

---

# **django-numeric Documentatation**

*Release 0.2.0*

**Huseyin Yilmaz**

April 25, 2015



<b>1</b>	<b>What is numerics</b>	<b>3</b>
<b>2</b>	<b>What is django-numerics</b>	<b>5</b>
<b>3</b>	<b>Why this is helpful</b>	<b>7</b>
<b>4</b>	<b>Install django-numerics</b>	<b>13</b>
<b>5</b>	<b>Integration</b>	<b>15</b>
<b>6</b>	<b>django-numerics settings</b>	<b>17</b>
6.1	DJANGO_NUMERICS_SALT (Mandatory)	17
6.2	DJANGO_NUMERICS_SECRET_KEY (Mandatory for crypto serializer)	17
6.3	DJANGO_NUMERICS_SERIALIZER_BACKEND	17
6.4	DJANGO_NUMERICS_VIEW	17
6.5	DJANGO_NUMERICS_HELP_VIEW	17
6.6	DJANGO_NUMERICS_ENABLED	18
6.7	Run tests	18
6.8	Build documentation	18
<b>7</b>	<b>Usage</b>	<b>19</b>
7.1	Registration	19
7.2	Implementing an endpoint	20
<b>8</b>	<b>Widgets</b>	<b>21</b>
8.1	Label from JSON data widget	21
8.2	Number from JSON data widget	21
<b>9</b>	<b>Authentication</b>	<b>23</b>
<b>10</b>	<b>Permission</b>	<b>25</b>
<b>11</b>	<b>Serializers</b>	<b>27</b>
11.1	django numerics serializers.DebugSerializer (default)	27
11.2	django numerics serializers.BasicSerializer	27
11.3	django numerics serializers.CryptoSerializer	27



Give your app a mobile dashboard.

django-numeric is a numeric dashboard endpoint provider for django.



---

## What is numerics

---

Numerics is a dashboard application for your mobile phone and tablet. <http://cynapse.com/numerics/>





---

## What is django-numerics

---

django-numerics is a django application that provides 2 things. An easy interface to create endpoints for custom json widgets for numeric dashboard. Provides a user interface that lists all the endpoints that current user have access.

📄 192.168.1.124:8000/numerics/

### Numerics endpoints

## Numeric endpoints.

Just copy endpoint url and use the url to create a custom json widget in numerics application.

EndPoint	Url	Integration
daily-failed-transactions	<a href="#">Click to open endpoint</a>	<a href="#">Integration help</a>
db-status	<a href="#">Click to open endpoint</a>	<a href="#">Integration help</a>
number_of_messages	<a href="#">Click to open endpoint</a>	<a href="#">Integration help</a>
daily-sent-email	<a href="#">Click to open endpoint</a>	<a href="#">Integration help</a>
cache-status	<a href="#">Click to open endpoint</a>	<a href="#">Integration help</a>
daily-successful-transactions	<a href="#">Click to open endpoint</a>	<a href="#">Integration help</a>
number-of-users	<a href="#">Click to open endpoint</a>	<a href="#">Integration help</a>

Interface also provides an integration guide that specialized for each endpoint.



