replaces this currently attached instance. If it is not valid instance to replace, this option will be ignored entirely.
- **viewArgs.tabber** (`Tabber`) – Attach the instance inside this tabber. Will attempt to respect the following 2 parameters (for numeric placement). If it is not valid tabber, this option will be ignored entirely.
- **viewArgs.leftOf** (*MainView/Number*) – See `attach()` of `UINodeCollection()`.
- **viewArgs.rightOf** (*MainView/Number*) – See `attach()` of `UINodeCollection()`.

- **viewArgs.focusIndex** (*Number*) – The focus index for the instance to attach. The higher the index the more recent this instance had focus. The currently focused instance has the highest index. If this value is not present or *0* it will default to the currently highest index. Negative values will be used as and offset from the currently highest value.
- **viewArgs.activeIndex** (*Number*) – The active index. See *attach()* of *Tabber()*.

Note:

Both *viewArgs.leftOf* and *viewArgs.rightOf* are interpreted this way:

- If it is not a valid Number it will be set to the highest index.
- Negative numbers will be subtracted from the currently highest active index, but are clipped at *0*.

detach (*instance*)

Detaches a currently attached instance.

Arguments

- **instance** (*MainView*) – Instance to detach.

focusNeighbor (*dir*)

Focuses the neighboring instance left/right of the currently focused one.

Arguments

- **dir** (*String*) – The direction of the neighbor. Valid values are “+” for left or “-” for right.

destroy ()

Destroys itself: Detaches all instances, stops listening to events and renders itself unusable.

byId (*id*)

Retrieves an attached view by id.

Arguments

- **id** (*String*) – The *id* of the view to retrieve.

Returns The view, if available, else *null*.

Return type *MainView*/*null*

8.1.8 mixins

mixins/state

mixin StateMixin ()

Mixin that defines the state getter and loader.

read-onlystate

Type *StateProxy*

The object managing the state,

class StateProxy (*base*)

Extends *Emitter*

Provides the state handling functions.

Warning: There is no need to instantiate this object directly, this is all handled by the state mixin.

Arguments

- **base** (*Emitter*) – The instance of which the state is managed. Used for informing it about state changes.

read-onlyobj

Type Object

The object holding the state information. Manipulating its values will not cause the appropriate events to be emitted.

read-onlybase

Type Emitter

The instance of which the state is managed.

get (*stateKey* [, *defaultValue*])

Retrieves the value of a state. Returns undefined if it is not set.

Arguments

- **stateKey** (*String*) – The key of the value to be retrieved.
- **defaultValue** (*Any*) – The value to return if the given state key is not defined.

Returns The value for the given *stateKey*.

Return type Any

set (*stateKey*, *value* [, *delay*])

Sets the value of a state key, then triggers the events in the order they are documented in. Finally calls `save()` and returns the final value.

Arguments

- **stateKey** (*String*) – The key of the value to set.
- **value** (*Any*) – The value to set. Must be serializable by JSON to be saved.
- **delay** (*Number*) – `save()` is called with this parameter.

Returns The new value for the *stateKey*. (This may actually differ from what was intended to be set, if it was changed during the events.)

Return type Any

Emits `*stateKey*(newValue, oldValue)*stateKey*(newValue, oldValue)` Events of the appropriate *stateKey* will be emitted when its value changed.

Emits `changedState(stateKey, newValue, oldValue)` Is emitted for state changes - but on the *base* object.

setup (*source* [, *auxSource*])

Will update the internal state with the values from the given parameters.

Arguments

- **source** (*Object*) – An object containing the values to apply to the state.
- **auxSource** (*Object*) – An additional object containing the values to apply to the state.

Emits `changedState(null,null)` Is always emitted afterwards - but on the *base* object.

save ([, *delay*])

Save the current state.

Arguments

- **delay** (*Number*) – The maximum delay in seconds to save the current state to the server. For more details see [save\(\)](#) of `SessionManager()`.

mixins/load

mixin `LoadMixin()`

Mixin that provides load state functionality.

loading

Type Boolean

Is *true* if any loading (not necessarily the initial loading) is still in process. Setting its value is the same as calling `setLoading` with this value.

setLoading (*loading* [, *delay* [, *setter*]])

In-/Decreases the internal loading counter. May be overloaded to visually represent the loading process.

Arguments

- **loading** (*Boolean*) – A logical *true* value will increase the loading count, anything else will decrease it.
- **delay** (*Number|null*) – An optional delay to use for calling the setter.
- **setter** (*Function*) – A function to update the visual loading state. This parameter only to be used by the views extending the `BaseView()`.

Emits `loadingDone` When the internal loading counter has reached 0 again.

read-only `loaded`

Type bool

Is *true* if the initial loading has finished.

Note: Is to be overridden.

load (*callback*)

This function is expected to start the initial loading process.

Note: Is to be overridden.

Callback args

1. **err** (*Error|null*) – A possible error.

Emits loaded Once loading has finished after the callback was called.

mixins/preferences

PreferencesMixin

mixin PreferencesMixin (*category*, *group* [, *prefs*])

Mixin that provides preference functionality.

Arguments

- **category** (*String*) – Preferences category.
- **group** (*String*) – Preferences group.
- **prefs** (*Object*) – The config for the preferences to create. See `:js:class:PrefGroup`'s *data* parameter for details.

prefs

Type PreferencesProxy

All preferences related function are collected here.

applyPreferences (*key*, *value*)

This function will be bound to the *changed* event of the preferences object. This is also triggered for non-auto applied shortcut.

Arguments

- **key** (*String*) – The key of the setting changed. May be undefined if multiple settings changed at once - e.g. after loading the from the DB.
- **value** – The new value of the setting changed. Only available if the *key* was set.

prefs.vue

Type PrefsGroup (Vue)

The preferences group.

prefs.get (*key*)

Retrieves the value of a preference item given by key.

Arguments

- **key** (*String*) – The key of the preferences item.

Returns The current value of the preferences item.

prefs.set (*key* [, *value* [, *global*]])

Retrieves the value of a preference item given by key.

Arguments

- **key** (*String*) – The key of the preferences item.
- **value** – The new value for the preferences item.
- **global** (*Bool*) – The value will be set globally, ignoring eventual binding.

prefs.watch (*key*, *callback*)

Watches a specific preferences items value for changes.

Note: This uses the *\$watch* method of *vuejs* internally.

Arguments

- **key** (*String*) – The key of the preferences item to watch. If *null* will watch all preferences of the entire group.

Callback args

1. **oldValue** – The old value of the preference before it was changed.
2. **newValue** – The new value of the preference after it was changed.

read-only`prefs.autoShortcuts`

Type

Object(shortcutDefinition => callback)

All shortcuts do be automatically applied (by *ShortcutsMixin()*).

mixins/ajax

mixin AjaxMixin (*dynamicBase*, *staticBase*[, *defaultQuery*])

Mixin that provides ajax functionality, such as GET and POST methods as well as cached file retrieval.

Arguments

- **dynamicBase** (*String*) – See *dynamicBase*.
- **staticBase** (*String*) – See *staticBase*.
- **defaultQuery** (*String*|*Function*) – Defaults to *""*. See *defaultQuery*.

read-only`dynamicBase`

Type *String*

The dynamic base for ajax calls.

read-only`staticBase`

Type *String*

The static base for file retrievals via *getFile()*.

read-only`defaultQuery`

Type *String*|*Function*

The default query string when sending ajax requests.

Note: When this value is assigned to a function, the getter still returns a string by evaluating the function without arguments.

staticURL (*url*)

Returns the fully prefixed URL for a static resource. If the given url starts with a slash it will return a global URL (same as calling this function on the vispa object).

Arguments

- **url** (*String*) – The URL of the resource to get.

Returns The fully prefixed URL.

Return type String

dynamicURL (*url*)

Returns the fully prefixed URL for a dynamic resource (usually ajax). If the given url starts with a slash it will return a global URL (same as calling this function on the vispa object).

Arguments

- **url** (*String*) – The URL of the resource to get.

Returns The fully prefixed URL.

Return type String

getFile (*url* [, *callback*])

Retrieves the content of a file. This function caches file contents identified by *url*.

Arguments

- **url** (*String*) – The URL to a file to retrieve, relative to *staticBase*.

Callback args

1. **err** (*Error* | *null*) – A possible error.
2. **content** (*String*) – The content of the retrieved file.

ajax (*method*, *url* [, *data*] [, *callback*])

Ajax wrapper method that represents the core of any ajax call. It should be used instead of jQuery's methods since vispa implements the callback-as-last-argument style instead of using jqXHR/Deferred objects.

Arguments

- **method** (*String*) – The http method to use. Currently, only “GET”, “POST” and “DELETE” are supported.
- **url** (*String*) – The url of the endpoint to use, relative to the dynamic base.
- **data** (*Object* | *String*) – The data to send. When a string is passed, it is assumed that it is an encoded json object and the content type will be set accordingly. When an object is passed, it is forwarded to jQuery's ajax method the normal way.

Callback args

1. **err** (*Error* | *null*) – A possible error.
2. **data** – The value of the *data* attribute of the response.
3. **res** (*Object*) – The response object.

GET (...)

Convenience wrapper around *ajax()* for sending GET requests.

POST (...)

Convenience wrapper around *ajax()* for sending POST requests.

DELETE (...)

Convenience wrapper around *ajax()* for sending DELETE requests.

mixins/shortcuts

ShortcutsMixin

mixin ShortcutsMixin (*shortcutsContext*)

Mixin that steers shortcuts.

Arguments

- **shortcutsContext** (*String*) – See `shortcutsContext`.

read-only shortcutsGroup

Type `String`

The `shortcuts group` that shortcuts are registered to.

applyShortcuts ()

Applies the current shortcuts. All shortcuts will be reset at the beginning of this function.

mixins/link

mixin LinkMixin (*workspace*)

Mixin that provides *optional* links to a `Workspace()`.

Arguments

- **workspace** (`Workspace`) – See `workspace`.

read-only workspace

Type `Workspace`

The workspace this object is bound to.

read-only workspaceId

Type `Number|null`

The linked workspace Id or *null*.

mixins/logger

LoggerMixin

mixin LoggerMixin (*loggerContext*)

Mixin that provides logger functionality. It is safe to be initialized multiple times.

Arguments

- **loggerContext** (*String*) – See `JQuery.Logger()`.

read-only logger

Type `JQuery.Logger`

The logger for this view.

createSubLogger (*name*)

Creates a new `JQuery.Logger` as a sub-logger of the current *logger*.

Arguments

- **name** (*String*) – The name to append to the logger identifier. A leading "." is not required.

Returns The new sub-logger.

Return type JQuery.Logger

mixins/socket

SocketMixin

mixin **SocketMixin** (*socketContext*)

Mixin that provides websocket functionality.

Arguments

- **socketContext** (*String*) – See `SocketProxy.socketContext`.

read-onlysocket

Type SocketProxy

The contained socket proxy instance.

SocketProxy

class **SocketProxy** (*socketContext*)

Warning: This class is not exported and its instances are only available through the implementers of the `SocketMixin()`.

A lightweight socket proxy implementation that basically yields the emitter-like `emit()` and `on()` methods. Internally, these calls are forwarded to the global `vispa.socket` instance with `socketContext` being prefixed to all events.

Arguments

- **socketContext** (*String*) – See `socketContext`.

read-onlysocketContext

Type String

The socket context which is used to as a prefix for all events.

emit (*event*, ...)

Emits an *event* with all supplied arguments.

Arguments

- **event** (*String*) – The name of the event to emit to.

on (*event*, *callback*)

Listens to an *event* with a *callback*.

Arguments

- **event** (*String*) – The name of the event to listen to.

- **callback** (*Function*) – The callback function that is invoked with all arguments supplied in the corresponding event emission.

8.1.9 jsondata

class JSONData (*key* [, *workspace*])

Extends Emitter

The mannager of a JSONData object saved to the server.

Arguments

- **key** (*String*) – The key of the data-object
- **workspace** (*Workspace* | *Int*) – If given, the workspace or its id for which the object is stored.

read-onlykey

Type String

The key of the JSONData object.

read-onlywid

Type Int

The workspace id of the JSONData object.

data

Type

Any

The data content of the JSONData object.

pull ([, *callback*])

Instructs the JSONData to be fetched from the server. The callback is triggered when the pull has finished.

Emits pulled When the pull has finished.

push ([, *callback*])

Instructs the JSONData to be sent to the server. The callback is triggered when the push has finished. Will not do anything (except calling the callback) when there is nothing to do.

Emits pulled When the push has finished.

load (*infos*)

Sets up the JSONData objects for the given keys in infos. This will internally call `js:function:loadKey`.

Arguments

- **infos** (*Object* (*key*=>*JSONDataInfos*)) – Has a *infos* objects with the same syntax as in `loadKey()` for every *key*.

loadKey (*key*, *JSONDataInfos*)

Sets up the JSONData obejct for the given workspaces in JSONDataInfos.

Arguments

- => **JSONDataInfo** **JSONDataInfos** (*Object* (*workspace.id*) – Every *key* holds a JSONData info object (and internal data collection) from which it is initialized.

getInst (*key* [, *workspace*])

Retrives the JSONData instance for the given key and workspace. It will be created if it does not already exist. This should be preferred over directly instantiating `JSONData()` itself.

Arguments

- **key** (*String*) – The key of the JSONData.
- **workspace** (*Workspace* | *Int* (*workspace.id*)) – The workspace the JSONData is bound to. If omitted, it will be global.

Returns The requested JSONData.

Return type JSONData

syncObj (*key* [, *push*])

Access to synchronous data objects for a given *key*. To be used with VueJS.

Arguments

- **key** (*String*) – The key of the JSONData group to update.
- **push** (*Boolean*) – If true, all the JSONData objects will be instructed to push their data.

Returns The synchronous data object for the given *key*.

Return type Object(workspace.id => Any)

8.1.10 filehandler

class FileHandler ()

The file handler for vispa. One can add `MenuItem()` like objects, which can be used within a menu (like headless) or directly. `getFileHandlers()` then returns all registered file handlers with the property *value* set to the filename, which is given to the function as parameter. The handlers themselves should update themselves via the filename. Classes deriving from `BaseView()` can set the member property *fileHandlers*, which must be a list of `MenuItem()` like objects. These are automatically added to the file handler. Additionally, `BaseView()` adds `spawn()` to the handler so that inside `callback()` *this.spawn* can be used to easily spawn an instance.

addFileHandler (*key*, *handler*)

Add a single file handler with key.

addFileHandlers (*handlers*)

Add a dict of file handlers.

getFileHandlers (*file*)

Get all file handlers. Each is updated previously with *handler.value = file*.

getDefaultFileHandler (*file*)

Get the file handler with the highest position or *null* if it has no callback, e.g. for a divider or a header.

runDefaultHandler (*file*, *view*)

Runs the default file handler if available for the given view.

Arguments

- **file** (*String*) – The file to run the handler for.
- **view** (*BaseView*) – The view that intends to open the file.

8.1.11 module

class **Module** (*name*, *prefs*)

Extends Emitter

Mixins Logger

Mixins Preferences

A class for internal modules without any visual representation.

Arguments

- **name** (*String*) – The name of the module.
- **prefs** (*Object*) – The preference to use for this module. See [PreferencesMixin\(\)](#) for details.

Throws **Error** – If the name is not a string

read-only**name**

Type String

The name of the module.

8.1.12 prefs

8.1.13 views

views/dialog

class **DialogView** (*args*, *viewArgs*, *baseArgs*)

Extends BaseView

This view is opened in a dialog. Usually within a [MainView\(\)](#).

Arguments

- **args** (*Object*) – See [BaseView\(\)](#).
- **viewArgs** (*Object*) – See [BaseView\(\)](#).
- **baseArgs** (*Object*) – See [BaseView\(\)](#).

dialogExtraOpts

Type Object

An object providing the base vue options for the underlying [DialogViewDialog](#).

read-only**\$content**

Type jQuery

The jQuery object containing the node that is holding the views content.

read-only**menu**

Type MenuButton

The vue instance representing the menu of this view.

isVisible()

Returns Wether this view is visible. Always true.

Return type Bool

isFocused()

Returns Wether this view has focus.

Return type Bool

icon

Type String

The CSS class names assigned to the icon of the dialog.

label

Type String

The label of the dialog, displayed as title

setLabel(*label*)

Sets the label. For compatiblity with `MainView`'s :js:func:`setLabel()`.

Arguments

- **label** (*String*) – The label to be set.

setLoading(*loading*[, *delay*])

Behaves just like the normal `setLoading` with the only difference that you can provide a delay to the visual representation of loading (when not loading previously). It is inthendes to be used with low delays (such as 0.2s) to hold up the illusion of a very responsive UI.

Arguments

- **loading** (*Bool*) – Wether to increase or decrease the internal loading semaphore.
- **delay** (*Number*) – The delay in seconds until the loading indicator will be shown after it was not already.

views/base

class BaseView (*args*, *viewArgs*, *baseArgs*)

The base class for all views. The decendants are expected to implement the details of the visual representation.

Extends UICollection

Mixins AjaxMixin

Mixins LinkMixin

Mixins LoadMixin

Mixins LoggerMixin

Mixins PreferencesMixin

Mixins ShortcutsMixin

Mixins SocketMixin

Mixins StateMixin

Arguments

- **args** (*Object*) – An Object containing any parameters for the specific View. A subset of it is usually used to initialize the state of the view.
- **viewArgs** (*Object*) – An Object containing parameters of interest (e.g. position) for the view kind (Dialog, Main, SideBar).
- **viewArgs.callee** (*BaseView*) – The view that opened this view. May be *undefined* if not applicable.
- **baseArgs** (*Object*) – An Object containing parameters of interest for this class.
- **baseArgs.workspace** (*Workspace*) – The *Workspace()* this view is to be linked to.
- **baseArgs.baseContext** (*String*) – See *baseContext*. If not present will be generated automatically.
- **baseArgs.defaultQuery** (*String*) – See *defaultQuery*. If not present it will be generated automatically.

Throws

- **Error** – If the *baseArgs* is not an object or it is missing critical parameters.
- **Error** – If the *maxInstances* is reached.

Note: All classes extending this one can simply call the the init super by using:

```
init._super.apply(this, arguments);
```

Note: The shortcuts are disabled by default.

read-onlyid

Type String

The id of the view.

read-onlyrid

Type Number

A running id.

read-onlybaseContext

Type String

An identifier string used for *logger*, *SocketMixin()*, and *ShortcutsMixin()*.

processArgs (args, viewArgs, baseArgs)

For singletons, this function will be called automatically when instantiation of it was requested again by spawn, in order to be able to process the arguments anyway. Will be called before focus.

Note: This function should be overridden if needed.

Note: For information on the arguments, see *init()*.

messageContainer**Type** JQuery

The JQuery object (containing exactly one DOM node) to contain the messages/dialogs.

Throws Error If attempting to set it to an invalid value.

read-onlymessageCount**Type** Number

The of buffered messages for this view in the *vispa.messenger*.

alert (*message* [, *opts*])

See *alert()* of *Dialog()*

prompt (*message*, *callback* [, *opts*])

See *prompt()* of *Dialog()*

confirm (*message*, *callback* [, *opts*])

See *confirm()* of *Dialog()*

dialog (*opts*)

See *dialog()* of *Dialog()*

focus (*opts*)

Will focus the view. A *ViewManager()* may also call *show()* in the process.

Arguments

- **opts** – This may be used to pass further options to the view manager.

Emits focus(opts) This event is used by the *ViewManager()* and can also be used by others.

blur ()

Will blur (lose focus) the view.

Emits blur This event is used by the *ViewManager()* and can also be used by others.

show (*opts*)

Will show the view. If it is not *loaded* is will call *load()* automatically with itself provided as callback.

