django-mama-cas Documentation

Release 0.6.0

Jason Bittel

January 21, 2014

django-mama-cas is a Python implementation of the Central Authentication Service (CAS) server protocol, providing single sign-on server functionality as a Django application. It implements the CAS 1.0 and 2.0 specifications, as well as some commonly used extensions to the protocol.

CAS is an HTTP-based protocol that provides single sign-on functionality to web services. It operates using tickets, unique text strings that are provided and validated by the server, allowing web services to authenticate a user without having access to the user's credentials.

CHAPTER 1

Contents

1.1 Installation

1.1.1 Prerequisites

The primary prerequisite of django-mama-cas is Django itself. For django-mama-cas 0.6.0, Django 1.4 or later is required. Earlier versions of Django may work, but are not tested or supported. See the Django downloads page for information on downloading and installing Django.

If you're installing django-mama-cas manually, such as from the GitHub repository, you'll need to install the Python requests library. Install it with:

```
pip install requests
```

1.1.2 Installing

There are several different ways to install django-mama-cas, depending on your preferences and needs. In all cases, it is recommended to run the installation within a virtualenv for isolation from other system packages.

Via pip

The easiest way to install it is with pip:

```
pip install django-mama-cas
```

Via a downloaded package

If you cannot access pip or prefer to install the package manually, download it from PyPI. Extract the downloaded archive and install it with:

```
python setup.py install
```

Via GitHub

To stay current with the latest development, clone the active development repository on GitHub:

```
git clone git://github.com/jbittel/django-mama-cas.git
```

If you don't want a full git repository, download the latest code from GitHub as a tarball.

1.1.3 Configuring

First, add django-mama-cas to the `INSTALLED_APPS` setting within your project's `settings.py` (or equivalent) file:

```
INSTALLED_APPS = (
    # ...existing apps...
    'mama_cas',
)
```

Once added, run `manage.py syncdb` to create the required database tables.

URL paths

django-mama-cas includes a Django `URLconf` that provides the required CAS URIs (e.g. `login/`, `logout/`, `validate/`, etc.). They are located in `mama_cas.urls` and can be included directly in your project's root `URLconf` with the following:

```
urlpatterns = patterns('',
    # ...existing urls...
    (r'', include('mama_cas.urls')),
)
```

This makes the CAS server available at the top level of your project's URL. If you prefer to access it within a subdirectory, add a base to the included URLs. For example, to make the CAS server available under the `/cas/` directory, use this instead:

```
(r'^cas/', include('mama_cas.urls')),
```

Changing the URLs within `mama_cas.urls` is not recommended as it will likely break standard CAS behavior.

Sessions

django-mama-cas relies on standard Django sessions to govern single sign-on sessions. There are two Django session settings that will likely need to be changed from their defaults:

SESSION_COOKIE_AGE It is recommended this be set shorter than the default of two weeks.

SESSION_EXPIRE_AT_BROWSER_CLOSE This should be set to `True` to conform to the CAS specification.

For information on how sessions work within Django, read the session documentation. These settings ought to be configured to fit your environment and security requirements.

1.1.4 Authentication

At least one authentication backend must be installed and configured, depending on your authoritative authentication source. django-mama-cas does not perform authentication itself, but relies on the active authentication backends for that task. The process of configuring authentication backends will change depending on the individual backend.

See also:

- Django user authentication documentation
- Authentication packages for Django

1.2 Settings

django-mama-cas can be configured using several custom settings. None are required and have sane defaults, but can be used to customize the behavior.

`django.conf.settings.`**`MAMA_CAS_TICKET_EXPIRE`**

Default `5`

Controls the length of time, in minutes, between when a service or proxy ticket is generated and when it expires. If the ticket is not validated before this time has elapsed, it will become invalid. This does **not** affect the duration of a user's single sign-on session.

`django.conf.settings.`**`MAMA_CAS_TICKET_RAND_LEN`**

Default `32`

Sets the number of random characters created as part of the ticket string. It should be long enough that the ticket string cannot be brute forced within a reasonable amount of time. Longer values are more secure, but could cause compatibility problems with some clients.

`django.conf.settings.`**`MAMA_CAS_USER_ATTRIBUTES`**

Default `{}`

A dictionary of name and User attribute values to be returned along with a service or proxy validation success. The key can be any meaningful string, while the value must correspond with an attribute on the User object. For example:

```
MAMA_CAS_USER_ATTRIBUTES = {
    'givenName': 'first_name',
    'sn': 'last_name',
    'email': 'email',
}
```

`django.conf.settings.`**`MAMA_CAS_PROFILE_ATTRIBUTES`**

Default `{}`

A dictionary of name and User profile attribute values to be returned along with a service or proxy validation success. The key can be any meaningful string, while the value must correspond with an attribute on the User profile object. If no User profile is configured or available, this setting will be ignored. For example:

```
MAMA_CAS_PROFILE_ATTRIBUTES = {
    'employeeID': 'id_number',
}
```

Note: This setting is intended for use with Django 1.4. In Django 1.5 and later, the recommended method for storing custom profile information is through a custom User model.