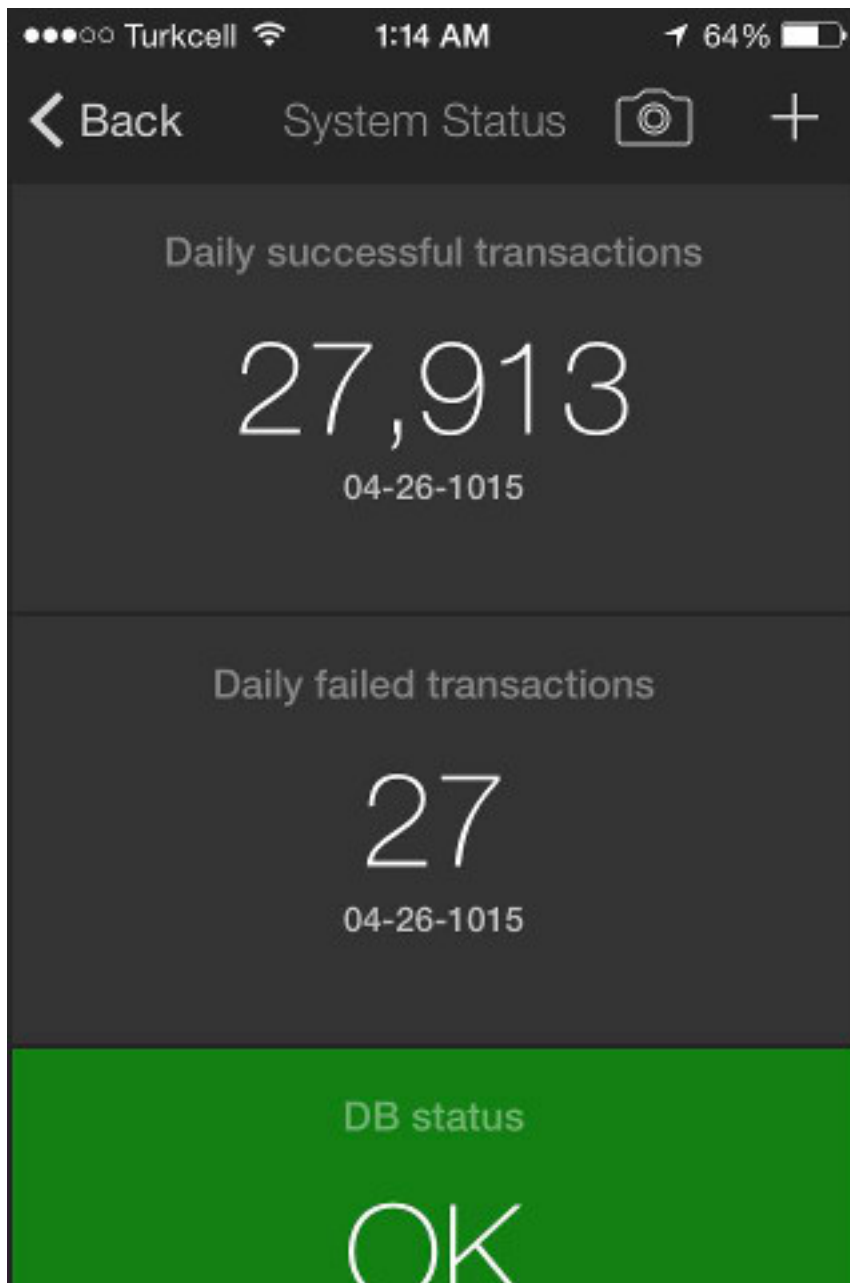
---

### Why this is helpful

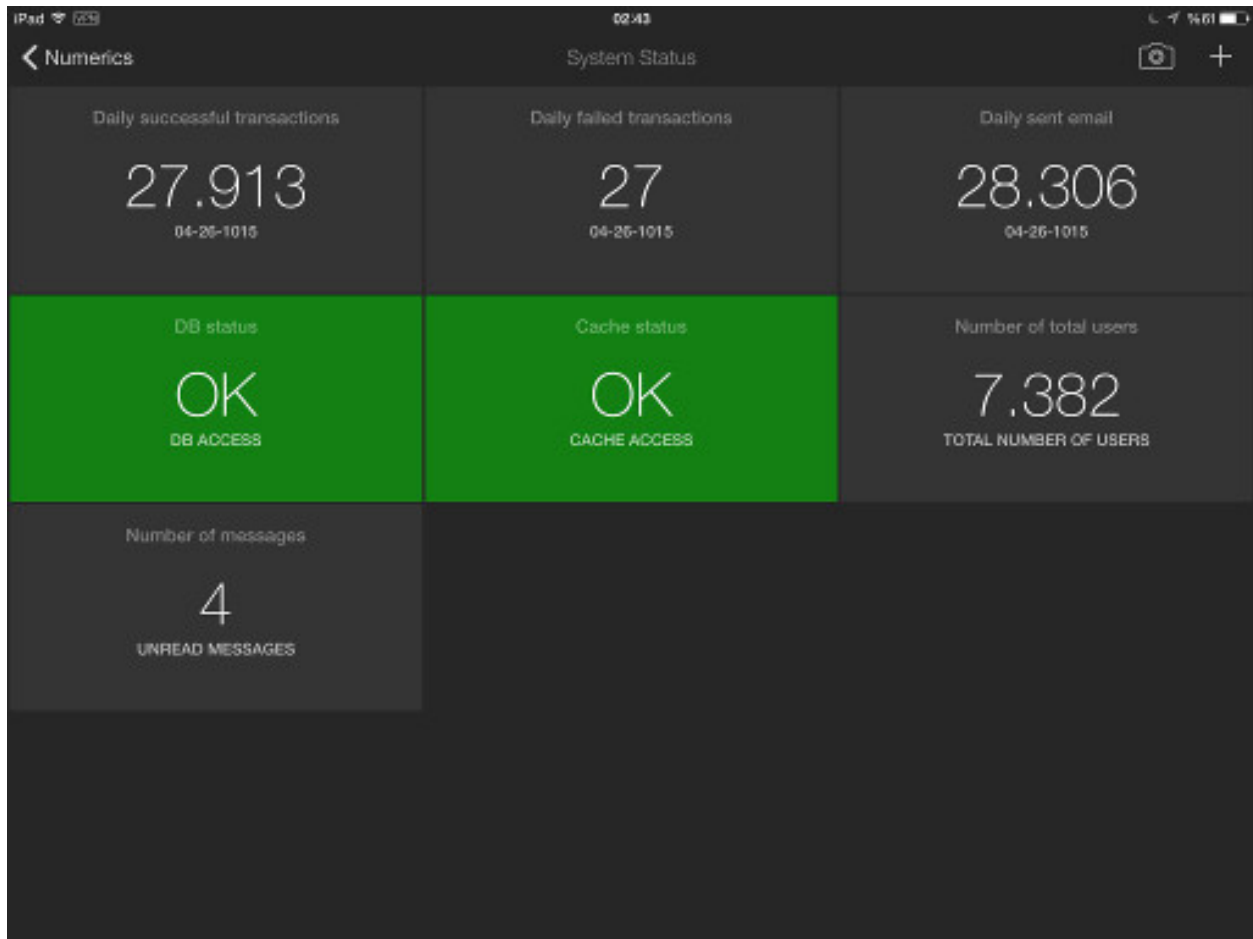
---

Every project that has some data that can should be checked frequently. with numerics dashboard and