Arguments

- **opts** – This may be used to pass further options to the view manager.

Emits show(opts) This event is used by the *ViewManager()* and can also be used by others.

hide ()

Will hide the view. A *ViewManager()* may also call *blur()* in the process.

Emits hide This event is used by the *ViewManager()* and can also be used by others.

close (*force*)

Will close the view. A *ViewManager()* may also call *blur()* and *hide()* in the process. Implies the destruction of the instance - it must not be used thereafter.

Arguments

- **force** (*Bool*) – If this is *true* *onBeforeClose()* will not be consulted.

Emits close This event is used by the *ViewManager()* and can also be used by others.

destroy ()

Destroys the instance. It must not be used thereafter.

render (*node*)

The function to render/refresh the displayed content.

Arguments

- **node** (*JQuery*) – The JQuery object containing the one DOM node in which the content is to be rendered.

Note: Is to be overridden.

read-onlyname

Type String

The name of the view. Same as *_class._members.name*.

read-onlyextension

Type Extension

The extension of the view. Same as *_class._members.extension*.

spawnInstance (*extName*, *viewName* [, *args* [, *viewArgs* [, *baseArgs*]]])

Spawn a new view from the given parameters.

Arguments

- **extName** (*String*) – The name of the extension that has the view.
- **viewName** (*String*) – The name of the view to open.
- **args** (*Object*) – The args to be provided to the view.
- **viewArgs** (*Object*) – The viewArgs to be provided to the view.
- **baseArgs** (*Object*) – The baseArgs to be provided to the view.

Returns The view that was opened. It is most likely not ready to use yet.

Return type BaseView

Throws Error – If either the extension of the view was not found.

fileURL (*path*)

Retrieves the URL to directly read a file from the workspace.

Returns The full URL.

Return type String

Throws Error – if this view is not bound to a workspace

spawn (*args*, *viewArgs*, *baseArgs* [, *opts*])

A singleton safe spawning method for a new view. For an existing singleton only the *focus* method will be called.

Arguments

- **args** (*Object*) – Passed on to *baseView()* instantiation (if required).
- **viewArgs** (*Object*) – Passed on to *baseView()* instantiation (if required).
- **baseArgs** (*Object*) – Passed on to *baseView()* instantiation (if required).
- **opts** (*Object*) – Options passed to the *focus()* methods.

Returns The view spawned (or activated in the case of an existing singleton).

Return type BaseView

read-only class-membername

Type String

The name of the view. Must be overridden by extending classes.

Throws Error If it is not overridden.

read-only class-membermaxInstances

Type Number

Value Infinity

The maximum allowed count of concurrent instances. Should be overridden as needed.

read-only class-memberextension

Type Extensionnull

Value null

The extension this view is linked to by default. Will be updated automatically when registering the view to an extension.

read-only class-memberinstances

Type Array(BaseView)

A list of all current instances of this class.

read-only class-membercount

Type Number

The count of instances of this class currently active.

closeAllInstances ()

Closes all open instance of this viewClass.

class-memberworkspace

Type The default workspace binding type for this view. If it is *null* this view will only be global (no workspace binding). If it is a number, it will always be bound to a workspace. In any other case, it can be both.

views/side

class SidebarView (*args*, *viewArgs*, *baseArgs*)

Extends BaseView

The view used for the sidebar. Every view is a singleton

Arguments

- **args** (*Object*) – An Object containing any parameters for the specific View. A subset of it is usually used to initialize the state of the view.

read-onlytoggle

Type jQuery

The VueComponent of the toggle.

read-only\$content

Type jQuery

The jQuery object containing the node that is holding the views content.

showSidebar()

Will show the whole sidebar, same as setting *vispa.sidebar.active = true*.

hide()

Will hide the whole sidebar, same as setting *vispa.sidebar.active = false*.

show()

Will show the view, same as putting the toggle to state *selected = true*. It also calls *show()* of *BaseView()*.

hide()

Will hide the view, same as putting the toggle to state *selected = false*. It also calls *hide()* of *BaseView()*.

updatePopovers()

Update popovers. Shortcut for *sidebar.updatePopovers*.

views/main

class *MainView* (*args*, *viewArgs*, *baseArgs*)

Extends BaseView

The view used for most things. It's instances are managed by a *ViewManager()* instance and allow for arbitrary position and size within the grid.

Arguments

- **args** (*Object*) – See *BaseView()*.
- **viewArgs** (*Object*) – See *BaseView()* and *attach()* of *ViewManager()*.
- **baseArgs** (*Object*) – See *BaseView()*.

read-only\$tab**Type** jQuery

The jQuery object holding the tab associated with this instance.

read-only\$content**Type** jQuery

The jQuery object containing the node that is holding the views content.

read-onlymenu**Type** MenuButton

The vue instance representing the menu of this view.

isManaged()

Returns Whether this view is managed. If it is not applicable (no *master*) will return *null*.

Return type Boolean

isVisible()

Returns Whether this view has focus. If it is applicable (not *isManaged()*) will return *null*.

Return type Boolean

isFocused()

Returns Whether this view has focus. If it applicable (not *isManaged()*) will return *null*.

Return type Boolean

icon

Type String

The CSS class names assigned to the icon of the tab.

label

Type String

The label that is displayed in the tab. Setting this is the same as calling *setLabel()* with the new label and *false* as second parameter.

setLabel(label[, isPath])

Sets the label to a new value. If *isPath* is true it will interpret the name as a path. The path will be shortened as much as possible while avoiding the labels of peers.

Arguments

- **label** (*String*) – The new label to be set.
- **isPath** (*Boolean*) – Whether the label is to be interpreted as a path.

modified

Type Boolean

Whether the tabs closing icon (cross) should indicate that the contents of this were modified but not yet saved by replacing it with an appropriate icon (pencil).

setLoading/loading[, delay])

Behaves just like the normal *setLoading* with the only difference that you can provide a delay to the visual representation of loading (when not loading previously). It is intended to be used with low delays (such as 0.2s) to hold up the illusion of a very responsive UI.

Arguments

- **loading** (*Boolean*) – Whether to increase or decrease the internal loading semaphore.
- **delay** (*Number*) – The delay in seconds until the loading indicator will be shown after it was not already.

Note: All following functions are to be overridden as need. Calling their *_super* is not required.

getFragment()

Returns the fragment representing the the most important parts of the state of this view. Should use *getState()* to acquire the information for assembling the fragment.

Returns The fragment.

Return type String

applyFragment(fragment)

Loads a state from the fragment given. Should use *state.set()* to forward the information acquired through the fragment. It must silently fail if the given fragment is malformed in any way.

Arguments

- **fragment** (*String*) – The fragment the state is to be reconstructed from.

onFocus (*opts*)

Bound to the *focus* event.

Arguments

- **opts** – The same parameter as provided in *focus()* and it's *focus* event.

onBlur ()

Bound to the *blur* event.

onShow (*opts*)

Bound to the *show* event.

Arguments

- **opts** – The same parameter as provided in *show()* and in it's *show* event.

onHide ()

Bound to the *hide* event.

onClose ()

Bound to the *close* event.

onBeforeClose ()

This function is **always** called by *close()* to determine whether the closing of the view should be allowed. For example, if there are unsafe changes one use this function to check for them and trigger the safe process to prevent the changes from being lost.

Returns If *false* is returned the closing will not be performed.

Return type Bool

onResize ()

Bound to the *resize* event.

mainMenuEntry

Type

Object|null

If available, this will be an automatically created object that can be used in the main menu.

8.2 Python API

8.2.1 vispa package

Subpackages

vispa.controller package

Submodules

vispa.controller.ajax module

class `vispa.controller.ajax.AjaxController` (*root*)

Bases: `vispa.controller.AbstractController`

addworkspace (*name, host, login, key=None, cmd=None*)

```
connectworkspace (wid, password=None)  
deleteworkspace (wid)  
disconnectworkspace (wid)  
editworkspace (wid, name=None, host=None, login=None, key=None, cmd=None)  
feedback (content, anonymous)  
forgotpassword (username)  
getjson (key, wid=None)  
getworkspacedata (wid=None)  
localuser ()  
login (username, password)  
register (username, email)  
setjson (key, value, wid=None)  
setpassword (hash, password)
```

vispa.controller.bus module

```
class vispa.controller.bus.BusController  
    Bases: vispa.controller.AbstractController  
  
    index (*args, **kwargs)  
  
    poll (timeoutms=10000)  
  
    send (*args, **kwargs)
```

vispa.controller.error module

```
class vispa.controller.error.ErrorController  
    Bases: object  
  
    This class provides custom error-catching functions that are inserted into the cherrypy config.  
  
    STATUSMAP = {'404': 'filenotfound', '403': 'forbidden', '401': 'unauthorized', '400': 'badrequest', '500': 'internalserver'  
    TMPL = "<html><head><meta http-equiv='refresh' content='0;url=%s' /></head></html>"  
  
    get_error_data ()  
  
    index (*args, **kwargs)
```

vispa.controller.filesystem module

```
class vispa.controller.filesystem.FSAjaxController (mount_static=True)  
    Bases: vispa.controller.AbstractController  
  
    checkpermissions (path)  
  
    compress (paths, path, name='', isTmp=False)  
  
    createfile (path, name)
```

createfolder (*path, name*)

decompress (*file*)

exists (*path, filetype=None*)

expand (*path*)

filecount (*path, watch_id=None*)

filelist (*path, filefilter=None, reverse=False, watch_id=None*)

getfacl (*path*)

Get the facl of a certain path.

Parameters **path** – path of interest

Returns list of tuples with (type, user, mode[, default mode])

getfile (*path, watch_id=None, utf8=False*)

getsuggestions (*path, length=10, append_hidden=True*)

getworkspaceini (*request, fail_on_missing=False*)

isbrowserfile (*path*)

move (*source, destination*)

paste (*path, paths, cut*)

remove (*path*)

rename (*path, name, new_name*)

savefile (*path, content, watch_id=None, utf8=False*)

setfacl (*path, type, name, mode, remove=False, recursive=False, default=False*)

Set the facl entry for a certain user or group. On remote side, the commandline tool ‘setfacl’ is used, so this method delivers an interface to that.

Parameters

- **path** – path of interest
- **type** – type, either ‘user’, ‘group’, ‘mask’ or ‘other’
- **name** – name of the user or group of interest
- **mode** – mode to be set
- **remove** – remove all extended facl entries (option ‘-x’)
- **recursive** – apply changes to all files and directories recursively (option ‘-R’)
- **default** – edit the default values of an facl entry (option ‘-d’)

Raises AjaxException if type is not ‘user’, ‘group’, ‘mask’ or ‘other’

setworkspaceini (*request*)

unwatch (*watch_id=None*)

upload (**args, **kwargs*)

watch (*path, watch_id*)

class vispa.controller.filesystem.**FSController**

Bases: *vispa.controller.AbstractController*

getfile (*path*, *download=None*, *deleteoncomplete=None*, ***kwargs*)

thumbnail (*path*, *width=100*, *height=100*, ***kwargs*)

vispa.controller.root module

class `vispa.controller.root.RootController` (*server*)

Bases: `vispa.controller.AbstractController`

graceful_shutdown (**args*, ***kwargs*)

guest_login (**args*, ***kwargs*)

index (**args*, ***kwargs*)

login (**args*, ***kwargs*)

logout (*path='/'*)

mount_extension_controller (*mountpoint*, *controller*)

password (*hash*, **args*, ***kwargs*)

status (**args*, ***kwargs*)

workspace_data (*workspace=None*, *keys=None*)

vispa.controller.usermanagement module

class `vispa.controller.usermanagement.UMAjaxController` (*mount_static=True*)

Bases: `vispa.controller.AbstractController`

The UMAjaxController inherits all methods, which are necessary for the user management. Most of the function names are chosen self explanatory, while the first word corresponds to the object of interest and the rest to the action performed on the object, e.g. `user_get_groups` returns the groups of the user, who created the request. If the returned Objects are (lists of) database classes, they are converted into python/JSON objects with the values of interest as properties, e.g.: `user_get_groups()` returns a list of objects with “name” and “status” as properties.

group_add_child_group (*parent_group*, *child_group*)

Add child group to parent group.

Parent_group concerning parent group

Child_group concerning child group

Raises AjaxException

group_add_manager (*group*, *user*)

Add manager to group.

Parameters

- **group** – concerning group
- **user** – user to add as manager

Raises AjaxException

group_add_parent_group (*parent_group*, *child_group*)

Add parent group to child group.

Parent_group concerning parent group

Child_group concerning child group

Raises AjaxException

group_add_user (*group*, *user*)

Add user to group

Parameters

- **group** – concerning group
- **user** – user to be added

Raises AjaxException

group_confirm_child_group (*parent_group*, *child_group*)

Confirm child group in parent group.

Parent_group concerning parent group

Child_group concerning child group

Raises AjaxException

group_confirm_user (*group*, *user*)

Confirm user.

Parameters

- **group** – concerning group
- **user** – user to be confirmed

Raises AjaxException

group_create (*name*, *privacy=0*, *password=''*)

Create group.

Parameters **name** – name of group

Privacy *privacy*

Password *password*

Raises AjaxException

group_delete (*group*)

Delete group.

Parameters **group** – concerning group

Raises AjaxException

group_enter_parent_group (*parent_group*, *child_group*, *password=''*)

Enter parent group as child group.

Parent_group concerning parent group

Child_group concerning child group

Parameters **password** – password if parent group is protected

Raises AjaxException

group_get (*group*)

Get group by its name.

Parameters **group** – name of the group

Returns dict with “name”, “privacy” and “status”

Raises AjaxException

group_get_all ()

Get all groups.

Returns list of dict with “name”, “privacy” and “status”

group_get_child_groups (*group*, *recursion_depth=-1*)

Get child groups of group.

Parameters **group** – concerning group

Recursion_depth recursion depth for getting child groups

Returns list of dict with “name” and “status” of membership

Raises AjaxException

group_get_managers (*group*)

Get managers of group.

Parameters **group** – concerning group

Returns list of strings (names of managers)

Raises AjaxException

group_get_parent_groups (*group*, *recursion_depth=-1*)

Get parent groups of group.

Parameters **group** – concerning group

Recursion_depth recursion depth for getting parent groups

Returns list of dict with “name” and “status” of membership

Raises AjaxException

group_get_users (*group*, *recursion_depth=-1*)

Get users of group.

Parameters

- **group** – concerning group
- **recursion_depth** – recursion_depth for getting users

Returns list of dict with “name” and “status” and membership

Raises AjaxException

group_leave_parent_group (*parent_group*, *child_group*)

Leave parent group as child group.

Parent_group concerning parent group

Child_group concerning child group

Raises AjaxException

group_remove_child_group (*parent_group*, *child_group*)

Remove child group from parent group.

Parent_group concerning parent group

Child_group concerning child group

Raises AjaxException

group_remove_manager (*group, manager*)

Remove manager from group.

Parameters

- **group** – concerning group
- **manager** – manager to be removed

Raises AjaxException

group_remove_user (*group, user*)

Remove user.

Parameters

- **group** – concerning group
- **user** – user to be removed

Raises AjaxException

group_rename (*group, name*)

Rename group.

Parameters

- **group** – concerning group
- **name** – new name

Raises AjaxException

group_set_password (*group, password*)

Set password of group.

Parameters

- **group** – concerning group
- **password** – new password

Raises AjaxException

group_set_privacy (*group, privacy*)

Set privacy of group.

Parameters

- **group** – concerning group
- **privacy** – new privacy

Raises AjaxException

group_set_status (*group, status*)

Set status of group.

Parameters

- **group** – concerning group
- **status** – new status

permission_create (*name*)

Create permission.

Parameters **name** – name of new permission

Raises `AjaxException`

permission_delete (*permission*)

Delete permission.

Parameters **permission** – concerning permission

Raises `AjaxException`

permission_get_all ()

Get all permissions.

Returns list of strings

Raises `AjaxException`

permission_rename (*permission, name*)

Rename permission.

Parameters

- **permission** – concerning permission
- **name** – new name of permission

Raises `AjaxException`

project_add_group (*project, group*)

Add group to project.

Parameters

- **project** – concerning project
- **group** – group to be added

Raises `AjaxException`

project_add_manager (*project, user*)

Add manager to project.

Parameters

- **project** – concerning project
- **user** – user to be added as manager

Raises `AjaxException`

project_add_user (*project, user*)

Add user to project.

Parameters

- **project** – concerning project
- **user** – user to be added

Raises `AjaxException`

project_create (*name*)

Create project.

Parameters **name** – name of new project

Raises `AjaxException`

project_delete (*project*)
Delete project.

Parameters **project** – concerning project

Raises AjaxException

project_get_all ()
Get all projects.

Returns list of dict with “name” and “status”

Raises AjaxException

project_get_groups (*project*)
Get groups of project.

Parameters **project** – concerning project

Returns list of dict with “name”

Raises AjaxException

project_get_managers (*project*)
Get managers of project.

Parameters **project** – concerning project

Returns list of dict with “name”

Raises AjaxException

project_get_roles_of_group (*project, group*)
Get roles of group inside project.

Parameters

- **project** – concerning project
- **group** – concerning group

Returns list of strings

Raises AjaxException

project_get_roles_of_user (*project, user*)
Get roles of user inside project.

Parameters

- **project** – concerning project
- **user** – concerning user

Returns list of strings

Raises AjaxException

project_get_users (*project*)
Get users of project.

Parameters **project** – concerning project

Returns list of dict with “name”

Raises AjaxException

project_remove_group (*project, group*)
Remove group from project.

Parameters

- **project** – concerning project
- **group** – group to be removed

Raises `AjaxException`**project_remove_manager** (*project, manager*)

Remove manager from project.

Parameters

- **project** – concerning project
- **manager** – manager to be removed

Raises `AjaxException`**project_remove_user** (*project, user*)

Remove user from project.

Parameters

- **project** – concerning project
- **user** – user to be removed

Raises `AjaxException`**project_rename** (*project, name*)

Rename project.

Parameters

- **project** – concerning project
- **name** – new name of project

Raises `AjaxException`**project_set_roles_of_group** (*project, group, roles*)

Set roles of group inside project.

Parameters

- **project** – concerning project
- **group** – concerning group
- **roles** – JSON encoded list of roles with “name” and “assignment” bool

Raises `AjaxException`**project_set_roles_of_user** (*project, user, roles*)

Set roles of user inside project.

Parameters

- **project** – concerning project
- **user** – concerning user
- **roles** – JSON encoded list of roles with “name” and “assignment” bool

Raises `AjaxException`**project_set_status** (*project, status*)

Set status of project.

Parameters

- **project** – concerning project
- **status** – new status of project

Raises AjaxException**role_create** (*name*)

Create role.

Parameters **name** – name of new role**Raises** AjaxException**role_delete** (*role*)

Delete role.

Parameters **role** – concerning role**Raises** AjaxException**role_get_all** ()

Get all roles.

Returns list of strings**role_get_permissions** (*role*)

Get permissions of role.

Parameters **role** – concerning role**Returns** list of strings**Raises** AjaxException**role_rename** (*role*, *name*)

Rename role.

Parameters

- **role** – concerning role
- **name** – new name of role

Raises AjaxException**role_set_permissions** (*role*, *permissions*)

Set permissions of role.

Parameters

- **role** – concerning role
- **permissions** – JSON encoded list of permissions with “name” and “assignment” bool

Raises AjaxException**user_enter_group** (*group*, *password*=’')

Enter group as current user.

Parameters

- **group** – concerning group
- **password** – password for protected groups

Raises AjaxException

user_get_groups ()

Get groups of the current user.