`django.conf.settings.`**`MAMA_CAS_VALID_SERVICES`**

Default `()`

A list of valid Python regular expressions that a service URL is tested against when a ticket is validated. If none of the regular expressions match the provided URL, the validation request fails. If no valid services are configured, any service URL is allowed. For example:

```
MAMA_CAS_VALID_SERVICES = (
    'https?://www\.example\.edu/secure/.*',
    'https://.*\.example\.com/.*',
)
```

The `url` parameter is also checked against this list of services at logout. If the provided URL does not match one of these regular expressions, it will be ignored.

`django.conf.settings.`**`MAMA_CAS_FOLLOW_LOGOUT_URL`**

Default `False`

Controls the client redirection behavior when the `url` parameter is specified at logout. When this setting is `False`, the client will be redirected to the login page with the specified URL displayed as a recommended link to follow. When this setting is `True`, the client will be redirected to the specified URL.

If the `url` parameter is not specified or if it is not a valid service URL, the client will be redirected to the login page with no message displayed, irrespective of this setting.

Note: The default setting of `False` conforms to the CAS protocol specification.

1.3 Templates

django-mama-cas comes with several templates implementing both internal and external functionality. Depending on your needs, you will likely want to either customize the HTML templates or replace them entirely.

Note: Changes made directly to these template files will be lost when django-mama-cas is updated.

mama_cas/login.html

This is the main authentication form template used by `LoginView` implementing standard username and password authentication and is used when user credentials are required. During the login process, it displays appropriate success or failure information to the user. When the user logs out, they are redirected back to this template, with a message indicating a successful logout.

mama_cas/warn.html

This template is used by `LoginView` when the warn parameter from `LoginFormWarn` is in effect. This causes the login process to not be transparent to the user. When an authentication attempt occurs, this template is displayed to provide continue or cancel options for the user.

mama_cas/validate.xml

This template is used for CAS 2.0 service and proxy validation responses. Within the body of the authentication success block is an include for returning custom user attributes. These attributes are not part of the official CAS specification, and three different attribute formats have emerged. The include path should point to one of these templates:

mama_cas/attributes-jasig.xml

Provides custom user attributes in JASIG format:

```
<cas:attributes>
    <cas:givenName>Ellen</cas:givenName>
    <cas:sn>Cohen</cas:sn>
    <cas:email>ellen@example.com</cas:email>
</cas:attributes>
```

Note: This is the default custom attributes format.

mama_cas/attributes-rubycas.xml

Provides custom user attributes in RubyCAS format:

```
<cas:givenName>Ellen</cas:givenName>
<cas:sn>Cohen</cas:sn>
<cas:email>ellen@example.com</cas:email>
```

mama_cas/attributes-namevalue.xml

Provides custom user attributes in Name-Value format:

```
<cas:attribute name='givenName' value='Ellen' />
<cas:attribute name='sn' value='Cohen' />
<cas:attribute name='email' value='ellen@example.com' />
```

mama_cas/proxy.xml

Used for CAS 2.0 proxy-granting ticket validation responses.

1.3.1 Extending

The default templates provide a number of blocks for easily inserting and modifying content. If only minor changes are required, simply extend an existing template. Here are the basic steps:

1. Create a new template file within a templates directory of your project. For example, to add a header above the login form with additional styling, create a file with the additional content:

```
{% extends "mama_cas/login.html" %}

{% block extra_head %}
    {{ block.super }}
    <style>#header { font-size:3em; text-align:center; color:#aaa; }</style>
{% endblock extra_head %}

{% block header %}
    <h1>If You Can Believe Your Eyes and Ears</h1>
{% endblock header %}
```

2. Make sure the template directory where this file is located is accessible by a template loader.

3. Tell django-mama-cas to use the new template by specifying it within the URLconf. For example, if you are overriding the login template, it could look something like this:

```
urlpatterns = patterns('',
    url(r'^login/?$',
    LoginView.as_view(template_name="login.html"),
    name='cas_login'),
    # ...
)
```

As the final step is a bit of a hack, it is worthwhile to use something like django-overextends to enable simultaneously overriding and extending the stock template.

1.3.2 Replacing

If the required changes are substantial, it may be easier to replace the stock template. Instead of extending the template as described in step one, replace it entirely. If it is located in a similar path to the original template and appears earlier in the template search order, it will be used in place of the stock template.

In addition to the login form, some elements a custom login template should include are:

Messages The `messages` framework displays information to the user for a login or logout event.

Non-field errors The form's `non_field_errors` inform the user of authentication failures and other login problems.

1.4 Management Commands

django-mama-cas ships with custom management commands to aid in some common tasks. You can see which management commands are available by running:

```
manage.py
```

The commands specific to django-mama-cas will show up underneath the `[mama_cas]` heading. To run a given command:

```
manage.py <command name>
```

1.4.1 Commands

cleanupcas Tickets created by django-mama-cas are not removed from the database at the moment of invalidation. Running this command will delete all invalid tickets from the database. Tickets are invalidated either when they expire a configurable number of minutes after creation or by being consumed. Either situation means the ticket is no longer valid for future authentication attempts and can be safely deleted.