django-numerics application it is very easy to have a very slick interface on your mobile phone for your project.

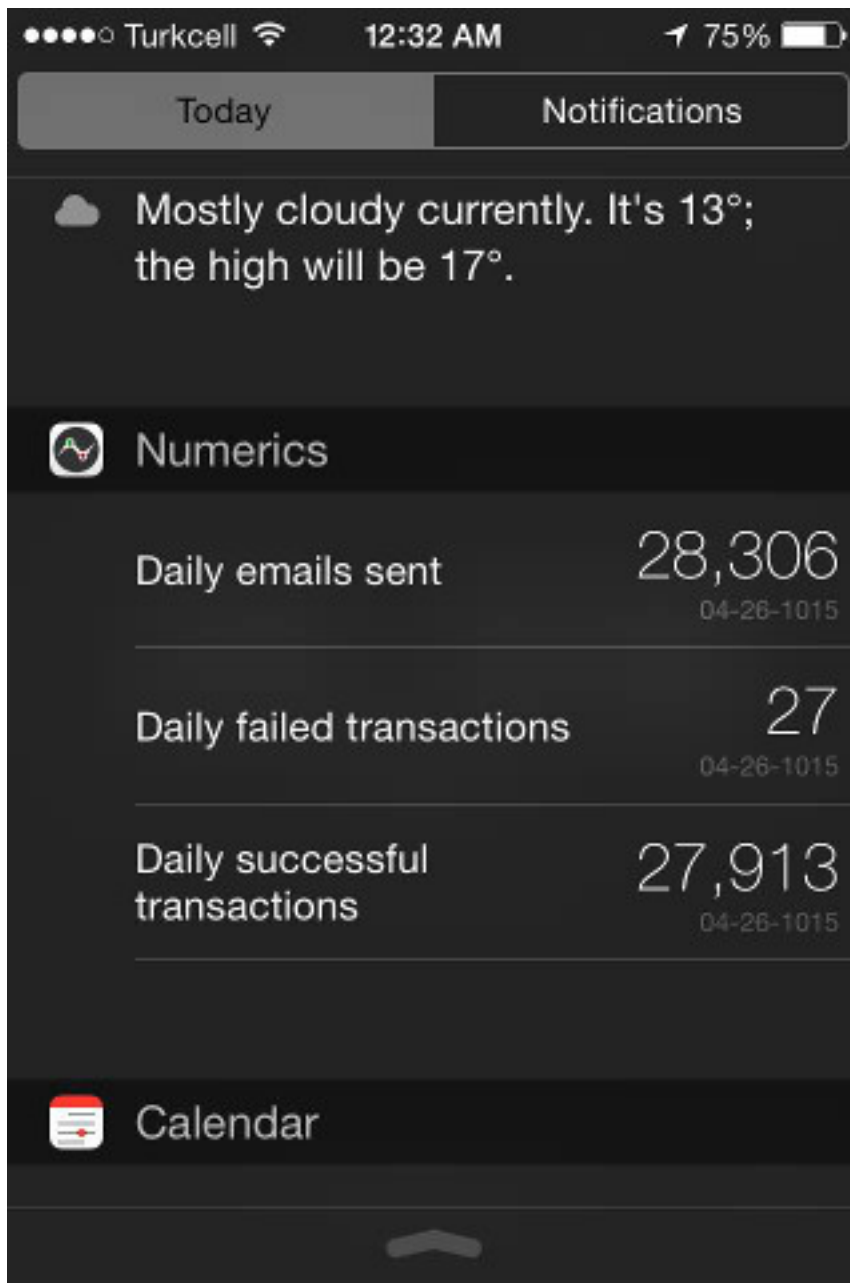
1. Your project will have a dashboard on mobile devices