Returns list of dict with “name” and “status”

user_get_managed_groups ()

Get managed groups of current user.

Returns list of dict with “name”, “privacy” and “status”

user_get_managed_projects ()

Get managed projects of current user.

Returns list of dict with “name” and “status”

user_get_managed_workgroups ()

Get managed workgroups of current user.

Returns list of dict with “name”

user_get_permissions (*project*)

Get permissions of current user in project.

Parameters **project** – concerning project

Returns list of strings

Raises AjaxException

user_get_projects ()

Get projects of current user.

Returns list of dict with “name” and “status”

user_get_roles (*project*)

Get roles of current user in project.

Parameters **project** – concerning project

Returns list of dict with “name”

Raises AjaxException

user_get_workgroups ()

Get workgroups of current user.

Returns list of dict with “name”

user_leave_group (*group*)

Leave group as current user.

Parameters **group** – concerning group

Raises AjaxException

user_leave_workgroup (*workgroup*)

Leave workgroup as current user.

Raises AjaxException

workgroup_add_manager (*workgroup*, *user*)

Add manager to workgroup.

Parameters

- **workgroup** – concerning workgroup
- **user** – user to be added as manager

Raises AjaxException

workgroup_add_user (*workgroup*, *user*)

Add user to workgroup.

Parameters

- **workgroup** – concerning workgroup
- **user** – user to be added

Raises AjaxException

workgroup_create (*name*)

Create workgroup.

Parameters **name** – name of new workgroup

Raises AjaxException

workgroup_delete (*workgroup*)

Delete workgroup.

Parameters **workgroup** – workgroup to be deleted

Raises AjaxException

workgroup_get_managers (*workgroup*)

Get managers of workgroup.

Parameters **workgroup** – concerning workgroup

Returns list of dict with “name”

Raises AjaxException

workgroup_get_users (*workgroup*)

Get users of workgroup.

Parameters **workgroup** – concerning workgroup

Returns list of dict with “name”

Raises AjaxException

workgroup_remove_manager (*workgroup*, *manager*)

Remove manager from workgroup.

Parameters

- **workgroup** – concerning workgroup
- **manager** – manager to be removed

Raises AjaxException

workgroup_remove_user (*workgroup*, *user*)

Remove user from workgroup.

Parameters

- **workgroup** – concerning workgroup
- **user** – user to be removed

Raises AjaxException

workgroup_rename (*workgroup*, *name*)

Rename workgroup.

Parameters

- **workgroup** – concerning workgroup
- **name** – new name of workgroup

Raises `AjaxException`

Module contents

```
class vispa.controller.AbstractController (mount_static=True)
    Bases: object

    cache (workspace_id, key)

    convert (value, flag)

    get (key, *args, **kwargs)

    mount_static (path=None, url='static')

    release ()

    release_database ()

    release_session ()

    set_cache (workspace_id, item, key=None)

class vispa.controller.StaticController (path)
    Bases: object

vispa.controller.strongly_expire (func)
    Decorator that sends headers that instruct browsers and proxies not to cache.
```

vispa.extensions package**Subpackages****vispa.extensions.codeeditor package****Subpackages****vispa.extensions.codeeditor.workspace package****Module contents**

```
class vispa.extensions.codeeditor.workspace.CodeEditorRpc (window_id, view_id)

    BURST_BUFFER = 8000

    BURST_DELAY = 0.25

    MAX_BURST = 1000

    MAX_RATE = 2000

    SIGTEM_SIGKILL_DELAY = 0.1
```

```
    abort ()
    close ()
    runningjob ()
    start (cmd, base)

vispa.extensions.codeeditor.workspace.expand (path)
```

Submodules

vispa.extensions.codeeditor.controller module

```
class vispa.extensions.codeeditor.controller.EditorController (mount_static=True)
    Bases: vispa.controller.AbstractController
    abort ()
    close ()
    execute (cmd, base)
    getrpc ()
    runningjob ()
```

Module contents

```
class vispa.extensions.codeeditor.CodeEditorExtension (server)
    Bases: vispa.server.AbstractExtension
    dependencies ()
    name ()
    setup ()
```

vispa.extensions.core package

Module contents

```
class vispa.extensions.core.CoreController (mount_static=True)
    Bases: vispa.controller.AbstractController
class vispa.extensions.core.CoreExtension (server)
    Bases: vispa.server.AbstractExtension
    dependencies ()
    name ()
    setup ()
```

vispa.extensions.demo package

Subpackages

vispa.extensions.demo.workspace package

Module contents

```
class vispa.extensions.demo.workspace.DemoRpc
```

```
    ls (path)
```

Submodules

vispa.extensions.demo.controller module

```
class vispa.extensions.demo.controller.DemoController (mount_static=True)  
    Bases: vispa.controller.AbstractController
```

```
    ls (path=None)
```

Module contents

```
class vispa.extensions.demo.DemoExtension (server)  
    Bases: vispa.server.AbstractExtension
```

```
    dependencies ()
```

```
    name ()
```

```
    setup ()
```

vispa.extensions.dummy package

Subpackages

vispa.extensions.dummy.workspace package

Module contents

```
class vispa.extensions.dummy.workspace.DummyRpc
```

```
    dummy ()
```

```
    wait (cb)
```

```
class vispa.extensions.dummy.workspace.Scheduler
```


Submodules

vispa.extensions.dummy.controller module

```
class vispa.extensions.dummy.controller.DummyController (extension)
    Bases: vispa.controller.AbstractController

    data ()

    failure (msg=None)

    sigtest (o, l, i, s)
```

Module contents

```
class vispa.extensions.dummy.DummyExtension (server)
    Bases: vispa.server.AbstractExtension

    dependencies ()

    name ()

    setup ()
```

vispa.extensions.file package

Submodules

vispa.extensions.file.controller module

```
class vispa.extensions.file.controller.FileController
    Bases: vispa.controller.AbstractController
```

Module contents

```
class vispa.extensions.file.FileBrowserController (mount_static=True)
    Bases: vispa.controller.AbstractController

class vispa.extensions.file.FileBrowserExtension (server)
    Bases: vispa.server.AbstractExtension

    dependencies ()

    name ()

    setup ()
```

vispa.extensions.gallery package

Module contents

```
class vispa.extensions.gallery.GalleryController (mount_static=True)
    Bases: vispa.controller.AbstractController
```

```
class vispa.extensions.gallery.GalleryExtension (server)
    Bases: vispa.server.AbstractExtension

    dependencies ()

    name ()

    setup ()
```

vispa.extensions.ldap-export package

Module contents

vispa.extensions.terminal package

Subpackages

vispa.extensions.terminal.workspace package

Module contents

```
class vispa.extensions.terminal.workspace.Terminal
    Bases: object

    close ()

    communicate (input_data, timeout=0.05)

    open (window_id, view_id, shell=None)

    read (timeout=0.05, buffer_size=16384)

    resize (w, h)

    write (input_data)
```

Module contents

```
class vispa.extensions.terminal.TerminalController (mount_static=True)
    Bases: vispa.controller.AbstractController

    close (tid)

    communicate (tid, input_data)

    open ()

    read (tid, timeout=10)

    resize (tid, w, h)

    write (tid, input_data)

class vispa.extensions.terminal.TerminalExtension (server)
    Bases: vispa.server.AbstractExtension

    dependencies ()

    name ()
```

```
setup()
```

Module contents

vispa.models package

Subpackages

vispa.models.alembic package

Submodules

vispa.models.alembic.env module

Module contents

```
vispa.models.alembic.migrate(db, revision='head')
```

Submodules

vispa.models.group module

```
class vispa.models.group.Group_User_Assoc(**kwargs)
    Bases: sqlalchemy.ext.declarative.api.Base
```

The Group_User_Assoc object is a association object representing the membership of a user in a group. In addition to the membership itself, it has a status flag, which indicates, whether the membership is confirmed or not.

```
CONFIRMED = 0
```

```
UNCONFIRMED = 1
```

```
group_id
```

```
status
```

```
user
```

```
user_id
```

```
class vispa.models.group.Group_Group_Assoc(**kwargs)
    Bases: sqlalchemy.ext.declarative.api.Base
```

The Group_Group_Assoc object is a association object representing the membership of one group in another, called child group and parent group. In addition to the membership itself, it has a status flag, which indicates whether the membership is confirmed or not.

```
CONFIRMED = 0
```

```
UNCONFIRMED = 1
```

```
child_group
```

```
child_group_id
```

```
parent_group_id
```

status

```
class vispa.models.group.Group (**kwargs)
    Bases: sqlalchemy.ext.declarative.api.Base
```

A group is a collection of users and other groups. Every group has an id, a unique name and a creation timestamp. Furthermore there is a privacy integer as follows:

0 - public: the group and its members can be seen by everyone and everyone can join
1 - protected: the group is visible, members only for other members, joining via request to group manager
2 - private: group and members are invisible, joining via password

A group is organized by managers, who can edit the name, privacy etc. as well as the memberships of users and group. Additionally, the managers can join the group into a parentgroup.

ACTIVE = 1

DELETED = 2

INACTIVE = 0

PRIVATE = 2

PROTECTED = 1

PUBLIC = 0

add_child_group (*session*, *child_group*, *confirmed=1*)

Add child group to parent group. Loops in groups are permitted. The necessary Group_Group_Assoc object is added to the database.

Parameters

- **session** – current session of database
- **child_group** (*Group*) – concerning child group

Raises TypeError of child_group is not instance of Group

Raises Exception if a loop is detected

Raises Exception if child_group is already in the group

add_manager (*user*)

Add manager to group.

Parameters **user** (*User*) – user which has to be added

Raises TypeError if user is not instance of User

Raises Exception if user is already manager of the group,

add_user (*session*, *user*, *confirmed=1*)

Add user to group. The necessary Group_User_Assoc object is added to the database. By default, the membership is unconfirmed.

Parameters

- **session** – current session of database
- **user** (*User*) – user which has to be added

Raises TypeError if user is not instance of User

Raises Exception if user already in group

static all (*session*)

Get all groups.

Parameters `session` – current session of database

Returns list of all Group objects

child_groups

confirm_child_group (*child_group*)

Confirm a child group in a private parent group.

Parameters `child_group` (`Group`) – concerning child group

Raises `TypeError` if `child_group` is not instance of `Group`

Raises `Exception` if `child_group` is not in the group

confirm_user (*user*)

Confirm a user in a private group.

Parameters `user` (`User`) – concerning user

Raises `TypeError` if `user` is not instance of `User`

Raises `Exception` if `user` not in group

created

delete ()

Delete group. Internally, the delete flag is set, its not deleted from the database.

static get (*session, group*)

Get a group by name or id. If the group parameter is an instance of `Group`, it is directly returned.

Parameters

- **session** – current session of database
- **group** – name or id to look for. if group is instance of `Group`, group is returned

Returns `Group`

Raises `Exception` if group parameter invalid or no group can be found

static get_by_id (*session, gid*)

Get a group by its id.

Parameters

- **session** – current session of database
- **gid** – id to look for

Returns `Group` or `None` if inexistent

static get_by_name (*session, name*)

Get a group by its name.

Parameters

- **session** – current session of database
- **name** – name to look for

Returns `Group` or `None` if inexistent

get_child_groups (*recursion_depth=-1*)

Get child groups of a parent group. This function works as `get_users` regarding the recursion depth.

Parameters **recursion_depth** – number of steps for recursion

Returns Set of Group_Group_Assoc objects

get_managers ()

Get managers of the group.

Returns list of User objects, which are managers of the group

static get_or_create_by_name (*session, name, privacy=0, password=''*)

Get a group by name or create it as public group, if it does not exists.

Parameters

- **session** – current session of database
- **name** – name of the group

Returns Group

Raises Exception if no group with name exists and either name or privacy are invalid

get_parent_groups (*recursion_depth=-1*)

Get parent groups (confirmed and unconfirmed) of a child group. This function works as get_child_groups but in opposite direction. The recursion is only done for confirmed memberships.

Parameters **recursion_depth** – number of steps for recursion

Returns Set of Group_Group_Assoc objects

get_projects ()

Returns active projects of the group and all (confirmed) parent groups.

Returns Set of Project_Group_Assoc objects

get_users (*recursion_depth=-1*)

Get all users of the group. The recursion depth for the child groups can be given as additional argument. E.g. 0 is only the group itself, 1 also includes all direct child groups. -1 belongs to infinite recursion depth. Notice: since loops in groups are permitted, there is only a finite number of recursions for a finite number of groups.

Parameters **recursion_depth** – number of recursions for child groups

Returns Set of Group_User_Assoc objects

id

managers

name

password

privacy

remove_child_group (*session, child_group*)

Remove child group from parent group. The concerning Group_Group_Assoc object is deleted from the database.

Parameters

- **session** – current session of database
- **child_group** (*Group*) – concerning child group

Raises TypeError of child_group is not instance of Group

Raises Exception if child_group is not in the group

remove_manager (*manager*)

Remove manager from group.

Parameters **manager** (*User*) – manager which has to be removed

Raises *TypeError* if manager is not instance of *User*

Raises *Exception* if manager is not manager of the group

remove_user (*session*, *user*)

Remove user from group. The concerning *Group_User_Assoc* object is deleted from the database.

Parameters

- **session** – current session of database
- **user** (*User*) – user which has to be removed

Raises *TypeError* if user is not instance of *User*

Raises *Exception* if user not in group

rename (*session*, *newname*)

Rename group.

Parameters

- **session** – current session of database, used to validate new name
- **newname** – new name of the group

Raises *Exception* if newname is invalid or already existent

set_password (*password*)

Set the password of a group.

Parameters **password** – new password

Raises *Exception* if password is invalid

set_privacy (*privacy*)

Set privacy of group.

Parameters **privacy** – new privacy of the group (0, 1, or 2)

Raises *Exception* if privacy is invalid

set_status (*status*)

Set status of group.

Parameters **status** – new status of the group, either 0 for inactive or 1 for active

Raises *Exception* if status is invalid

status

users

vispa.models.jsondata module

class `vispa.models.jsondata.JSONData` (***kwargs*)

Bases: `sqlalchemy.ext.declarative.api.Base`

get_info_data ()

static **get_item** (*db*, *user_id*, *key*, *workspace_id*, *create=False*)

```
static get_value (db, user_id, key, workspace_id)
static get_values_by_key (db, user_id, key=None)
id
key
static set_value (db, user_id, key, workspace_id, value)
timestamp
user_id
value
workspace_id
```

vispa.models.project module

```
class vispa.models.project.Project_User_Assoc (**kwargs)
    Bases: sqlalchemy.ext.declarative.api.Base
```

The Project_User_Assoc object is an association object, which connects a project and a user. It has a many-to-many relationship to roles, which gives the user roles inside the project. For the association table, the project and the user id are used.

```
get_permissions ()
    Get the permissions of this Project User connection.
```

Returns Set of Permission objects

```
project_id
roles
user
user_id
```

```
class vispa.models.project.Project_Group_Assoc (**kwargs)
    Bases: sqlalchemy.ext.declarative.api.Base
```

The Project_Group_Assoc object is an association object, which connects a project and a group. It has a many-to-many relationship to roles, which gives the group roles inside the project. For the association table, the project and the group id are used.

```
get_permissions ()
    Get the permissions of this Project Group connection.
```

Returns Set of Permission objects

```
group
group_id
project_id
roles
```

```
class vispa.models.project.Project (**kwargs)
    Bases: sqlalchemy.ext.declarative.api.Base
```

A project connects users and groups to some content (ProjectItems) and also assigns permissions to them via roles. Permissions can be e.g. read and write rights on the content.

ACTIVE = 1

DELETED = 2

INACTIVE = 0

add_group (*session*, *group*)

Adds group to project without any roles. The necessary Project_Group_Assoc object is added to the database.

Parameters

- **session** – current session of database
- **group** (*Group*) – concerning group

Raises TypeError if group is not instance of Group

add_manager (*user*)

Adds new manager to project.

Parameters **user** (*User*) – concerning user

Raises TypeError if user is not instance of User

add_roles_to_group (*group*, *roles*)

Add role to group.

Parameters

- **group** (*Group*) – concerning group
- **roles** (*list of Role objects*) – list of roles

Raises TypeError if type if group or roles is invalid

Raises Exception if group not in project

add_roles_to_user (*user*, *roles*)

Add role to a user.

Parameters

- **user** (*User*) – concerning user
- **roles** (*list of Role objects*) – list of roles

Raises TypeError if type if user or roles is invalid

Raises Exception if user not in project

add_user (*session*, *user*)

Adds user to project without any roles.

Parameters

- **session** – current session of database
- **user** (*User*) – concerning user

Raises TypeError if user is not instance of User

Raises Exception if user already in project

static all (*session*)

Returns all existing projects.

Parameters **session** – current session of database

Returns list of Project objects

static create (*session, name*)

Create new project.

Parameters

- **session** – current session of database
- **name** – name of the project

Raises Exception if name is invalid or project with same name already exists

created

delete ()

Delete project. Internally, the delete flag is set, its not really deleted.

static get (*session, project*)

Get a project by name. If the project parameter is an instance of Project, it is directly returned.

Parameters

- **session** – current session of database
- **project** – name to look for. If project is instance of Project it is directly returned

Returns Project object

Raises Exception if project parameter is invalid or no project can be found

static get_by_id (*session, gid*)

Get a project by its id.

Parameters

- **session** – current session of database
- **gid** – given id, which is looked for

Returns Project object or None if inexistent

static get_by_name (*session, name*)

Get a project by its name.

Parameters

- **session** – current session of database
- **name** – name which is looked for

Returns Project object or None if inexistent

get_groups ()

Get groups of project.

Returns list of Project_Group_Assoc objects

get_items (*itemtype=None*)

Get ProjectItems.

Parameters **itemtype** – optional selector on the item type

Returns list of ProjectItem objects

get_managers ()

Get managers of project.

Returns list of User objects

static `get_or_create_by_name` (*session*, *name*)

Get or create a project by its name.

Parameters

- **session** – current session of database
- **name** – name of the project

Returns Project

get_roles_of_group (*group*)

Return roles of group.

Parameters **group** (*Group*) – concerning group

Returns list of Role objects

Raises TypeError if group is not instance of Group

Raises Exception if group not in project

get_roles_of_user (*user*)

Return roles of user.

Parameters **user** (*User*) – concerning user

Returns list of Role objects

Raises TypeError if user is not instance of User

Raises Exception if user not in project

get_users ()

Returns users of project.

Returns list of Project_User_Assoc objects

groups

has_group (*group*)

Check if group is already in project.

Parameters **group** (*Group*) – concerning group

Returns bool whether group is in project

Raises TypeError if group is not instance of Group

id

items

managers

name

remove_group (*session*, *group*)

Removes group from Project.

Parameters

- **session** – current session of database
- **group** (*Group*) – concerning group

Raises TypeError if group is not instance of Group

Raises Exception if group is not in project

remove_manager (*manager*)

Removes manager from Project.

Parameters **manager** (*User*) – concerning manager

Raises *TypeError* if manager is not instance of *User*

Raises *Exception* if manager is not manager of project

remove_user (*session*, *user*)

Removes user from Project. The concerning *Project_User_Assoc* is deleted from the database.

Parameters

- **session** – current session of database

- **user** (*User*) – concerning user

Raises *TypeError* if user is not instance of *User*

Raises *Exception* if user not in project

rename (*session*, *newname*)

Rename project.

Parameters

- **session** – current session of database, used to validate new name

- **newname** – new name of the project

Raises *Exception* if newname is invalid or project with newname already exists

set_roles_of_group (*group*, *roles*)

Sets the roles of a group in a project.