It is recommended that this command be run on a regular basis so invalid tickets do not become a performance or storage concern.

1.5 Forms

django-mama-cas includes a form class implementing standard username and password authentication. In most cases, this will be the form of authentication required. Trust authentication can be used with CAS, but the requirements will be highly implementation dependent.

1.5.1 Authentication Forms

class `mama_cas.forms.`**`LoginForm`**

This is the base form for handling standard username and password authentication credentials. It contains the following fields:

`username` The username of the client requesting authentication. This field is required.

password The password of the client requesting authentication. This field is required.

service The service the client is attempting to access, typically represented as a URL. This is a hidden, optional field and is automatically added to the form when present.

The form's `clean()` method attempts authentication against the configured authentication backends and verifies the user account is active. If either check fails, a `FormValidation` error is raised with an appropriate error message.

The following form classes all inherit from `LoginForm`, providing additional or alternate behavior during the login process.

class `mama_cas.forms.`**LoginFormWarn**
A subclass of `LoginForm` which adds one additional, optional field:

warn A checkbox that indicates whether or not the single sign-on process should be transparent. This causes the user to be notified before being authenticated to another service and provides the option to continue the authentication attempt or end the single sign-on session.

class `mama_cas.forms.`**LoginFormEmail**
A subclass of `LoginForm` which adds no additional fields but performs additional cleanup on the `username` field. If an email address is provided for the username, it extracts only the username portion of the string.

1.5.2 Additional Forms

class `mama_cas.forms.`**WarnForm**
This form is used when warning the user that an authentication attempt is taking place. It contains no visible form fields, but contains two hidden fields, `service` and `gateway`. This allows the values of these parameters to be passed through the warning process.

1.6 CAS Protocol

The official CAS protocol specification can be found at http://www.jasig.org/cas/protocol. Where appropriate, comments within the code include numbers in parenthesis (e.g. `(2.3)`) corresponding to the section number within the CAS protocol documentation where that functionality is described. Additionally, views are labeled with a CAS version number in brackets (e.g. `[CAS 2.0]`) corresponding to the CAS version that defines that particular URI.

CAS 1.0 is primarily a text-based protocol that returns a simple "yes" or "no" response indicating a validation success or failure. CAS 2.0 returns XML fragments for validation responses and is capable of including a great deal of additional data in the process.

See also:

- CAS Protocol
- CAS User Manual
- CAS 1 Architecture
- CAS 2 Architecture
- Proxy Authentication

1.6.1 Protocol Deviations

There are some areas where django-mama-cas deviates from the official CAS specification to take advantage of built-in Django functionality. These changes do not alter the contract between the client, service and CAS server.

Login ticket (3.5) This ticket string created for the login form is passed along with the username and password to prevent the replaying of credentials. django-mama-cas does not implement login tickets and instead relies on the built-in CSRF protection for the login form.

Ticket-granting ticket (3.6) This ticket string is stored on the server and keys to a ticket-granting cookie provided by the client to identify an existing single sign-on session. django-mama-cas does not implement ticket-granting tickets and instead uses Django sessions to determine whether or not a single sign-on session has been established.

Custom attributes User attributes can be returned along with a CAS 2.0 service or proxy validation success. This is not part of the official CAS specification, but is widely used in practice.

Follow logout url Setting `MAMA_CAS_FOLLOW_LOGOUT_URL` to `True` alters the server's behavior at logout from the CAS specification. Depending on the services configured for CAS usage, this change can provide a more expected logout behavior.

1.7 Changelog

These are the notable changes for each django-mama-cas release. Backwards incompatible changes or other upgrade issues are also described here. For additional detail, read the complete commit history. From version 0.4.0 and following, version numbers follow the semantic versioning scheme.

django-mama-cas 0.6.0 [2013-09-04]

- Add Python 3 compatibility
- Add a setting to follow provided logout URLs

django-mama-cas 0.5.0 [2013-04-29]

- Fix login template not validating data properly
- Respect REQUESTS_CA_BUNDLE environment variable
- Fix login failures with case-sensitive authentication backends
- Support for Django 1.5 custom User models

django-mama-cas 0.4.0 [2013-01-31]

- Implement service management setting
- Improve logging levels and specificity
- Fix ticket expiration setting name
- Fix PGTs expiring according to the standard expiration value

django-mama-cas 0.3 [2012-10-26]

- Implement warn parameter for the credential acceptor
- Parse XML in tests to better check validity
- Fix partial logout with the renew parameter
- Implement custom attributes returned with a validation success

django-mama-cas 0.2 [2012-07-12]

- Implement internationalization
- Add proxy ticket validation
- Substantial improvements to the test suite

- Add traversed proxies to proxy validation response
- Add form class to extract usernames from email addresses

m

mama_cas.forms, ??