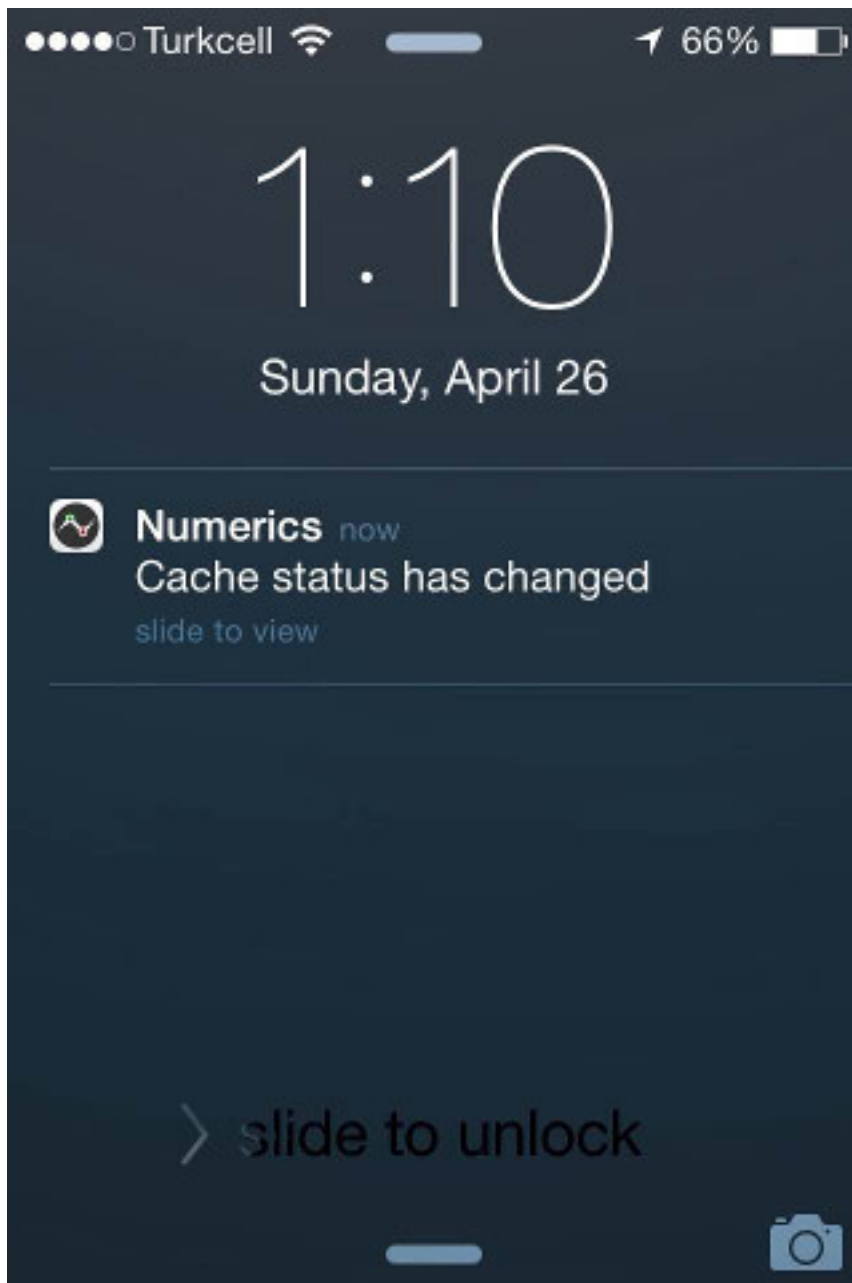
and tablets



2. You can show your data on *today screen* of your mobile phone and tablet.



3. You can receive notification when certain widget data is changed.



4. If you have a apple watch you can show your data on your watch.





---

## Install django-numeric

---

django-numeric can be installed using pip.

```
$ pip install django-numeric
```

Or source code can be downloaded from [github](#).