Parameters

- **group** (*Group*) – concerning group

- **roles** (*list of Role objects*) – list of roles

Raises *TypeError* if type of group or roles is invalid

Raises *Exception* if group not in project

set_roles_of_user (*user*, *roles*)

Sets the roles of a user in a project.

Parameters

- **user** (*User*) – concerning user

- **roles** (*list of Role objects*) – list of roles

Raises *TypeError* if type of user or roles is invalid

Raises *Exception* if user not in project

set_status (*status*)

Set status of project.

Parameters **status** – new status of the project, either 0 for inactive or 1 for active

Raises *Exception* if status is invalid

status

users

```
class vispa.models.project.ProjectItem (**kwargs)
    Bases: sqlalchemy.ext.declarative.api.Base
```

A project item represents the actual content of a project. It is simply an object consisting of a type (e.g. file) and its content (e.g. the path of the file)

content

```
static create (session, project, itemtype, content)
```

Create an project item. The item is added to the database.

Parameters

- **session** – current session of database
- **project** – project of the new item
- **itemtype** – type of the item
- **content** – content of the item

```
delete (session)
```

Delete the item. It is removed from the database.

Parameters **session** – current session of database

```
static get (session, item)
```

Get a ProjectItem by its id. If the item is already a ProjectItem object, it is directly returned.

Parameters

- **session** – current session of database
- **item** – id to look for. if type of item is ProjectItem, item is returned

Returns ProjectItem

Raises Exception if item is invalid or no item can be found

```
static get_by_id (session, gid)
```

Get a ProjectItem by its id.

Parameters

- **session** – current session of database
- **gid** – given id, which is looked for

Returns ProjectItem or None if inexistent

```
get_project ()
```

Get the project of the item.

Returns Project

id

itemtype

project_id

```
set_content (content)
```

Set the content of the item.

Parameters **content** – new content

vispa.models.role module

class vispa.models.role.**Permission** (**kwargs)

Bases: sqlalchemy.ext.declarative.api.Base

A Permission object is self-explanatory a permission for a user. It is only characterized by its name, which explains its meaning.

static all (session)

Get all existing permissions.

Parameters **session** – current session of database

Returns list of Permission objects

static create (session, name)

Create new permission.

Parameters

- **session** – current session of database
- **name** – name of the permission

Raises Exception if name is invalid or already in use

created

delete (session)

Delete permission. It is also deleted from the database.

Parameters **session** – current session of database

static get (session, permission)

Get a permission by name. If permission is already a Permission object, it is directly returned.

Parameters

- **session** – current session of database
- **permission** – name to look for. if type of permission is Permission, it is returned

Returns Permission

Raises Exception if permission parameter is invalid or no permission can be found

static get_by_id (session, gid)

Get a permission by its id.

Parameters

- **session** – current session of database
- **gid** – given id, which is looked for

Returns Permission or None

static get_by_name (session, name)

Get a permission by its name.

Parameters

- **session** – current session of database
- **name** – given name, which is looked for

Returns Permission or None

static `get_or_create_by_name` (*session*, *name*)

Get or create a permission by name.

Parameters

- **session** – current session of database
- **name** – name of the permission

Returns Permission

id

name

rename (*session*, *newname*)

Rename Permission.

Parameters

- **session** – current session of database
- **newname** – new name of the permission

Raises Exception if newname is already in use

class `vispa.models.role.Role` (***kwargs*)

Bases: `sqlalchemy.ext.declarative.api.Base`

A Role is a collection of permissions (Permission objects).

add_permissions (*permissions*)

Add a list of permissions to the role.

Parameters **permissions** (*list of Permission objects*) – list of permissions

Raises TypeError if type of permissions is invalid

static **all** (*session*)

Get all existing permissions.

Parameters **session** – current session of database

Returns list of Role objects

static **create** (*session*, *name*)

Create new role.

Parameters

- **session** – current session of database
- **name** – name of the role

Raises Exception if name is invalid or already in use

created

delete (*session*)

Delete role. It is also deleted from the database.

Parameters **session** – current session of database

static **get** (*session*, *role*)

Get a role by name. If role is already a Role object, it is directly returned.

Parameters

- **session** – current session of database

- **role** – name to look for. if type of role is Role, role is returned

Returns Role

Raises Exception if role parameter is invalid or no role can be found

static **get_by_id** (*session*, *gid*)

Get a role by its id.

Parameters

- **session** – current session of database
- **gid** – given id, which is looked for

Returns Role or None

static **get_by_name** (*session*, *name*)

Get a role by its name.

Parameters

- **session** – current session of database
- **name** – given name, which is looked for

Returns Role or None

static **get_or_create_by_name** (*session*, *name*)

Get or create a role by name.

Parameters

- **session** – current session of database
- **name** – name of the role

Returns Role

id

name

permissions

rename (*session*, *newname*)

Rename role.

Parameters

- **session** – current session of database
- **newname** – new name of the role

Raises Exception if newname is already in use

set_permissions (*permissions*)

Sets the permissions of a role.

Parameters **permissions** (*list of Permission objects*) – list of permissions

Raises TypeError if type of permissions is invalid

vispa.models.user module

```

class vispa.models.user.User (**kwargs)
    Bases: sqlalchemy.ext.declarative.api.Base

    ACTIVE = 1

    FORBIDDEN_NAMES = ['data', 'guest', 'global', 'user', 'delete', 'select', 'insert', 'update', 'drop']

    INACTIVE = 0

    MIN_PW_LENGTH = 8

    NAME_CHARS = 'abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ01234567890_-+'

    NAME_LENGTH = [6, 30]

    PASSWORD_RESET_DELAY = 30

    active()

    static all(session)

    created

    email

    static forgot_password(db, name_or_mail)

    static generate_hash(length=10)

    static get(session, user)

    static get_by_email(session, email)

    static get_by_hash(session, hash)

    static get_by_id(session, uid)

    static get_by_name(session, name)

    get_groups()
        Get groups of a user. Only active groups are returned.

        Returns list of Group_User_Assoc objects

    get_managed_groups()
        Get managed groups of the user. Only not deleted groups are returned.

        Returns list of Group objects

    get_managed_projects()
        Get managed projects of the user.

        Returns list of Project objects

    get_managed_workgroups()
        Get managed workgroups of the user.

        Returns list of Workgroup objects

    static get_or_create_by_name(session, name, **kwargs)

    get_permissions(project)
        Get permissions of the user in the given project.

        Parameters project (Project) – concerning project

        Returns list of Permission objects

```

Raises `TypeError` if project is not instance of `Project`

Raises Exception if user not in project

get_projects ()

Get active projects of the user.

Returns Set of `Project_User_Assoc` objects, Set of `Project_Group_Assoc` objects

get_roles (*project*)

Get roles of the user in the given project.

Parameters **project** (`Project`) – concerning project

Returns list of `Role` objects

Raises `TypeError` if project is not instance of `Project`

Raises Exception if user not in project

get_workgroups ()

Get workgroups of the user.

Returns list of `Workgroup` objects

static guest_login (*session*)

has_permission (*permissions*, *project*)

Check if the user has the given permissions in the given project

Parameters

- **permission** (*Permission or list of Permission*) – permission or list of permissions to be checked
- **project** (`Project`) – concerning project

Returns bool is all permissions are present

Raises `TypeError` if project is not instance of `Project`

Raises Exception if user not in project

hash

id

static is_active (*session*, *uid*)

is_in_workgroup (*workgroup*)

Check if the user is in the workgroup, either user or manager.

Parameters **workgroup** (`Workgroup`) – concerning workgroup

Returns bool whether user is manager or user of workgroup

Raises `TypeError` if workgrop is not instance of `Workgroup`

last_password_reset

last_request

static login (*session*, *username*, *password*)

name

password

static register (*session*, *name*, *email*)

```

static send_registration_mail (name, email, hash)

serveradmin

static set_password (db, hash, password)

status

static update_last_request (session, uid)

```

vispa.models.workgroup module

```

class vispa.models.workgroup.Workgroup (**kwargs)
    Bases: sqlalchemy.ext.declarative.api.Base

```

A workgroup is a lighter version of a project. It only contains users, no groups, and there is no explicit management of roles and permission, which means, all users have the same rights. The memberships are organized by managers.

```
add_manager (user)
```

Add manager to workgroup.

Parameters **user** (`User`) – user which has to be added as manager

Raises `TypeError` if user is not instance of `User`

Raises `Exception` if user is already manager of workgroup

```
add_user (user)
```

Add user to workgroup.

Parameters **user** (`User`) – user which has to be added

Raises `TypeError` if user is not instance of `User`

Raises `Exception` if user is already in workgroup

```
static all (session)
```

Get all existing workgroups.

Parameters **session** – current session of database

Returns list of `Workgroup` objects

```
static create (session, name)
```

Create new workgroup.

Parameters

- **session** – current session of database
- **name** – name of the workgroup

Raises `Exception` if parameter name is invalid or already in use

```
created
```

```
delete (session)
```

Delete workgroup. It is also deleted from the database.

Parameters **session** – current session of database

```
static get (session, workgroup)
```

Returns workgroup, whichs name or id is given. If the parameter workgroup is already an instance of `Workroup` it is directly returned.

Parameters

- **session** – current session of database
- **workgroup** – name or id to look for. If workgroup is instance of Workgroup, it is directly returned

Returns Workgroup**Raises** Exception if parameter workgroup is invalid if no workgroup can be found**static** **get_by_id** (*session, gid*)

Get a workgroup by its id.

Parameters **session** – current session of database**Para** **gid** id, which is looked for**Returns** Workgroup or None**static** **get_by_name** (*session, name*)

Get a workgroup by its name.

Parameters **session** – current session of database**Para** **name** name, which is looked for**Returns** Workgroup or None**get_items** (*itemtype=None*)

Get WorkgroupItem.

Parameters **itemtype** – optional selector on the item type**Returns** list of WorkgroupItem objects**get_managers** ()

Get managers of workgroup.

Returns list of User objects**get_users** ()

Get users of the workgroup.

Returns list of User objects**id****items****managers****name****remove_manager** (*manager*)

Remove manager from workgroup.

Parameters **manager** (*User*) – manager which has to be removed**Raises** *TypeError* if manager is not instance of *User***Raises** Exception if manager is not manager of workgroup**remove_user** (*user*)

Remove user from workgroup.

Parameters **user** (*User*) – user which has to be removed**Raises** *TypeError* if user is not instance of *User*

Raises Exception if user is not in workgroup

rename (*session*, *newname*)

Rename workgroup.

Parameters

- **session** – current session of database
- **newname** – new name of the workgroup

Raises Exception if newname is already in use

users

class vispa.models.workgroup.**WorkgroupItem** (***kwargs*)

Bases: sqlalchemy.ext.declarative.api.Base

A workgroup item represents the actual content of a workgroup. It is simply an object consisting of a type (e.g. file) and its content (e.g. the path of the file)

content

static create (*session*, *workgroup*, *itemtype*, *content*)

Create an workgroup item. The item is added to the database.

Parameters

- **session** – current session of database
- **workgroup** – workgroup of the new item
- **itemtype** – type of the item
- **content** – content of the item

delete (*session*)

Delete the item. It is also deleted from the database.

Parameters **session** – current session of database

static get (*session*, *item*)

Returns item, whichs id given. If the parameter item is an instance of WorkgroupItem object, it is directly returned.

Parameters

- **session** – current session of database
- **item** – id to look for. If item is an instance of WorkgroupItem, item is directly returned

Returns WorkgroupItem

Raises Exception if parameter item is invalid or no WorkgroupItem can be found

static get_by_id (*session*, *gid*)

Get a workgroup item by its id.

Parameters

- **session** – current session of database
- **gid** – given id, which is looked for

Returns WorkgroupItem or None

get_workgroup ()

Get the workgroup of the item.

Returns Workgroup

id

itemtype

set_content (*content*)

Set the content of the item.

Parameters **content** – new content

workgroup_id

vispa.models.workspace module

```
class vispa.models.workspace.Workspace (**kwargs)
    Bases: sqlalchemy.ext.declarative.api.Base
    KEYS = ['id', 'user_id', 'name', 'host', 'login', 'command', 'created', 'auto_connect', 'key']
    static add (db, user, name, host, login, key=None, command=None, add=True)
    auto_connect
    can_edit (user)
    command
    created
    static get_by_id (db, id)
    static get_user_workspace_count (db, user)
    static get_user_workspaces (db, user)
    has_access (user)
    host
    id
    is_valid ()
    key
    login
    login_credentials
    make_dict (keys=None)
    name
    static remove (db, id)
    static update (db, id, **kwargs)
    user_id
```

Module contents

vispa.plugins package

Submodules

vispa.plugins.template module

```
class vispa.plugins.template.MakoPlugin (bus, base_dir=None, module_dir=None, collec-
                                     tion_size=50, encoding='utf-8')
    Bases: cherry.py.process.plugins.SimplePlugin
    lookup_template (name)
    start ()
    stop ()
```

Module contents

vispa.remote package

Subpackages

vispa.remote.fsmonitor package

Submodules

vispa.remote.fsmonitor.common module

```
class vispa.remote.fsmonitor.common.FSEvent (watch, action, name='')
    Bases: object
    Access = 1
    All = 511
    Attrib = 4
    Create = 8
    Delete = 16
    DeleteSelf = 32
    Modify = 2
    MoveFrom = 64
    MoveSelf = 256
    MoveTo = 128
    action_name
    action_names = {16: 'delete', 1: 'access', 2: 'modify', 4: 'attrib', 32: 'delete self', 8: 'create', 64: 'move from', 128: 'm
    path
```

user

exception `vispa.remote.fsmonitor.common.FSMonitorError`

Bases: `exceptions.Exception`

exception `vispa.remote.fsmonitor.common.FSMonitorOSError`

Bases: `exceptions.OSError`, `vispa.remote.fsmonitor.common.FSMonitorError`

vispa.remote.fsmonitor.linux module

class `vispa.remote.fsmonitor.linux.FSMonitor`

Bases: `object`

add_dir_watch (*path*, *flags*=511, *user*=None)

add_file_watch (*path*, *flags*=511, *user*=None)

close ()

disable_watch (*watch*)

enable_watch (*watch*, *enable*=True)

read_events (*timeout*=None)

remove_all_watches ()

remove_watch (*watch*)

watches

class `vispa.remote.fsmonitor.linux.FSMonitorWatch` (*wd*, *path*, *flags*, *user*)

Bases: `object`

`vispa.remote.fsmonitor.linux.convert_flags` (*flags*)

`vispa.remote.fsmonitor.linux.parse_events` (*s*)

vispa.remote.fsmonitor.polling module

class `vispa.remote.fsmonitor.polling.FSMonitor`

Bases: `object`

add_dir_watch (*path*, *flags*=511, *user*=None)

add_file_watch (*path*, *flags*=511, *user*=None)

disable_watch (*watch*)

enable_watch (*watch*, *enable*=True)

read_events (*timeout*=None)

remove_all_watches ()

remove_watch (*watch*)

watches

class `vispa.remote.fsmonitor.polling.FSMonitorDirWatch` (*path*, *flags*, *user*)

Bases: `object`

delstate ()


```

    getstate()
    classmethod new_state(path)
    setstate(state)
    state
class vispa.remote.fsmonitor.polling.FSMonitorFileWatch(path, flags, user)
    Bases: object
    delstate()
    getstate()
    classmethod new_state(path)
    setstate(state)
    state
class vispa.remote.fsmonitor.polling.FSMonitorWatch(path, flags, user)
    Bases: object
vispa.remote.fsmonitor.polling.get_dir_contents(path)
vispa.remote.fsmonitor.polling.round_fs_resolution(t)

```

vispa.remote.fsmonitor.win32 module

Module contents

```

class vispa.remote.fsmonitor.FSMonitor
    Bases: object
    add_dir_watch(path, flags=511, user=None)
    add_file_watch(path, flags=511, user=None)
    close()
    disable_watch(watch)
    enable_watch(watch, enable=True)
    read_events(timeout=None)
    remove_all_watches()
    remove_watch(watch)
    watches
class vispa.remote.fsmonitor.FSMonitorThread(callback=None)
    Bases: threading.Thread
    add_dir_watch(path, flags=511, user=None)
    add_file_watch(path, flags=511, user=None)
    read_events()
    remove_all_watches()
    remove_watch(watch)
    run()

```

```
    stop()

exception vispa.remote.fsmonitor.FSMonitorError
    Bases: exceptions.Exception

exception vispa.remote.fsmonitor.FSMonitorOSError
    Bases: exceptions.OSError, vispa.remote.fsmonitor.common.FSMonitorError

class vispa.remote.fsmonitor.FSEvent (watch, action, name='')
    Bases: object

    Access = 1
    All = 511
    Attrib = 4
    Create = 8
    Delete = 16
    DeleteSelf = 32
    Modify = 2
    MoveFrom = 64
    MoveSelf = 256
    MoveTo = 128
    action_name
    action_names = {16: 'delete', 1: 'access', 2: 'modify', 4: 'attrib', 32: 'delete self', 8: 'create', 64: 'move from', 128: 'move to'}
    path
    user
```

Submodules

vispa.remote.filesystem module

```
class vispa.remote.filesystem.FileSystem (userid, workspaceid)
    Bases: object

    ADDITIONAL_MIMES = {'root': 'text/plain', 'pxlio': 'text/plain'}
    BROWSER_EXTENSIONS = ['png', 'jpg', 'jpeg', 'bmp']
    FILE_EXTENSIONS = ['png', 'jpg', 'jpeg', 'bmp', 'ps', 'eps', 'pdf', 'txt', 'xml', 'py', 'c', 'cpp', 'root', 'pxlio']
    GLOBAL_WORKSPACE_CONF = '/etc/vispa/workspace.ini'
    PRIVATE_WORKSPACE_CONF = '~/vispa/workspace.ini'

    checkPermissions (path, permission=2)
    check_file_extension (path, extensions=[])
    close()
    compress (paths, path, name, is_tmp=False)
    create_file (path, name)
    create_folder (path, name)
```

```

cut_slashes (path)
decompress (path)
exists (path, type=None)
expand (path)
get_file (path, binary=False, utf8=False, window_id=None, view_id=None, watch_id=None,
           max_size=20)
get_file_content (path, offset=0, length=None)
get_file_count (path, window_id=None, view_id=None, watch_id=None)
get_file_list (path, filter=None, reverse=False, hide_hidden=True, encode_json=True, window_id=None, view_id=None, watch_id=None)
get_mime_type (filepath)
get_mtime (path)
get_suggestions (path, length=1, append_hidden=True, encode_json=True)
get_workspaceini (request, fail_on_missing=False)
getfacl (path)
handle_file_name_collision (name, path)
is_browser_file (path)
move (source, destination)
paste (path, fullsrc, cut)
remove (path)
rename (path, name, new_name, force=False)
save_file (path, content, force=True, binary=False, utf8=False, window_id=None, view_id=None,
           watch_id=None)
save_file_content (filename, content, path=None, force=True, append=False)
set_workspaceini (request)
setfacl (path, type, name, mode, remove=False, recursive=False, default=False)
setup (basedir=None)
stat (path)
thumbnail (path, width=100, height=100, sharpen=True)
unwatch (window_id, view_id, watch_id=None)
watch (path, window_id, view_id, watch_id, pattern=None, reverse=False, hide_hidden=True)

class vispa.remote.filesystem.WatchService
    Bases: object

    stop ()

    subscribe (id, path, pattern=None, reverse=False, hide_hidden=True)

    unsubscribe (id)

class vispa.remote.filesystem.WatchSubscriber (service, id)
    Bases: object

```

```
EVENT_DELAYS = {'modify': [1.0, 0.2], 'change': [1.0, 0.1]}
MAX_INLINE_SUBJECTS = 10
MAX_SUBJECT_NAMES = 25

bind(path)

destroy()

emit(event)

flush(force=False)

process(event, subject='')

unbind()

update(path, pattern='', reverse=False, hide_hidden=True)

vispa.remote.filesystem.file_compare(a, b)

vispa.remote.filesystem.get_file_info(base, name)

vispa.remote.filesystem.string_compare(a, b)
```

vispa.remote.helper module

```
class vispa.remote.helper.UTF8Buffer
```

Bases: object

Buffers incoming UTF8 encoded data, and prevents chunks from being created in the middle of a multi byte sequence.

```
fill(data)
```

Fill the buffer with the given *data*.

```
passThru(data, count=None)
```

A convenience function for adding data to the buffer and reading up to *count* of it again. If *count* is *None* the current buffer length is used.

```
read(count=None)
```

Get a maximum of *count* bytes from the buffer. UTF8 multibyte characters will not be broken apart. If *count* is *None* the current buffer length is used.

Module contents

```
exception vispa.remote.AjaxException(message, code=None, alert=True)
```

Bases: exceptions.Exception

AjaxException that is handled by the ajax tool and that can be raised in controller methods. *message* is the error message to show. *code* should be an integer that represents a specific type of exception. If *code* is *None* and *message* is an integer representing a http status code, the error message is set to the standard error message for that http error. If *alert* is *True*, the message is shown in a dialog in the GUI.

```
vispa.remote.raise_ajax(fn=None, **kwargs)
```

Decorator that transforms raised exceptions into AjaxExceptions.

```
vispa.remote.send_topic(topic, data=None, window_id=None, user_id=None)
```

vispa.tools package

Submodules

vispa.tools.ajax module

Definition of the vispa ajax tool.

```
class vispa.tools.ajax.AjaxTool
    Bases: cherrypy._cptools.Tool
```

Ajax tool that takes the output of a wrapper inner function and returns a json encoded dictionary containing the following entries: - code: A response code, basically an appropriate http status. The status of the

cherrypy response object is also set to this value. Thus, the code is 200 if there were no errors.

- data: The attached payload with an arbitrary type.
- message: In case of an error, i.e. code != 200, an additional error message. On success, i.e. code = 200, this should be empty.
- alert: A boolean that describes whether the error message should be shown in the GUI using a dialog or something similar.

callable (*encoded=False*)

The callable of this tool. If *encoded* is *True*, the return value of the wrapped inner function is expected to be already json encoded. Otherwise, it will be encoded using `json.dumps()`.

vispa.tools.db module

```
class vispa.tools.db.SqlAlchemyTool (engine)
    Bases: cherrypy._cptools.Tool
```

bind_session ()

Attaches a session to the request's scope by requesting the SA plugin to bind a session to the SA engine.

commit_transaction ()

Commits the current transaction or rolls back if an error occurs. Removes the session handle from the request's scope.

vispa.tools.device module

```
class vispa.tools.device.DeviceTool
    Bases: cherrypy._cptools.Tool
```

devicematch (*agent, device='all'*)

get_device_name (*agent*)

groupmatch (*agent, group='all'*)

vispa.tools.json_parameters module

```
class vispa.tools.json_parameters.JsonParameters
    Bases: cherrypy._cptools.Tool
```

```
before_handler()
```

vispa.tools.method module

Definition of the vispa method tool.

```
class vispa.tools.method.MethodTool
```

```
    Bases: cherrypy._cptools.Tool
```

Basically, the method tool implements the same functionality as cherrypy's built-in "allow" tool, but in addition, this tool is compliant to our ajax tool.

```
    callable (accept=None, reject=None, ajax=True)
```

Actual tool logic. Checks whether a request is sent with a valid HTTP method using *accept* and *reject*. Both of them can be strings or iterables of strings. When the request not accepted and *ajax* is *True*, the ajax tool is used to send an error. Otherwise, a cherrypy.HTTPError is raised.

vispa.tools.parameters module

Cherrypy tool that moves parameters that start with an underscore from the querystring to request.private_params.

```
class vispa.tools.parameters.PrivateParameterFilter
```

```
    Bases: cherrypy._cptools.Tool
```

```
    before_handler()
```

vispa.tools.permission module

```
class vispa.tools.permission.PermissionTool
```

```
    Bases: cherrypy._cptools.Tool
```

use from config file:

```
[/path/to/protected/resource] tools.permission.on = True tools.permission.requiredPermissions = ['myextension.read', 'myextension.write']
```

use as decorator: @cherrypy.expose @cherrypy.tools.permission(requiredPermissions=['myextension.resources'])
def resource(self):

```
    return "Hello, %s!" % cherrypy.request.login
```

Parameters

- **requiredPermissions** – string or iterable of strings, which the user must all have in the global project. If the user management is not automatically setup, the tool does nothing
- **ignoreInexistentPermissions** – when true, requiredPermissions are ignored if they do not exist

vispa.tools.status module

```
class vispa.tools.status.StatusMonitor
```

```
    Bases: cherrypy._cptools.Tool
```

Register the status of each thread.

```

callable()
unregister()
    Unregister the current thread.

```

```

class vispa.tools.status.ThreadStatus (threadid)
    Bases: object

    end = None
    idle_time()
    last_req_time()
    start = None
    url = None

```

vispa.tools.template module

```

class vispa.tools.template.MakoTool (common_data=None)
    Bases: cherrypy._cptools.Tool

    callable (template=None, common_data=None, **kwargs)

```

vispa.tools.user module

Definition of the vispa user tool.

```

class vispa.tools.user.UserTool
    Bases: cherrypy._cptools.Tool

```

The user tool checks whether the session contains the field “user_id”. If it exists, a reference to the corresponding user is stored as “cherrypy.request.user”. Otherwise, the request is either redirected or a 401 error is returned using the ajax tool.

vispa.tools.workspace module

```

class vispa.tools.workspace.WorkspaceTool
    Bases: cherrypy._cptools.Tool

    before_handler (**conf)

```

Module contents

Submodules

vispa.browser module

```

vispa.browser.append_to_session (key, value)
    Adds a value to a list in the session

vispa.browser.client_agent()

vispa.browser.client_ip()

vispa.browser.client_referer()

```

```
vispa.browser.delete_cookie(key)
vispa.browser.delete_session()
vispa.browser.get_cookie(key)
vispa.browser.get_session_value(key)
vispa.browser.has_session_value(key)
vispa.browser.set_cookie(key, value='', age=None, path='/', version=None)
vispa.browser.set_session_value(key, value=None)
vispa.browser.update_to_session(key, data)
```

vispa.rest module

Dispatcher and controller for implementing a REST api in cherrypy.

```
class vispa.rest.RESTDispatcher(dispatch_method_name=None,                                trans-
                                late='x00x01x02x03x04x05x06x07x08tnx0bx0crx0ex0fx10x11x12x13x14x15x16x17x18x19
                                0123456789_____ABCDEFGHI-
                                JKLMNOPQRSTUVWXYZ_____abcdefghijklmnopqrstuvwxyz____x7fx80x81x82x83x84x
Bases: cherrypy._cpdispatch.Dispatcher
```

Dispatches request to handler functions that are decorated with resolvers. Use it in conjunction with the REST-Controller class. Example:

```
class Controller(RESTController):

    @POST("user/:id(^\\d+$/name/:name")
    def set_user_name_by_id(self, id, new_name):
        pass

    @POST("user/:name/name/:name")
    def set_user_name_by_name(self, name, new_name):
        pass
```

Calls to POST /user/4/name/tom will be routed to the first handler, calls to POST /user/tom/name/tim to the second one. Thus, the order of handler definitions is important for the dispatcher to take precedence rules into account.

find_handler (*path*)

Implements cherrypy.dispatch.Dispatcher.find_handler(path).

find_node (*path*)

Trails the handler tree to find the correct RESTController instance.

```
class vispa.rest.RESTController
```

Bases: object

RESTController that should be used in conjunction the a RESTDispatcher. It's only task is to prepare handlers at the end of its initialization.

```
vispa.rest.GET(fmt)
```

```
vispa.rest.POST(fmt)
```

```
vispa.rest.DELETE(fmt)
```


vispa.server module

```

class vispa.server.AbstractExtension (server)
    Bases: object

    Base class for Extensions

    add_controller (controller)
        Mount a CherryPy controller using the extension name for path.

        Parameters controller – filename relative to extension directory

    add_workspace_directory (directory='workspace')
        Add files to be transferred to the worker.

        Parameters directory – directory relative to extension directory

    clear_workspace_instance (name, key=None, user=None, workspace=None, db=None)

    config ()

    create_topic (topic='', view_id=None)

    dependencies ()
        Return a list of Extension names this Extension depends on.

    get_workspace_instance (name, key=None, user=None, workspace=None, db=None, **kwargs)

    name ()
        Return the name of the Extension. This name is used as part of the URL.

    setup ()
        Setup the extension.

class vispa.server.Server (**kwargs)
    Bases: object

    application ()

    extension (name)

    extensions ()

    run ()

    start ()

```

vispa.socketbus module

```

class vispa.socketbus.Bus

    received_message (msg, window_id)

    send (window_id=None, user_id=None, data=None, except_sessions=None, binary=False, encode_json=True, broadcast=False)

    send_topic (topic, *args, **kwargs)

    subscribe (topic, handler)

    unsubscribe (topic, handler=None)

class vispa.socketbus.CleanerThread (*kargs, **kwargs)
    Bases: threading.Thread

```

```
run ()

class vispa.socketbus.PollingPublisher (window_id, user_id)

    fetch (timeout=None)
    received_message (msg)
    send (data, binary=False, timeout=None)

class vispa.socketbus.SocketPublisher
vispa.socketbus.add_session (window_id, user_id, publisher)
vispa.socketbus.get_polling_publisher (window_id, user_id)
vispa.socketbus.remove_session (window_id, delay=False)
```

vispa.url module

Functions to translate a relative path into a valid url

```
vispa.url.clean (url)
    Clean the url, multiple fix slashes and strip whitespaces

vispa.url.dynamic (*parts, **kwargs)
    Create an absolute URL to non static content, e.g. controllers

vispa.url.join (*args)
    Join all parameters into one clean url path.

vispa.url.static (*parts, **kwargs)
    Create an absolute URL to static content, e.g. images
```

vispa.version module

vispa.workspace module

Functions to connect and control connections to workers

```
class vispa.workspace.Connection (userid, workspaceid, host, **kwargs)
    Bases: object

    CONNECTED = 2
    CONNECTING = 1
    DISCONNECTED = 0
    DISCONNECTING = 3
    STATUS_MESSAGES = {0: 'disconnected', 1: 'connecting', 2: 'connected', 3: 'disconnecting'}

    active ()
    close ()
    connected (timeout=10)
    errors ()
    host ()
```

```

    open (**kwargs)
    poll ()
    rpyc ()
    send_status ()
    static send_workspace_status (userid, workspaceid, status)
    status (new_status=None)
    stdin ()
    stdout ()
    tempdir ()
class vispa.workspace.ConnectionPool
    Bases: object
    SERVE_CONNECTION_INTERVAL = 0.2
    clear (user=None, workspace=None)
    connect (user, workspace, password=None)
    connections_of_user (user)
        Returns a list of tuples: (workspaceId, connection)
    get (user, workspace, **kwargs)
    get_workspace_connections (user, workspaces)
        return a list of tuples with the workspace and the associated connection or None
class vispa.workspace.InstancePool
    Bases: object
    clear (user=None, workspace=None, classname=None, key=None)
    get (_user, _workspace, classname=None, key=None, init_args=None, **kwargs)
class vispa.workspace.LocalConnectionFeeder
    Bases: threading.Thread
    callbacks = {}
    run ()
class vispa.workspace.LocalConnectionImpl (command, **kwargs)
    Bases: object
    close ()
    print_stderr (timeout=0)
    set_on_feed (cb)
    stream ()
    writeln (data)
class vispa.workspace.LoggingService (conn)
    Bases: rpyc.core.service.Service
    exposed_log (name, level, msg)
    exposed_send (*args, **kwargs)

```

```
class vispa.workspace.SSHConnectionImpl (command, address, miss-
                                     ing_host_key_policy='warning', **kwargs)
    Bases: object
    An SSH transport for Pushy, which uses Paramiko.
    TIMEOUT = 30
    address2host_port (host)
    close ()
    print_stderr ()
    set_on_feed (cb)
    stream ()
    writeln (data)
class vispa.workspace.WrappedChannelFile (file_, how)
    Bases: object
    close ()
    fileno ()
vispa.workspace.add_directory_files (local, remote, **kwargs)
vispa.workspace.add_package_files (pkg, target=None)
vispa.workspace.add_remote_file (localfile, remotefile)
vispa.workspace.add_remote_files (files)
vispa.workspace.clear_instance (classname, key=None, user=None, workspace=None,
                                db=None)
    Remove all instances of type classname from the pool
vispa.workspace.connect (workspace, user=None, db=None, password=None)
    Connect the selected workspace.
vispa.workspace.directory_files (local, remote, **kwargs)
vispa.workspace.disconnect (workspace=None, user=None, db=None)
    Disconnect the selected workspace and remove all pooled instances.
vispa.workspace.disconnect_all (user=None, db=None)
    Disconnect all workspaces and pooled instances of the selected user.
vispa.workspace.get_instance (classname, key=None, user=None, workspace=None, db=None,
                              init_args=None)
    Returns a pooled reference to a class instance of type classname.
vispa.workspace.module (modulename, user=None, workspace=None, db=None)
    Returns a reference to a remote module
vispa.workspace.package_files (pkg, target_path, **kwargs)
```

vispa.wsgi module

```
vispa.wsgi.application (environ, start_response)
```

Module contents

Basic functionality for the VISPA platform

exception `vispa.AjaxException` (*message*, *code=None*, *alert=True*)

Bases: `exceptions.Exception`

AjaxException that is handled by the ajax tool and that can be raised in controller methods. *message* is the error message to show. *code* should be an integer that represents a specific type of exception. If *code* is *None* and *message* is an integer representing a http status code, the error message is set to the standard error message for that http error. If *alert* is *True*, the message is shown in a dialog in the GUI.

class `vispa.Netstat`

Bases: `object`

Parses /dev/net/tcp to determine which user owns the specified port

static `get_socket_owner` (*local_ip*, *local_port*, *remote_ip*, *remote_port*)

Returns the system id of the local user which established the specifiec connection

Parameters

- **local_ip** – ip address of the local peer
- **local_ip** – port the local peer
- **remote_ip** – ip address of the remote peer
- **remote_ip** – port the remote peer

Return type user id of the user who opened this port or None

class `vispa.VispaConfigParser` (*defaults=None*, *dict_type=<class 'collections.OrderedDict'>*, *allow_no_value=False*)

Bases: `ConfigParser.SafeConfigParser`

`vispa.codepath` (**args*)

`vispa.configpath` (**args*)

`vispa.datapath` (**args*)

returns the path relative to the datapath

`vispa.dump_thread_status` (*f=None*)

`vispa.dump_thread_status_on_signal` (*signal*, *stack*)

`vispa.exception_string` ()

`vispa.fire_callback` (*topic*, **args*, ***kwargs*)

`vispa.log_exception` ()

`vispa.publish` (*topic*, **args*, ***kwargs*)

`vispa.register_callback` (*topic*, *callback*)

`vispa.send_mail` (*addr*, *subject=''*, *content=''*, *sender_addr=None*, *smtp_host=None*, *smtp_port=None*)

Send an email.

All arguments should be Unicode strings (plain ASCII works as well).

Only the real name part of sender and recipient addresses may contain non-ASCII characters.

The email will be properly MIME encoded and delivered though SMTP to localhost port 25. This is easy to change if you want something different.

The charset of the email will be the first one out of US-ASCII, ISO-8859-1 and UTF-8 that can represent all the characters occurring in the email.

`vispa.set_codepath(p)`

`vispa.set_configpath(p)`

`vispa.set_datapath(p)`

`vispa.setup_thread_dump()`

`vispa.subscribe(topic, callback)`

`vispa.thread_stacktraces()`

V

- `vispa`, 121
- `vispa.browser`, 115
- `vispa.controller`, 82
- `vispa.controller.ajax`, 69
- `vispa.controller.bus`, 70
- `vispa.controller.error`, 70
- `vispa.controller.filesystem`, 70
- `vispa.controller.root`, 72
- `vispa.controller.usermanagement`, 72
- `vispa.extensions`, 87
- `vispa.extensions.codeeditor`, 83
- `vispa.extensions.codeeditor.controller`, 83
- `vispa.extensions.codeeditor.workspace`, 82
- `vispa.extensions.core`, 83
- `vispa.extensions.demo`, 84
- `vispa.extensions.demo.controller`, 84
- `vispa.extensions.demo.workspace`, 84
- `vispa.extensions.dummy`, 85
- `vispa.extensions.dummy.controller`, 85
- `vispa.extensions.dummy.workspace`, 84
- `vispa.extensions.file`, 85
- `vispa.extensions.file.controller`, 85
- `vispa.extensions.gallery`, 85
- `vispa.extensions.terminal`, 86
- `vispa.extensions.terminal.workspace`, 86
- `vispa.models`, 107
- `vispa.models.alembic`, 87
- `vispa.models.group`, 87
- `vispa.models.jsondata`, 91
- `vispa.models.project`, 92
- `vispa.models.role`, 98
- `vispa.models.user`, 101
- `vispa.models.workgroup`, 103
- `vispa.models.workspace`, 106
- `vispa.plugins`, 107
- `vispa.plugins.template`, 107
- `vispa.remote`, 112
- `vispa.remote.filesystem`, 110
- `vispa.remote.fsmonitor`, 109
- `vispa.remote.fsmonitor.common`, 107
- `vispa.remote.fsmonitor.linux`, 108
- `vispa.remote.fsmonitor.polling`, 108
- `vispa.remote.helper`, 112
- `vispa.rest`, 116
- `vispa.server`, 117
- `vispa.socketbus`, 117
- `vispa.tools`, 115
- `vispa.tools.ajax`, 113
- `vispa.tools.db`, 113
- `vispa.tools.device`, 113
- `vispa.tools.json_parameters`, 113
- `vispa.tools.method`, 114
- `vispa.tools.parameters`, 114
- `vispa.tools.permission`, 114
- `vispa.tools.status`, 114
- `vispa.tools.template`, 115
- `vispa.tools.user`, 115
- `vispa.tools.workspace`, 115
- `vispa.url`, 118
- `vispa.version`, 118
- `vispa.workspace`, 118
- `vispa.wsgi`, 120