---

## Integration

---

1. To use django-numeric in a project first add it to `INSTALLED_APPS` in your project's `settings.py` file.

```
INSTALLED_APPS = (
    'django.contrib.admin',
    'django.contrib.auth',
    ...
    # add django-numeric to installed apps
    'django-numeric',
)
```

2. Add following settings to your project's `settings.py` file.

```
# for more info about django numeric settings:
# http://django-numeric.readthedocs.org/en/latest/#django-numeric-settings
DJANGO_NUMERIC_SALT='salt',
DJANGO_NUMERIC_SECRET_KEY= '_i7QFz8nH19camV1PTono1ruXNtdOCPPQRZo22ckZXg=',
DJANGO_NUMERIC_SERIALIZER_BACKEND = \
    'django-numeric.serializers.CryptoSerializer'
```

3. Go to main `urls` file and add django-numeric endpoints to `url` patterns

```
urlpatterns = patterns(
    '',
    url(r'', include(core.urls)),
    url(r'^admin/', include(admin.site.urls)),
    url(r'^accounts/', include(accounts.urls)),
    ## add django-numeric to urls.py
    url(r'^numeric/', include(django-numeric.urls)),
)
```

4. Register some endpoints for your dashboard. For instance following code adds number of current users as an endpoint.

```
from django-numeric import NumberResponse
from django-numeric import register
def total_users(user):
    """Return total number of users."""
    user_count = User.objects.filter(is_active=True).count()
    return NumberResponse(user_count, 'Total number of users')

# register endpoint to django-numeric
register('total-users', total_users, NumberResponse)
```

In this case, registered endpoint does not have user specific info. User specific info could be provided by using user argument of endpoint function.

After endpoint registration, open <http://localhost:8000/numeric> to see list of endpoints for current user. You will also find instructions about how to integrate those endpoints on that page. When you open the url, make sure that you are logged in with a user. If there is no logged in user, you will get a 404. This behaviour can be changed by providing a new permission function. See permission section for more information.

---

## django-numeric settings

---

### 6.1 DJANGO\_NUMERICS\_SALT (Mandatory)

Salt is used in creating md5 of the endpoint names. It is also useful to have project specific urls if you are using basic serializer.

### 6.2 DJANGO\_NUMERICS\_SECRET\_KEY (Mandatory for crypto serializer)

Hexadecimal value that will be used by crypto serializer. To generate a unique value, remove this setting and run the project. Generated error log will have uniquely generated SECRET\_KEY. You should be seeing following log message. Notice the unique key value at the end of log message:

```
django.core.exceptions.ImproperlyConfigured: DJANGO_NUMERICS_SECRET_KEY must be a hexadecimal value
```

Now you can add that uniquely generated SECRET\_KEY in settings

```
DJANGO_NUMERICS_SECRET_KEY = 'WZOjKcUw8mgnsMHHHklZX8azsDqvS5gY3PdNk6FIPIU='
```

### 6.3 DJANGO\_NUMERICS\_SERIALIZER\_BACKEND

Changes how djangonumeric endpoint urls are generated. Please see *serializers* section of documentation for choices.

### 6.4 DJANGO\_NUMERICS\_VIEW

django-numeric comes with a default interface. But you can change the default interface to fit your projects, design. default value is *djangonumerics/index.html*

### 6.5 DJANGO\_NUMERICS\_HELP\_VIEW

django-numeric comes with numeric dashboard installation instruction for every endpoint. With this setting, template that creates help pages can be changed. default value is *djangonumerics/help.html*

## 6.6 DJANGO\_NUMERICS\_ENABLED

With this settings, all djangonumerics endpoints can be disabled. Default value is True

## 6.7 Run tests

To run tests, first make sure that django is installed on current environment. Than run following command

```
$ python setup.py test
```

## 6.8 Build documentation

```
$ pip install -r doc_requirements.txt
$ python setup.py build_sphinx
```

---

## Usage

---

### 7.1 Registration

In order to add a new widget to your numerics dashboard, first you need to register an endpoint on your application. registration of an endpoint is a very simple process. just call `djangonumerics.api.register` with endpoint information. Here is signature of register function.

```
def register(name, func, response_type, args=None, kwargs=None,
            cache_timeout=0, permission_func=grant_access):
    ...
```

Here is the explanation of all arguments.