Symbols

`()`, 44

A

`abort()` (vispa.extensions.codeeditor.controller.EditorController method), 83

`abort()` (vispa.extensions.codeeditor.workspace.CodeEditorRpc method), 82

`AbstractController` (class in vispa.controller), 11, 82

`AbstractExtension` (class in vispa.server), 8, 117

`Access` (vispa.remote.fsmonitor.common.FSEvent attribute), 107

`Access` (vispa.remote.fsmonitor.FSEvent attribute), 110

`action_name` (vispa.remote.fsmonitor.common.FSEvent attribute), 107

`action_name` (vispa.remote.fsmonitor.FSEvent attribute), 110

`action_names` (vispa.remote.fsmonitor.common.FSEvent attribute), 107

`action_names` (vispa.remote.fsmonitor.FSEvent attribute), 110

`ACTIVE` (vispa.models.group.Group attribute), 88

`ACTIVE` (vispa.models.project.Project attribute), 92

`ACTIVE` (vispa.models.user.User attribute), 101

`active()` (vispa.models.user.User method), 101

`active()` (vispa.workspace.Connection method), 118

`add()` (vispa.models.workspace.Workspace static method), 106

`add_child_group()` (vispa.models.group.Group method), 88

`add_controller()` (vispa.server.AbstractExtension method), 8, 117

`add_dir_watch()` (vispa.remote.fsmonitor.FSMonitor method), 109

`add_dir_watch()` (vispa.remote.fsmonitor.FSMonitorThread method), 109

`add_dir_watch()` (vispa.remote.fsmonitor.linux.FSMonitor method), 108

`add_dir_watch()` (vispa.remote.fsmonitor.polling.FSMonitor method), 108

`add_directory_files()` (in module vispa.workspace), 120

`add_file_watch()` (vispa.remote.fsmonitor.FSMonitor method), 109

`add_file_watch()` (vispa.remote.fsmonitor.FSMonitorThread method), 109

`add_file_watch()` (vispa.remote.fsmonitor.linux.FSMonitor method), 108

`add_file_watch()` (vispa.remote.fsmonitor.polling.FSMonitor method), 108

`add_group()` (vispa.models.project.Project method), 93

`add_manager()` (vispa.models.group.Group method), 88

`add_manager()` (vispa.models.project.Project method), 93

`add_manager()` (vispa.models.workgroup.Workgroup method), 103

`add_package_files()` (in module vispa.workspace), 120

`add_permissions()` (vispa.models.role.Role method), 99

`add_remote_file()` (in module vispa.workspace), 120

`add_remote_files()` (in module vispa.workspace), 120

`add_roles_to_group()` (vispa.models.project.Project method), 93

`add_roles_to_user()` (vispa.models.project.Project method), 93

`add_session()` (in module vispa.socketbus), 118

`add_user()` (vispa.models.group.Group method), 88

`add_user()` (vispa.models.project.Project method), 93

`add_user()` (vispa.models.workgroup.Workgroup method), 103

`add_workspace_directoy()` (vispa.server.AbstractExtension method), 8, 117

`addFileHandler()`, 60

`addFileHandlers()`, 60

`ADDITIONAL_MIMES`

(vispa.remote.filesystem.FileSystem attribute), 110

`address2host_port()` (vispa.workspace.SSHConnectionImpl method), 120

`addView()`, 44

`addViews()`, 44

`addworkspace()` (vispa.controller.ajax.AjaxController method), 69

ajax(), 56
 AjaxController (class in vispa.controller.ajax), 69
 AjaxException, 112, 121
 AjaxMixin() (mixin), 55
 AjaxTool (class in vispa.tools.ajax), 113
 alert(), 64
 All (vispa.remote.fsmonitor.common.FSEvent attribute), 107
 All (vispa.remote.fsmonitor.FSEvent attribute), 110
 all() (vispa.models.group.Group static method), 88
 all() (vispa.models.project.Project static method), 93
 all() (vispa.models.role.Permission static method), 98
 all() (vispa.models.role.Role static method), 99
 all() (vispa.models.user.User static method), 101
 all() (vispa.models.workgroup.Workgroup static method), 103
 append_to_session() (in module vispa.browser), 115
 application() (in module vispa.wsgi), 120
 application() (vispa.server.Server method), 117
 applyFragment(), 68
 applyPreferences(), 54
 applyShortcuts(), 57
 asyncRequire(), 43
 attach(), 45, 46, 48, 50
 Attrib (vispa.remote.fsmonitor.common.FSEvent attribute), 107
 Attrib (vispa.remote.fsmonitor.FSEvent attribute), 110
 auto_connect (vispa.models.workspace.Workspace attribute), 106

B

BaseView() (class), 62
 before_handler() (vispa.tools.json_parameters.JsonParameters method), 113
 before_handler() (vispa.tools.parameters.PrivateParameters method), 114
 before_handler() (vispa.tools.workspace.WorkspaceTool method), 115
 bind() (vispa.remote.filesystem.WatchSubscriber method), 112
 bind_session() (vispa.tools.db.SqlAlchemyTool method), 113
 blur(), 64
 BROWSER_EXTENSIONS (vispa.remote.filesystem.FileSystem attribute), 110
 BURST_BUFFER (vispa.extensions.codeeditor.workspace.CodeEditorRpc attribute), 82
 BURST_DELAY (vispa.extensions.codeeditor.workspace.CodeEditorRpc attribute), 82
 Bus (class in vispa.socketbus), 117
 BusController (class in vispa.controller.bus), 70
 byId(), 31, 51
 byName(), 31, 44

C

cache() (vispa.controller.AbstractController method), 11, 82
 callable() (vispa.tools.ajax.AjaxTool method), 113
 callable() (vispa.tools.method.MethodTool method), 114
 callable() (vispa.tools.status.StatusMonitor method), 114
 callable() (vispa.tools.template.MakoTool method), 115
 callback(), 42
 callbacks (vispa.workspace.LocalConnectionFeeder attribute), 119
 can_edit() (vispa.models.workspace.Workspace method), 106
 capitalize(), 42
 check_file_extension() (vispa.remote.filesystem.FileSystem method), 110
 checkpermissions() (vispa.controller.filesystem.FSAjaxController method), 70
 checkPermissions() (vispa.remote.filesystem.FileSystem method), 110
 child_group (vispa.models.group.Group_Group_Assoc attribute), 87
 child_group_id (vispa.models.group.Group_Group_Assoc attribute), 87
 child_groups (vispa.models.group.Group attribute), 89
 clean() (in module vispa.url), 118
 CleanerThread (class in vispa.socketbus), 117
 clear() (vispa.workspace.ConnectionPool method), 119
 clear() (vispa.workspace.InstancePool method), 119
 clear_instance() (in module vispa.workspace), 120
 clear_workspace_instance() (vispa.server.AbstractExtension method), 8, 117
 client_agent() (in module vispa.browser), 115
 client_ip() (in module vispa.browser), 115
 client_referer() (in module vispa.browser), 115
 close(), 64
 close() (vispa.extensions.codeeditor.controller.EditorController method), 83
 close() (vispa.extensions.codeeditor.workspace.CodeEditorRpc method), 83
 close() (vispa.extensions.terminal.TerminalController method), 86
 close() (vispa.extensions.terminal.workspace.Terminal method), 86
 close() (vispa.remote.filesystem.FileSystem method), 110
 close() (vispa.remote.fsmonitor.FSMonitor method), 109
 close() (vispa.remote.fsmonitor.linux.FSMonitor method), 108
 close() (vispa.workspace.Connection method), 118
 close() (vispa.workspace.LocalConnectionImpl method), 119
 close() (vispa.workspace.SSHConnectionImpl method), 120

- close() (vispa.workspace.WrappedChannelFile method), 120
- closeAllInstances(), 31, 44, 66
- CodeEditorExtension (class in vispa.extensions.codeeditor), 83
- CodeEditorRpc (class in vispa.extensions.codeeditor.workspace), 82
- codepath() (in module vispa), 121
- Color() (class), 37
- command (vispa.models.workspace.Workspace attribute), 106
- commit_transaction() (vispa.tools.db.SqlAlchemyTool method), 113
- communicate() (vispa.extensions.terminal.TerminalController method), 86
- communicate() (vispa.extensions.terminal.workspace.TerminalController method), 86
- compProxy(), 34
- compress() (vispa.controller.filesystem.FSAjaxController method), 70
- compress() (vispa.remote.filesystem.FileSystem method), 110
- config() (vispa.server.AbstractExtension method), 8, 117
- configpath() (in module vispa), 121
- confirm(), 64
- confirm_child_group() (vispa.models.group.Group method), 89
- confirm_user() (vispa.models.group.Group method), 89
- CONFIRMED (vispa.models.group.Group_Group_Assoc attribute), 87
- CONFIRMED (vispa.models.group.Group_User_Assoc attribute), 87
- connect(), 30
- connect() (in module vispa.workspace), 120
- connect() (vispa.workspace.ConnectionPool method), 119
- CONNECTED (vispa.workspace.Connection attribute), 118
- connected() (vispa.workspace.Connection method), 118
- CONNECTING (vispa.workspace.Connection attribute), 118
- Connection (class in vispa.workspace), 118
- ConnectionPool (class in vispa.workspace), 119
- connections_of_user() (vispa.workspace.ConnectionPool method), 119
- connectworkspace() (vispa.controller.ajax.AjaxController method), 69
- content (vispa.models.project.ProjectItem attribute), 97
- content (vispa.models.workgroup.WorkgroupItem attribute), 105
- convert() (vispa.controller.AbstractController method), 11, 82
- convert_flags() (in module vispa.remote.fsmonitor.linux), 108
- CoreController (class in vispa.extensions.core), 83
- CoreExtension (class in vispa.extensions.core), 83
- Create (vispa.remote.fsmonitor.common.FSEvent attribute), 107
- Create (vispa.remote.fsmonitor.FSEvent attribute), 110
- create() (vispa.models.project.Project static method), 94
- create() (vispa.models.project.ProjectItem static method), 97
- create() (vispa.models.role.Permission static method), 98
- create() (vispa.models.role.Role static method), 99
- create() (vispa.models.workgroup.Workgroup static method), 103
- create() (vispa.models.workgroup.WorkgroupItem static method), 105
- create_file() (vispa.remote.filesystem.FileSystem method), 110
- create_folder() (vispa.remote.filesystem.FileSystem method), 110
- create_topic() (vispa.server.AbstractExtension method), 8, 117
- created (vispa.models.group.Group attribute), 89
- created (vispa.models.project.Project attribute), 94
- created (vispa.models.role.Permission attribute), 98
- created (vispa.models.role.Role attribute), 99
- created (vispa.models.user.User attribute), 101
- created (vispa.models.workgroup.Workgroup attribute), 103
- created (vispa.models.workspace.Workspace attribute), 106
- createfile() (vispa.controller.filesystem.FSAjaxController method), 70
- createfolder() (vispa.controller.filesystem.FSAjaxController method), 70
- createNew(), 31
- createSubLogger(), 57
- cut_slashes() (vispa.remote.filesystem.FileSystem method), 111
- ## D
- data() (vispa.extensions.dummy.controller.DummyController method), 85
- datapath() (in module vispa), 121
- decompress() (vispa.controller.filesystem.FSAjaxController method), 71
- decompress() (vispa.remote.filesystem.FileSystem method), 111
- del(), 34
- Delete (vispa.remote.fsmonitor.common.FSEvent attribute), 107
- Delete (vispa.remote.fsmonitor.FSEvent attribute), 110
- DELETE(), 56
- DELETE() (in module vispa.rest), 116

- delete() (vispa.models.group.Group method), 89
- delete() (vispa.models.project.Project method), 94
- delete() (vispa.models.project.ProjectItem method), 97
- delete() (vispa.models.role.Permission method), 98
- delete() (vispa.models.role.Role method), 99
- delete() (vispa.models.workgroup.Workgroup method), 103
- delete() (vispa.models.workgroup.WorkgroupItem method), 105
- delete_cookie() (in module vispa.browser), 115
- delete_session() (in module vispa.browser), 116
- DELETED (vispa.models.group.Group attribute), 88
- DELETED (vispa.models.project.Project attribute), 93
- DeleteSelf (vispa.remote.fsmonitor.common.FSEvent attribute), 107
- DeleteSelf (vispa.remote.fsmonitor.FSEvent attribute), 110
- deleteworkspace() (vispa.controller.ajax.AjaxController method), 70
- delstate() (vispa.remote.fsmonitor.polling.FSMonitorDirWatch method), 108
- delstate() (vispa.remote.fsmonitor.polling.FSMonitorFileWatch method), 109
- DemoController (class in vispa.extensions.demo.controller), 84
- DemoExtension (class in vispa.extensions.demo), 84
- DemoRpc (class in vispa.extensions.demo.workspace), 84
- dependencies() (vispa.extensions.codeeditor.CodeEditorExtension method), 83
- dependencies() (vispa.extensions.core.CoreExtension method), 83
- dependencies() (vispa.extensions.demo.DemoExtension method), 84
- dependencies() (vispa.extensions.dummy.DummyExtension method), 85
- dependencies() (vispa.extensions.file.FileBrowserExtension method), 85
- dependencies() (vispa.extensions.gallery.GalleryExtension method), 86
- dependencies() (vispa.extensions.terminal.TerminalExtension method), 86
- dependencies() (vispa.server.AbstractExtension method), 8, 117
- destroy(), 51, 64
- destroy() (vispa.remote.filesystem.WatchSubscriber method), 112
- detach(), 45, 47, 48, 51
- detachAll(), 47
- devicematch() (vispa.tools.device.DeviceTool method), 113
- DeviceTool (class in vispa.tools.device), 113
- dialog(), 64
- Dialog() (class), 38
- Dialog.alert(), 40
- Dialog.confirm(), 41
- Dialog.dialog(), 40
- Dialog.prompt(), 41
- DialogView() (class), 61
- directory_files() (in module vispa.workspace), 120
- disable_watch() (vispa.remote.fsmonitor.FSMonitor method), 109
- disable_watch() (vispa.remote.fsmonitor.linux.FSMonitor method), 108
- disable_watch() (vispa.remote.fsmonitor.polling.FSMonitor method), 108
- disconnect(), 30
- disconnect() (in module vispa.workspace), 120
- disconnect_all() (in module vispa.workspace), 120
- DISCONNECTED (vispa.workspace.Connection attribute), 118
- DISCONNECTING (vispa.workspace.Connection attribute), 118
- disconnectworkspace() (vispa.controller.ajax.AjaxController method), 70
- dummy() (vispa.extensions.dummy.workspace.DummyRpc method), 84
- DummyController (class in vispa.extensions.dummy.controller), 85
- DummyExtension (class in vispa.extensions.dummy), 85
- DummyRpc (class in vispa.extensions.dummy.workspace), 84
- enable_thread_status() (in module vispa), 121
- dump_thread_status_on_signal() (in module vispa), 121
- dynamic() (in module vispa.url), 118
- dynamicURL(), 56
- E**
- edit(), 30
- EditorController (class in vispa.extensions.codeeditor.controller), 83
- editworkspace() (vispa.controller.ajax.AjaxController method), 70
- email (vispa.models.user.User attribute), 101
- emit(), 58
- emit() (vispa.remote.filesystem.WatchSubscriber method), 112
- enable_watch() (vispa.remote.fsmonitor.FSMonitor method), 109
- enable_watch() (vispa.remote.fsmonitor.linux.FSMonitor method), 108
- enable_watch() (vispa.remote.fsmonitor.polling.FSMonitor method), 108
- end (vispa.tools.status.ThreadStatus attribute), 115
- ErrorController (class in vispa.controller.error), 70
- errors() (vispa.workspace.Connection method), 118
- EVENT_DELAYS (vispa.remote.filesystem.WatchSubscriber attribute), 111

- exception_string() (in module vispa), 121
- execute() (vispa.extensions.codeeditor.controller.EditorController method), 101
- exists() (vispa.controller.filesystem.FSAjaxController method), 71
- exists() (vispa.remote.filesystem.FileSystem method), 111
- expand() (in module vispa.extensions.codeeditor.workspace), 83
- expand() (vispa.controller.filesystem.FSAjaxController method), 71
- expand() (vispa.remote.filesystem.FileSystem method), 111
- exposed_log() (vispa.workspace.LoggingService method), 119
- exposed_send() (vispa.workspace.LoggingService method), 119
- Extension() (class), 43
- extension() (vispa.server.Server method), 117
- extensions() (vispa.server.Server method), 117
- ## F
- failure() (vispa.extensions.dummy.controller.DummyController method), 85
- feedback() (vispa.controller.ajax.AjaxController method), 70
- fetch() (vispa.socketbus.PollingPublisher method), 118
- file_compare() (in module vispa.remote.filesystem), 112
- FILE_EXTENSIONS (vispa.remote.filesystem.FileSystem attribute), 110
- FileBrowserController (class in vispa.extensions.file), 85
- FileBrowserExtension (class in vispa.extensions.file), 85
- FileController (class in vispa.extensions.file.controller), 85
- filecount() (vispa.controller.filesystem.FSAjaxController method), 71
- FileHandler() (class), 60
- filelist() (vispa.controller.filesystem.FSAjaxController method), 71
- fileno() (vispa.workspace.WrappedChannelFile method), 120
- FileSystem (class in vispa.remote.filesystem), 110
- fileURL(), 65
- fill() (vispa.remote.helper.UTF8Buffer method), 112
- find_handler() (vispa.rest.RESTDispatcher method), 116
- find_node() (vispa.rest.RESTDispatcher method), 116
- fire_callback() (in module vispa), 121
- flush() (vispa.remote.filesystem.WatchSubscriber method), 112
- focus(), 64
- focusNeighbor(), 51
- FORBIDDEN_NAMES (vispa.models.user.User attribute), 101
- forgot_password() (vispa.models.user.User static method), 101
- forgotpassword() (vispa.controller.ajax.AjaxController method), 70
- format(), 42
- formatDate(), 43
- FSAjaxController (class in vispa.controller.filesystem), 70
- FSController (class in vispa.controller.filesystem), 71
- FSEvent (class in vispa.remote.fsmonitor), 110
- FSEvent (class in vispa.remote.fsmonitor.common), 107
- FSMonitor (class in vispa.remote.fsmonitor), 109
- FSMonitor (class in vispa.remote.fsmonitor.linux), 108
- FSMonitor (class in vispa.remote.fsmonitor.polling), 108
- FSMonitorDirWatch (class in vispa.remote.fsmonitor.polling), 108
- FSMonitorError, 108, 110
- FSMonitorFileWatch (class in vispa.remote.fsmonitor.polling), 109
- FSMonitorOSError, 108, 110
- FSMonitorThread (class in vispa.remote.fsmonitor), 109
- FSMonitorWatch (class in vispa.remote.fsmonitor.linux), 108
- FSMonitorWatch (class in vispa.remote.fsmonitor.polling), 109
- ## G
- GalleryController (class in vispa.extensions.gallery), 85
- GalleryExtension (class in vispa.extensions.gallery), 85
- generate_hash() (vispa.models.user.User static method), 101
- GET(), 56
- get(), 34, 52
- GET() (in module vispa.rest), 116
- get() (vispa.controller.AbstractController method), 11, 82
- get() (vispa.models.group.Group static method), 89
- get() (vispa.models.project.Project static method), 94
- get() (vispa.models.project.ProjectItem static method), 97
- get() (vispa.models.role.Permission static method), 98
- get() (vispa.models.role.Role static method), 99
- get() (vispa.models.user.User static method), 101
- get() (vispa.models.workgroup.Workgroup static method), 103
- get() (vispa.models.workgroup.WorkgroupItem static method), 105
- get() (vispa.workspace.ConnectionPool method), 119
- get() (vispa.workspace.InstancePool method), 119
- get_by_email() (vispa.models.user.User static method), 101
- get_by_hash() (vispa.models.user.User static method), 101
- get_by_id() (vispa.models.group.Group static method), 89

`get_by_id()` (vispa.models.project.Project static method), 94

`get_by_id()` (vispa.models.project.ProjectItem static method), 97

`get_by_id()` (vispa.models.role.Permission static method), 98

`get_by_id()` (vispa.models.role.Role static method), 100

`get_by_id()` (vispa.models.user.User static method), 101

`get_by_id()` (vispa.models.workgroup.Workgroup static method), 104

`get_by_id()` (vispa.models.workgroup.WorkgroupItem static method), 105

`get_by_id()` (vispa.models.workspace.Workspace static method), 106

`get_by_name()` (vispa.models.group.Group static method), 89

`get_by_name()` (vispa.models.project.Project static method), 94

`get_by_name()` (vispa.models.role.Permission static method), 98

`get_by_name()` (vispa.models.role.Role static method), 100

`get_by_name()` (vispa.models.user.User static method), 101

`get_by_name()` (vispa.models.workgroup.Workgroup static method), 104

`get_child_groups()` (vispa.models.group.Group method), 89

`get_cookie()` (in module vispa.browser), 116

`get_device_name()` (vispa.tools.device.DeviceTool method), 113

`get_dir_contents()` (in module vispa.remote.fsmonitor.polling), 109

`get_error_data()` (vispa.controller.error.ErrorController method), 70

`get_file()` (vispa.remote.filesystem.FileSystem method), 111

`get_file_content()` (vispa.remote.filesystem.FileSystem method), 111

`get_file_count()` (vispa.remote.filesystem.FileSystem method), 111

`get_file_info()` (in module vispa.remote.filesystem), 112

`get_file_list()` (vispa.remote.filesystem.FileSystem method), 111

`get_groups()` (vispa.models.project.Project method), 94

`get_groups()` (vispa.models.user.User method), 101

`get_info_data()` (vispa.models.jsondata.JSONData method), 91

`get_instance()` (in module vispa.workspace), 120

`get_item()` (vispa.models.jsondata.JSONData static method), 91

`get_items()` (vispa.models.project.Project method), 94

`get_items()` (vispa.models.workgroup.Workgroup method), 104

`get_managed_groups()` (vispa.models.user.User method), 101

`get_managed_projects()` (vispa.models.user.User method), 101

`get_managed_workgroups()` (vispa.models.user.User method), 101

`get_managers()` (vispa.models.group.Group method), 90

`get_managers()` (vispa.models.project.Project method), 94

`get_managers()` (vispa.models.workgroup.Workgroup method), 104

`get_mime_type()` (vispa.remote.filesystem.FileSystem method), 111

`get_mtime()` (vispa.remote.filesystem.FileSystem method), 111

`get_or_create_by_name()` (vispa.models.group.Group static method), 90

`get_or_create_by_name()` (vispa.models.project.Project static method), 94

`get_or_create_by_name()` (vispa.models.role.Permission static method), 98

`get_or_create_by_name()` (vispa.models.role.Role static method), 100

`get_or_create_by_name()` (vispa.models.user.User static method), 101

`get_parent_groups()` (vispa.models.group.Group method), 90

`get_permissions()` (vispa.models.project.Project_Group_Assoc method), 92

`get_permissions()` (vispa.models.project.Project_User_Assoc method), 92

`get_permissions()` (vispa.models.user.User method), 101

`get_polling_publisher()` (in module vispa.socketbus), 118

`get_project()` (vispa.models.project.ProjectItem method), 97

`get_projects()` (vispa.models.group.Group method), 90

`get_projects()` (vispa.models.user.User method), 102

`get_roles()` (vispa.models.user.User method), 102

`get_roles_of_group()` (vispa.models.project.Project method), 95

`get_roles_of_user()` (vispa.models.project.Project method), 95

`get_session_value()` (in module vispa.browser), 116

`get_socket_owner()` (vispa.Netstat static method), 121

`get_suggestions()` (vispa.remote.filesystem.FileSystem method), 111

`get_user_workspace_count()` (vispa.models.workspace.Workspace static method), 106

`get_user_workspaces()` (vispa.models.workspace.Workspace static method), 106

`get_users()` (vispa.models.group.Group method), 90

`get_users()` (vispa.models.project.Project method), 95

get_users() (vispa.models.workgroup.Workgroup method), 104	group (vispa.models.project.Project_Group_Assoc attribute), 92
get_value() (vispa.models.jsondata.JSONData static method), 91	group_add_child_group() (vispa.controller.usermanagement.UMAjaxController method), 72
get_values_by_key() (vispa.models.jsondata.JSONData static method), 92	group_add_manager() (vispa.controller.usermanagement.UMAjaxController method), 72
get_workgroup() (vispa.models.workgroup.WorkgroupItem method), 105	group_add_parent_group() (vispa.controller.usermanagement.UMAjaxController method), 72
get_workgroups() (vispa.models.user.User method), 102	group_add_user() (vispa.controller.usermanagement.UMAjaxController method), 73
get_workspace_connections() (vispa.workspace.ConnectionPool method), 119	group_confirm_child_group() (vispa.controller.usermanagement.UMAjaxController method), 73
get_workspace_instance() (vispa.server.AbstractExtension method), 8, 117	group_confirm_user() (vispa.controller.usermanagement.UMAjaxController method), 73
get_workspaceini() (vispa.remote.filesystem.FileSystem method), 111	group_create() (vispa.controller.usermanagement.UMAjaxController method), 73
getDefaultFileHandler(), 60	group_delete() (vispa.controller.usermanagement.UMAjaxController method), 73
getfac() (vispa.controller.filesystem.FSAjaxController method), 71	group_enter_parent_group() (vispa.controller.usermanagement.UMAjaxController method), 73
getfac() (vispa.remote.filesystem.FileSystem method), 111	group_get() (vispa.controller.usermanagement.UMAjaxController method), 73
getFile(), 56	group_get_all() (vispa.controller.usermanagement.UMAjaxController method), 74
getfile() (vispa.controller.filesystem.FSAjaxController method), 71	group_get_child_groups() (vispa.controller.usermanagement.UMAjaxController method), 74
getfile() (vispa.controller.filesystem.FSController method), 71	group_get_managers() (vispa.controller.usermanagement.UMAjaxController method), 74
getFileHandlers(), 60	group_get_parent_groups() (vispa.controller.usermanagement.UMAjaxController method), 74
getFragment(), 68	group_get_users() (vispa.controller.usermanagement.UMAjaxController method), 74
getId(), 31	Group_Group_Assoc (class in vispa.models.group), 87
getInst(), 59	group_id (vispa.models.group.Group_User_Assoc attribute), 87
getjson() (vispa.controller.ajax.AjaxController method), 70	group_id (vispa.models.project.Project_Group_Assoc attribute), 92
getNeighbor(), 46	group_leave_parent_group() (vispa.controller.usermanagement.UMAjaxController method), 74
getrpc() (vispa.extensions.codeeditor.controller.EditorController method), 83	group_remove_child_group() (vispa.controller.usermanagement.UMAjaxController method), 74
getstate() (vispa.remote.fsmonitor.polling.FSMonitorDirWatcher method), 108	group_remove_manager() (vispa.controller.usermanagement.UMAjaxController method), 75
getstate() (vispa.remote.fsmonitor.polling.FSMonitorFileWatcher method), 109	group_remove_user() (vispa.controller.usermanagement.UMAjaxController method), 75
getsuggestions() (vispa.controller.filesystem.FSAjaxController method), 71	
getView(), 44	
getworkspacedata() (vispa.controller.ajax.AjaxController method), 70	
getworkspaceini() (vispa.controller.filesystem.FSAjaxController method), 71	
GLOBAL_WORKSPACE_CONF (vispa.remote.filesystem.FileSystem attribute), 110	
graceful_shutdown() (vispa.controller.root.RootController method), 72	
Group (class in vispa.models.group), 88	

group_rename() (vispa.controller.usermanagement.UMAjaxController method), 75
group_set_password() (vispa.controller.usermanagement.UMAjaxController method), 75
group_set_privacy() (vispa.controller.usermanagement.UMAjaxController method), 75
group_set_status() (vispa.controller.usermanagement.UMAjaxController method), 75
Group_User_Assoc (class in vispa.models.group), 87
groupmatch() (vispa.tools.device.DeviceTool method), 113
groups (vispa.models.project.Project attribute), 95
guest_login() (vispa.controller.root.RootController method), 72
guest_login() (vispa.models.user.User static method), 102

H

handle_file_name_collision()
(vispa.remote.filesystem.FileSystem method), 111
has_access() (vispa.models.workspace.Workspace method), 106
has_group() (vispa.models.project.Project method), 95
has_permission() (vispa.models.user.User method), 102
has_session_value() (in module vispa.browser), 116
hash (vispa.models.user.User attribute), 102
hide(), 64, 67
host (vispa.models.workspace.Workspace attribute), 106
host() (vispa.workspace.Connection method), 118

I

id (vispa.models.group.Group attribute), 90
id (vispa.models.jsondata.JSONData attribute), 92
id (vispa.models.project.Project attribute), 95
id (vispa.models.project.ProjectItem attribute), 97
id (vispa.models.role.Permission attribute), 99
id (vispa.models.role.Role attribute), 100
id (vispa.models.user.User attribute), 102
id (vispa.models.workgroup.Workgroup attribute), 104
id (vispa.models.workgroup.WorkgroupItem attribute), 106
id (vispa.models.workspace.Workspace attribute), 106
idle_time() (vispa.tools.status.ThreadStatus method), 115
INACTIVE (vispa.models.group.Group attribute), 88
INACTIVE (vispa.models.project.Project attribute), 93
INACTIVE (vispa.models.user.User attribute), 101
index() (vispa.controller.bus.BusController method), 70
index() (vispa.controller.error.ErrorController method), 70
index() (vispa.controller.root.RootController method), 72
initPrefs(), 31
InstancePool (class in vispa.workspace), 119
is_active() (vispa.models.user.User static method), 102

is_browser_file() (vispa.remote.filesystem.FileSystem method), 111
is_valid() (vispa.models.user.User method), 102
is_valid() (vispa.models.workspace.Workspace method), 102
isbrowserfile() (vispa.controller.filesystem.FSAjaxController method), 71
isFocused(), 50, 62, 67
isManaged(), 50, 67
isVisible(), 50, 61, 67
items (vispa.models.project.Project attribute), 95
items (vispa.models.workgroup.Workgroup attribute), 104
itemsAcc() (mixin), 34
itemtype (vispa.models.project.ProjectItem attribute), 97
itemtype (vispa.models.workgroup.WorkgroupItem attribute), 106

J

join() (in module vispa.url), 118
JSONData (class in vispa.models.jsondata), 91
JSONData() (class), 59
JsonParameters (class in vispa.tools.json_parameters), 113

K

key (vispa.models.jsondata.JSONData attribute), 92
key (vispa.models.workspace.Workspace attribute), 106
KEYS (vispa.models.workspace.Workspace attribute), 106

L

last_password_reset (vispa.models.user.User attribute), 102
last_req_time() (vispa.tools.status.ThreadStatus method), 115
last_request (vispa.models.user.User attribute), 102
LinkMixin() (mixin), 57
load(), 31, 44, 53, 59
loadKey(), 59
LoadMixin() (mixin), 53
LocalConnectionFeeder (class in vispa.workspace), 119
LocalConnectionImpl (class in vispa.workspace), 119
localuser() (vispa.controller.ajax.AjaxController method), 70
log_exception() (in module vispa), 121
LoggerMixin() (mixin), 57
LoggingService (class in vispa.workspace), 119
login (vispa.models.workspace.Workspace attribute), 106
login() (vispa.controller.ajax.AjaxController method), 70
login() (vispa.controller.root.RootController method), 72
login() (vispa.models.user.User static method), 102
login_credentials (vispa.models.workspace.Workspace attribute), 106

- logout() (vispa.controller.root.RootController method), 72
- lookup_template() (vispa.plugins.template.MakoPlugin method), 107
- ls() (vispa.extensions.demo.controller.DemoController method), 84
- ls() (vispa.extensions.demo.workspace.DemoRpc method), 84
- ## M
- MainView() (class), 67
- make_dict() (vispa.models.workspace.Workspace method), 106
- MakoPlugin (class in vispa.plugins.template), 107
- MakoTool (class in vispa.tools.template), 115
- managers (vispa.models.group.Group attribute), 90
- managers (vispa.models.project.Project attribute), 95
- managers (vispa.models.workgroup.Workgroup attribute), 104
- MAX_BURST (vispa.extensions.codeeditor.workspace.CodeEditorRpc attribute), 82
- MAX_INLINE_SUBJECTS (vispa.remote.filesystem.WatchSubscriber attribute), 112
- MAX_RATE (vispa.extensions.codeeditor.workspace.CodeEditorRpc attribute), 82
- MAX_SUBJECT_NAMES (vispa.remote.filesystem.WatchSubscriber attribute), 112
- MenuButton() (class), 36
- MenuHeadless() (class), 37
- MenuItem() (class), 35
- MethodTool (class in vispa.tools.method), 114
- migrate() (in module vispa.models.alembic), 87
- MIN_PW_LENGTH (vispa.models.user.User attribute), 101
- Modify (vispa.remote.fsmonitor.common.FSEvent attribute), 107
- Modify (vispa.remote.fsmonitor.FSEvent attribute), 110
- Module() (class), 61
- module() (in module vispa.workspace), 120
- mount_extension_controller() (vispa.controller.root.RootController method), 72
- mount_static() (vispa.controller.AbstractController method), 11, 82
- move() (vispa.controller.filesystem.FSAjaxController method), 71
- move() (vispa.remote.filesystem.FileSystem method), 111
- MoveFrom (vispa.remote.fsmonitor.common.FSEvent attribute), 107
- MoveFrom (vispa.remote.fsmonitor.FSEvent attribute), 110
- MoveSelf (vispa.remote.fsmonitor.common.FSEvent attribute), 107
- MoveSelf (vispa.remote.fsmonitor.FSEvent attribute), 110
- MoveTo (vispa.remote.fsmonitor.common.FSEvent attribute), 107
- MoveTo (vispa.remote.fsmonitor.FSEvent attribute), 110
- ## N
- name (vispa.models.group.Group attribute), 90
- name (vispa.models.project.Project attribute), 95
- name (vispa.models.role.Permission attribute), 99
- name (vispa.models.role.Role attribute), 100
- name (vispa.models.user.User attribute), 102
- name (vispa.models.workgroup.Workgroup attribute), 104
- name (vispa.models.workspace.Workspace attribute), 106
- name() (vispa.extensions.codeeditor.CodeEditorExtension method), 83
- name() (vispa.extensions.core.CoreExtension method), 83
- name() (vispa.extensions.demo.DemoExtension method), 84
- name() (vispa.extensions.dummy.DummyExtension method), 85
- name() (vispa.extensions.file.FileBrowserExtension method), 85
- name() (vispa.extensions.gallery.GalleryExtension method), 86
- name() (vispa.extensions.terminal.TerminalExtension method), 86
- name() (vispa.server.AbstractExtension method), 8, 117
- NAME_CHARS (vispa.models.user.User attribute), 101
- NAME_LENGTH (vispa.models.user.User attribute), 101
- Netstat (class in vispa), 121
- new_state() (vispa.remote.fsmonitor.polling.FSMonitorDirWatch class method), 109
- new_state() (vispa.remote.fsmonitor.polling.FSMonitorFileWatch class method), 109
- ## O
- objClone(), 37
- on(), 58
- onBeforeClose(), 69
- onBlur(), 69
- onClose(), 69
- onFocus(), 68
- onHide(), 69
- onResize(), 69
- onShow(), 69
- open() (vispa.extensions.terminal.TerminalController method), 86
- open() (vispa.extensions.terminal.workspace.Terminal method), 86

`open()` (`vispa.workspace.Connection` method), 118

P

`package_files()` (in module `vispa.workspace`), 120

`parent_group_id` (`vispa.models.group.Group_Group_Assoc` attribute), 87

`parse_events()` (in module `vispa.remote.fsmonitor.linux`), 108

`passThru()` (`vispa.remote.helper.UTF8Buffer` method), 112

`password` (`vispa.models.group.Group` attribute), 90

`password` (`vispa.models.user.User` attribute), 102

`password()` (`vispa.controller.root.RootController` method), 72

`PASSWORD_RESET_DELAY` (`vispa.models.user.User` attribute), 101

`paste()` (`vispa.controller.filesystem.FSAjaxController` method), 71

`paste()` (`vispa.remote.filesystem.FileSystem` method), 111

`path` (`vispa.remote.fsmonitor.common.FSEvent` attribute), 107

`path` (`vispa.remote.fsmonitor.FSEvent` attribute), 110

`Permission` (class in `vispa.models.role`), 98

`permission_create()` (`vispa.controller.usermanagement.UMAjaxController` method), 75

`permission_delete()` (`vispa.controller.usermanagement.UMAjaxController` method), 76

`permission_get_all()` (`vispa.controller.usermanagement.UMAjaxController` method), 76

`permission_rename()` (`vispa.controller.usermanagement.UMAjaxController` method), 76

`permissions` (`vispa.models.role.Role` attribute), 100

`PermissionTool` (class in `vispa.tools.permission`), 114

`poll()` (`vispa.controller.bus.BusController` method), 70

`poll()` (`vispa.workspace.Connection` method), 119

`PollingPublisher` (class in `vispa.socketbus`), 118

`POST()`, 56

`POST()` (in module `vispa.rest`), 116

`PreferencesMixin()` (mixin), 54

`prefs.get()`, 54

`prefs.set()`, 54

`prefs.watch()`, 54

`print_stderr()` (`vispa.workspace.LocalConnectionImpl` method), 119

`print_stderr()` (`vispa.workspace.SSHConnectionImpl` method), 120

`privacy` (`vispa.models.group.Group` attribute), 90

`PRIVATE` (`vispa.models.group.Group` attribute), 88

`PRIVATE_WORKSPACE_CONF`
(`vispa.remote.filesystem.FileSystem` attribute), 110

`PrivateParameterFilter` (class in `vispa.tools.parameters`), 114

`process()` (`vispa.remote.filesystem.WatchSubscriber` method), 112

`processArgs()`, 63

`Project` (class in `vispa.models.project`), 92

`project_add_group()` (`vispa.controller.usermanagement.UMAjaxController` method), 76

`project_add_manager()` (`vispa.controller.usermanagement.UMAjaxController` method), 76

`project_add_user()` (`vispa.controller.usermanagement.UMAjaxController` method), 76

`project_create()` (`vispa.controller.usermanagement.UMAjaxController` method), 76

`project_delete()` (`vispa.controller.usermanagement.UMAjaxController` method), 76

`project_get_all()` (`vispa.controller.usermanagement.UMAjaxController` method), 77

`project_get_groups()` (`vispa.controller.usermanagement.UMAjaxController` method), 77

`project_get_managers()` (`vispa.controller.usermanagement.UMAjaxController` method), 77

`project_get_roles_of_group()`
(`vispa.controller.usermanagement.UMAjaxController` method), 77

`project_get_roles_of_user()`
(`vispa.controller.usermanagement.UMAjaxController` method), 77

`project_get_users()` (`vispa.controller.usermanagement.UMAjaxController` method), 77

`Project_Group_Assoc` (class in `vispa.models.project`), 92

`project_id` (`vispa.models.project.Project_Group_Assoc` attribute), 92

`project_id` (`vispa.models.project.Project_User_Assoc` attribute), 92

`project_id` (`vispa.models.project.ProjectItem` attribute), 97

`project_remove_group()` (`vispa.controller.usermanagement.UMAjaxController` method), 77

`project_remove_manager()`
(`vispa.controller.usermanagement.UMAjaxController` method), 78

`project_remove_user()` (`vispa.controller.usermanagement.UMAjaxController` method), 78

`project_rename()` (`vispa.controller.usermanagement.UMAjaxController` method), 78

`project_set_roles_of_group()`
(`vispa.controller.usermanagement.UMAjaxController` method), 78

`project_set_roles_of_user()`
(`vispa.controller.usermanagement.UMAjaxController` method), 78

`project_set_status()` (`vispa.controller.usermanagement.UMAjaxController` method), 78

`Project_User_Assoc` (class in `vispa.models.project`), 92

`ProjectItem` (class in `vispa.models.project`), 96

prompt(), 64
 PromptDialog() (class), 40
 PROTECTED (vispa.models.group.Group attribute), 88
 PUBLIC (vispa.models.group.Group attribute), 88
 publish() (in module vispa), 121
 pull(), 59
 push(), 59

R

raise_ajax() (in module vispa.remote), 112
 read() (vispa.extensions.terminal.TerminalController method), 86
 read() (vispa.extensions.terminal.workspace.Terminal method), 86
 read() (vispa.remote.helper.UTF8Buffer method), 112
 read_events() (vispa.remote.fsmonitor.FSMonitor method), 109
 read_events() (vispa.remote.fsmonitor.FSMonitorThread method), 109
 read_events() (vispa.remote.fsmonitor.linux.FSMonitor method), 108
 read_events() (vispa.remote.fsmonitor.polling.FSMonitor method), 108
 received_message() (vispa.socketbus.Bus method), 117
 received_message() (vispa.socketbus.PollingPublisher method), 118
 register() (vispa.controller.ajax.AjaxController method), 70
 register() (vispa.models.user.User static method), 102
 register_callback() (in module vispa), 121
 release() (vispa.controller.AbstractController method), 11, 82
 release_database() (vispa.controller.AbstractController method), 11, 82
 release_session() (vispa.controller.AbstractController method), 11, 82
 remove(), 31
 remove() (vispa.controller.filesystem.FSAjaxController method), 71
 remove() (vispa.models.workspace.Workspace static method), 106
 remove() (vispa.remote.filesystem.FileSystem method), 111
 remove_all_watches() (vispa.remote.fsmonitor.FSMonitor method), 109
 remove_all_watches() (vispa.remote.fsmonitor.FSMonitorThread method), 109
 remove_all_watches() (vispa.remote.fsmonitor.linux.FSMonitor method), 108
 remove_all_watches() (vispa.remote.fsmonitor.polling.FSMonitor method), 108
 remove_child_group() (vispa.models.group.Group method), 90
 remove_group() (vispa.models.project.Project method), 95
 remove_manager() (vispa.models.group.Group method), 90
 remove_manager() (vispa.models.project.Project method), 95
 remove_manager() (vispa.models.workgroup.Workgroup method), 104
 remove_session() (in module vispa.socketbus), 118
 remove_user() (vispa.models.group.Group method), 91
 remove_user() (vispa.models.project.Project method), 96
 remove_user() (vispa.models.workgroup.Workgroup method), 104
 remove_watch() (vispa.remote.fsmonitor.FSMonitor method), 109
 remove_watch() (vispa.remote.fsmonitor.FSMonitorThread method), 109
 remove_watch() (vispa.remote.fsmonitor.linux.FSMonitor method), 108
 remove_watch() (vispa.remote.fsmonitor.polling.FSMonitor method), 108
 rename() (vispa.controller.filesystem.FSAjaxController method), 71
 rename() (vispa.models.group.Group method), 91
 rename() (vispa.models.project.Project method), 96
 rename() (vispa.models.role.Permission method), 99
 rename() (vispa.models.role.Role method), 100
 rename() (vispa.models.workgroup.Workgroup method), 105
 rename() (vispa.remote.filesystem.FileSystem method), 111
 render(), 64
 replace(), 45, 47, 48
 resize() (vispa.extensions.terminal.TerminalController method), 86
 resize() (vispa.extensions.terminal.workspace.Terminal method), 86
 RESTController (class in vispa.rest), 116
 RESTDispatcher (class in vispa.rest), 116
 Role (class in vispa.models.role), 99
 role_create() (vispa.controller.usermanagement.UMAjaxController method), 79
 role_delete() (vispa.controller.usermanagement.UMAjaxController method), 79
 role_get_all() (vispa.controller.usermanagement.UMAjaxController method), 79
 role_get_permissions() (vispa.controller.usermanagement.UMAjaxController method), 79
 role_rename() (vispa.controller.usermanagement.UMAjaxController method), 79
 role_set_permissions() (vispa.controller.usermanagement.UMAjaxController method), 79
 roles (vispa.models.project.Project_Group_Assoc attribute), 92

- roles (vispa.models.project.Project_User_Assoc attribute), 92
- RootController (class in vispa.controller.root), 72
- ROP(), 42
- round_fs_resolution() (in module vispa.remote.fsmonitor.polling), 109
- rpyc() (vispa.workspace.Connection method), 119
- run() (vispa.remote.fsmonitor.FSMonitorThread method), 109
- run() (vispa.server.Server method), 117
- run() (vispa.socketbus.CleanerThread method), 117
- run() (vispa.workspace.LocalConnectionFeeder method), 119
- runDefaultHandler(), 60
- runningjob() (vispa.extensions.codeeditor.controller.EditorController method), 83
- runningjob() (vispa.extensions.codeeditor.workspace.CodeEditorRpc method), 96
- runSync(), 33
- S**
- save(), 53
- save_file() (vispa.remote.filesystem.FileSystem method), 111
- save_file_content() (vispa.remote.filesystem.FileSystem method), 111
- savefile() (vispa.controller.filesystem.FSAjaxController method), 71
- Scheduler (class in vispa.extensions.dummy.workspace), 84
- send() (vispa.controller.bus.BusController method), 70
- send() (vispa.socketbus.Bus method), 117
- send() (vispa.socketbus.PollingPublisher method), 118
- send_mail() (in module vispa), 121
- send_registration_mail() (vispa.models.user.User static method), 102
- send_status() (vispa.workspace.Connection method), 119
- send_topic() (in module vispa.remote), 112
- send_topic() (vispa.socketbus.Bus method), 117
- send_workspace_status() (vispa.workspace.Connection static method), 119
- SERVE_CONNECTION_INTERVAL (vispa.workspace.ConnectionPool attribute), 119
- Server (class in vispa.server), 117
- serveradmin (vispa.models.user.User attribute), 103
- set(), 34, 52
- set_cache() (vispa.controller.AbstractController method), 11, 82
- set_codepath() (in module vispa), 122
- set_configpath() (in module vispa), 122
- set_content() (vispa.models.project.ProjectItem method), 97
- set_content() (vispa.models.workgroup.WorkgroupItem method), 106
- set_cookie() (in module vispa.browser), 116
- set_datapath() (in module vispa), 122
- set_on_feed() (vispa.workspace.LocalConnectionImpl method), 119
- set_on_feed() (vispa.workspace.SSHConnectionImpl method), 120
- set_password() (vispa.models.group.Group method), 91
- set_password() (vispa.models.user.User static method), 103
- set_permissions() (vispa.models.role.Role method), 100
- set_privacy() (vispa.models.group.Group method), 91
- set_roles_of_group() (vispa.models.project.Project method), 96
- set_roles_of_user() (vispa.models.project.Project method), 96
- set_session_value() (in module vispa.browser), 116
- set_status() (vispa.models.group.Group method), 91
- set_status() (vispa.models.project.Project method), 96
- set_value() (vispa.models.jsondata.JSONData static method), 92
- set_workspaceini() (vispa.remote.filesystem.FileSystem method), 111
- setfacl() (vispa.controller.filesystem.FSAjaxController method), 71
- setfacl() (vispa.remote.filesystem.FileSystem method), 111
- setjson() (vispa.controller.ajax.AjaxController method), 70
- setLabel(), 62, 68
- setLoading(), 53, 62, 68
- setpassword() (vispa.controller.ajax.AjaxController method), 70
- setROP(), 42
- setstate() (vispa.remote.fsmonitor.polling.FSMonitorDirWatch method), 109
- setstate() (vispa.remote.fsmonitor.polling.FSMonitorFileWatch method), 109
- setup(), 52
- setup() (vispa.extensions.codeeditor.CodeEditorExtension method), 83
- setup() (vispa.extensions.core.CoreExtension method), 83
- setup() (vispa.extensions.demo.DemoExtension method), 84
- setup() (vispa.extensions.dummy.DummyExtension method), 85
- setup() (vispa.extensions.file.FileBrowserExtension method), 85
- setup() (vispa.extensions.gallery.GalleryExtension method), 86
- setup() (vispa.extensions.terminal.TerminalExtension method), 86

- setup() (vispa.remote.filesystem.FileSystem method), 111
 - setup() (vispa.server.AbstractExtension method), 8, 117
 - setup_thread_dump() (in module vispa), 122
 - setworkspaceini() (vispa.controller.filesystem.FSAjaxController method), 71
 - shortcutPP(), 43
 - ShortcutsMixin() (mixin), 57
 - show(), 64, 67
 - showSidebar(), 67
 - Sidebar() (class), 32
 - SidebarView() (class), 66
 - SIGTEM_SIGKILL_DELAY (vispa.extensions.codeeditor.workspace.CodeEditorRng attribute), 82
 - sigtest() (vispa.extensions.dummy.controller.DummyController method), 85
 - Socket() (class), 41
 - SocketMixin() (mixin), 58
 - SocketProxy() (class), 58
 - SocketPublisher (class in vispa.socketbus), 118
 - spawn(), 65
 - spawnFromFile(), 37
 - spawnInstance(), 29, 65
 - Split() (class), 48
 - SqlAlchemyTool (class in vispa.tools.db), 113
 - SSHConnectionImpl (class in vispa.workspace), 119
 - start (vispa.tools.status.ThreadStatus attribute), 115
 - start() (vispa.extensions.codeeditor.workspace.CodeEditorRng method), 83
 - start() (vispa.plugins.template.MakoPlugin method), 107
 - start() (vispa.server.Server method), 117
 - stat() (vispa.remote.filesystem.FileSystem method), 111
 - state (vispa.remote.fsmonitor.polling.FSMonitorDirWatch attribute), 109
 - state (vispa.remote.fsmonitor.polling.FSMonitorFileWatch attribute), 109
 - StateMixin() (mixin), 51
 - StateProxy() (class), 51
 - static() (in module vispa.url), 118
 - StaticController (class in vispa.controller), 82
 - staticURL(), 55
 - status (vispa.models.group.Group attribute), 91
 - status (vispa.models.group.Group_Group_Assoc attribute), 88
 - status (vispa.models.group.Group_User_Assoc attribute), 87
 - status (vispa.models.project.Project attribute), 96
 - status (vispa.models.user.User attribute), 103
 - status() (vispa.controller.root.RootController method), 72
 - status() (vispa.workspace.Connection method), 119
 - STATUS_MESSAGES (vispa.workspace.Connection attribute), 118
 - STATUSMAP (vispa.controller.error.ErrorController attribute), 70
 - StatusMonitor (class in vispa.tools.status), 114
 - stdin() (vispa.workspace.Connection method), 119
 - stdout() (vispa.workspace.Connection method), 119
 - stop() (vispa.plugins.template.MakoPlugin method), 107
 - stop() (vispa.remote.filesystem.WatchService method), 111
 - stop() (vispa.remote.fsmonitor.FSMonitorThread method), 110
 - stream() (vispa.workspace.LocalConnectionImpl method), 119
 - stream() (vispa.workspace.SSHConnectionImpl method), 120
 - str_cmp() (in module vispa.remote.filesystem), 112
 - strongly_expire() (in module vispa.controller), 82
 - subscribe() (in module vispa), 122
 - subscribe() (vispa.remote.filesystem.WatchService method), 111
 - subscribe() (vispa.socketbus.Bus method), 117
 - syncObj(), 60
 - syncVueFrag(), 33
- ## T
- Tabber() (class), 45
 - tempdir() (vispa.workspace.Connection method), 119
 - Terminal (class in vispa.extensions.terminal.workspace), 86
 - TerminalController (class in vispa.extensions.terminal), 86
 - TerminalExtension (class in vispa.extensions.terminal), 86
 - thread_stacktraces() (in module vispa), 122
 - ThreadStatus (class in vispa.tools.status), 115
 - thumbnail() (vispa.controller.filesystem.FSController method), 72
 - thumbnail() (vispa.remote.filesystem.FileSystem method), 111
 - TIMEOUT (vispa.workspace.SSHConnectionImpl attribute), 120
 - timestamp (vispa.models.jsondata.JSONData attribute), 92
 - TMPL (vispa.controller.error.ErrorController attribute), 70
 - Toggle() (class), 32
- ## U
- UINode() (class), 49
 - UINodeCollection() (class), 46
 - UMAjaxController (class in vispa.controller.usermanagement), 72
 - unbind() (vispa.remote.filesystem.WatchSubscriber method), 112
 - UNCONFIRMED (vispa.models.group.Group_Group_Assoc attribute), 87

UNCONFIRMED (vispa.models.group.Group_User_Assoc attribute), 87

unregister() (vispa.tools.status.StatusMonitor method), 115

unsubscribe() (vispa.remote.filesystem.WatchService method), 111

unsubscribe() (vispa.socketbus.Bus method), 117

unwatch() (vispa.controller.filesystem.FSAjaxController method), 71

unwatch() (vispa.remote.filesystem.FileSystem method), 111

update(), 34

update() (vispa.models.workspace.Workspace static method), 106

update() (vispa.remote.filesystem.WatchSubscriber method), 112

update_last_request() (vispa.models.user.User static method), 103

update_to_session() (in module vispa.browser), 116

updatePopovers(), 67

upload() (vispa.controller.filesystem.FSAjaxController method), 71

url (vispa.tools.status.ThreadStatus attribute), 115

User (class in vispa.models.user), 101

user (vispa.models.group.Group_User_Assoc attribute), 87

user (vispa.models.project.Project_User_Assoc attribute), 92

user (vispa.remote.fsmonitor.common.FSEvent attribute), 107

user (vispa.remote.fsmonitor.FSEvent attribute), 110

user_enter_group() (vispa.controller.usermanagement.UMAjaxController method), 79

user_get_groups() (vispa.controller.usermanagement.UMAjaxController method), 79

user_get_managed_groups() (vispa.controller.usermanagement.UMAjaxController method), 80

user_get_managed_projects() (vispa.controller.usermanagement.UMAjaxController method), 80

user_get_managed_workgroups() (vispa.controller.usermanagement.UMAjaxController method), 80

user_get_permissions() (vispa.controller.usermanagement.UMAjaxController method), 80

user_get_projects() (vispa.controller.usermanagement.UMAjaxController method), 80

user_get_roles() (vispa.controller.usermanagement.UMAjaxController method), 80

user_get_workgroups() (vispa.controller.usermanagement.UMAjaxController method), 80

user_id (vispa.models.group.Group_User_Assoc attribute), 87

user_id (vispa.models.jsondata.JSONData attribute), 92

user_id (vispa.models.project.Project_User_Assoc attribute), 92

user_id (vispa.models.workspace.Workspace attribute), 106

user_leave_group() (vispa.controller.usermanagement.UMAjaxController method), 80

user_leave_workgroup() (vispa.controller.usermanagement.UMAjaxController method), 80

users (vispa.models.group.Group attribute), 91

users (vispa.models.project.Project attribute), 96

users (vispa.models.workgroup.Workgroup attribute), 105

UserTool (class in vispa.tools.user), 115

UTF8Buffer (class in vispa.remote.helper), 112

uuid(), 42

V

value (vispa.models.jsondata.JSONData attribute), 92

ViewManager() (class), 49

vispa (module), 121

vispa.browser (module), 115

vispa.controller (module), 82

vispa.controller.ajax (module), 69

vispa.controller.bus (module), 70

vispa.controller.error (module), 70

vispa.controller.filesystem (module), 70

vispa.controller.root (module), 72

vispa.controller.usermanagement (module), 72

vispa.extensions (module), 87

vispa.extensions.codeeditor (module), 83

vispa.extensions.codeeditor.controller (module), 83

vispa.extensions.codeeditor.workspace (module), 82

vispa.extensions.core (module), 83

vispa.extensions.demo (module), 84

vispa.extensions.demo.controller (module), 84

vispa.extensions.demo.workspace (module), 84

vispa.extensions.dummy (module), 85

vispa.extensions.dummy.controller (module), 85

vispa.extensions.dummy.workspace (module), 84

vispa.extensions.file (module), 85

vispa.extensions.file.controller (module), 85

vispa.extensions.gallery (module), 85

vispa.extensions.terminal (module), 86

vispa.extensions.terminal.workspace (module), 86

vispa.models (module), 107

vispa.models.alembic (module), 87

vispa.models.group (module), 87

vispa.models.jsondata (module), 91

vispa.models.project (module), 92

vispa.models.role (module), 98

vispa.models.user (module), 101

vispa.models.workgroup (module), 103

vispa.models.workspace (module), 106

vispa.plugins (module), 107
 vispa.plugins.template (module), 107
 vispa.remote (module), 112
 vispa.remote.filesystem (module), 110
 vispa.remote.fsmonitor (module), 109
 vispa.remote.fsmonitor.common (module), 107
 vispa.remote.fsmonitor.linux (module), 108
 vispa.remote.fsmonitor.polling (module), 108
 vispa.remote.helper (module), 112
 vispa.rest (module), 116
 vispa.server (module), 117
 vispa.socketbus (module), 117
 vispa.tools (module), 115
 vispa.tools.ajax (module), 113
 vispa.tools.db (module), 113
 vispa.tools.device (module), 113
 vispa.tools.json_parameters (module), 113
 vispa.tools.method (module), 114
 vispa.tools.parameters (module), 114
 vispa.tools.permission (module), 114
 vispa.tools.status (module), 114
 vispa.tools.template (module), 115
 vispa.tools.user (module), 115
 vispa.tools.workspace (module), 115
 vispa.url (module), 118
 vispa.version (module), 118
 vispa.workspace (module), 118
 vispa.wsgi (module), 120
 VispaConfigParser (class in vispa), 121

W

wait() (vispa.extensions.dummy.workspace.DummyRpc
 method), 84
 watch() (vispa.controller.filesystem.FSAjaxController
 method), 71
 watch() (vispa.remote.filesystem.FileSystem method),
 111
 watches (vispa.remote.fsmonitor.FSMonitor attribute),
 109
 watches (vispa.remote.fsmonitor.linux.FSMonitor at-
 tribute), 108
 watches (vispa.remote.fsmonitor.polling.FSMonitor at-
 tribute), 108
 WatchService (class in vispa.remote.filesystem), 111
 WatchSubscriber (class in vispa.remote.filesystem), 111
 Workgroup (class in vispa.models.workgroup), 103
 workgroup_add_manager()
 (vispa.controller.usermanagement.UMAjaxController
 method), 80
 workgroup_add_user() (vispa.controller.usermanagement.UMAjaxController
 method), 81
 workgroup_create() (vispa.controller.usermanagement.UMAjaxController
 method), 81

workgroup_delete() (vispa.controller.usermanagement.UMAjaxController
 method), 81
 workgroup_get_managers()
 (vispa.controller.usermanagement.UMAjaxController
 method), 81
 workgroup_get_users() (vispa.controller.usermanagement.UMAjaxController
 method), 81
 workgroup_id (vispa.models.workgroup.WorkgroupItem
 attribute), 106
 workgroup_remove_manager()
 (vispa.controller.usermanagement.UMAjaxController
 method), 81
 workgroup_remove_user()
 (vispa.controller.usermanagement.UMAjaxController
 method), 81
 workgroup_rename() (vispa.controller.usermanagement.UMAjaxController
 method), 81
 WorkgroupItem (class in vispa.models.workgroup), 105
 Workspace (class in vispa.models.workspace), 106
 Workspace() (class), 29
 workspace_data() (vispa.controller.root.RootController
 method), 72
 workspace_id (vispa.models.jsondata.JSONData at-
 tribute), 92
 WorkspaceTool (class in vispa.tools.workspace), 115
 WrappedChannelFile (class in vispa.workspace), 120
 write() (vispa.extensions.terminal.TerminalController
 method), 86
 write() (vispa.extensions.terminal.workspace.Terminal
 method), 86
 writeln() (vispa.workspace.LocalConnectionImpl
 method), 119
 writeln() (vispa.workspace.SSHConnectionImpl
 method), 120