1. **name:** name of the endpoint. This will be used as an identifier for you endpoint. Make sure that it is unique. If you try to register multiple endpoints with the same name latter ones will be ignored.
2. **func, args, kwargs:** Those should be your endpoint function and its arguments. Your endpoint function will be called as following.

```
endpoint_response = func(user, *args, **kwargs)
```

So normally your endpoint function will be a normal function that takes a django user as an argument and returns a response objects that is instance of one of widget responses from `djangonumerics.responses`. But you can provide extra arguments from `args` and `kwargs` variables.

3. **response\_type:** This is a response type of endpoint function. Every endpoint will be formatted for certain widget. So response type of the endpoints should stay same at all times. This value should be one of the response classes in `djangonumerics.responses` module. Chose the response type for widget that you will use this endpoint with.
4. **cache\_timeout:** Normally endpoint function will be called for every request. But you can cache the endpoint response for any period of time. By default caching is disabled.
5. **permission\_func:** This function is used to decide if a user has permission for that endpoint. it takes a user and an internal endpoint namedtuple as an argument and return a boolean value. `permission_func` will be explained more in permission section.

Here is some example registration calls.

```
# caching number of users value for 60 seconds.
register('total-users', total_users, NumberResponse, cache_timeout=60)
# caching the return value for a day
register('employee-of-the-month', calculate_eom, LabelResponse,
        cache_timeout=1*24*60*60)
# using same endpoint for different backends
register('invalid-paypal-transactions', invalid_transactions_endpoints,
```

```
    NumberResponse, kwargs={'backends': ['paypal']},)
register('invalid-payu-transactions', invalid_transactions_endpoints,
        NumberResponse, kwargs={'backends': ['payu']},)
```

## 7.2 Implementing an endpoint

endpoint is a very a function that accepts user object as argument and returns instance of a BaseResponse subclass like `django-numeric.LabelResponse` or `django-numeric.NumberResponse`.

Here we are creating a total user count endpoint and register it as `total-user`. We are returning `NumberResponse` object so this endpoint is for “Number from JSON” widget. See widgets section to learn which response type is for which widget:

```
from django-numeric import NumberResponse
from django-numeric import register

def total_users(user):
    """Return total number of users."""
    user_count = User.objects.filter(is_active=True).count()
    return NumberResponse(user_count, 'Total number of users')

# register endpoint to django-numeric
register('total-users', total_users, NumberResponse)
```

Another example would be an endpoint that provides employee of the month:

```
from django-numeric import LabelResponse
from django-numeric import register

def calculate_eom(user):
    """Return employee of the month."""
    user = User.objects.by_month().order('-success_rate')[0]
    return LabelResponse(user.username, 'Employee Of The Month')

# register endpoint to django-numeric
register('employee-of-the-month', calculate_eom, LabelResponse,
        cache_timeout=1*24*60*60)
```



---

## **Widgets**

---

For now two widgets are supported. Since I did not bought the rest of the custom json widgets, I did not implemented the wrappers for them. If you have them, feel free to contribute.

### **8.1 Label from JSON data widget**

This widget show a string on dashboard.

Endpoints that is implemented for this dashboard should return `djangonumerics.LabelResponse` object.

### **8.2 Number from JSON data widget**

This widget shows a number on dashboard.

Endpoints that is implemnted for this dashboard should return `djangonumerics.NumberResponse` object.



---

## **Authentication**

---

Since numerics dashboard does not provide any authentication method, django-numerics also do not have any authentication. So, either solve the security problem on network level or make sure that you are not sharing any sensitive data through numerics.

To at least provide a minimum privacy, django-numerics creates different urls for every user. I recommend using CryptoSerializer to generate endpoint urls. That way created endpoints will be near impossible to guess. But users that have the url, will be able to reach the endpoints. Please see serializers section to learn how to generate different type of endpoint urls.



---

## Permission

---

User permissions are decided by `permission_func` argument of `register` function. If a user has permission to reach an endpoint, endpoint link will be available `django-numeric` index page. Otherwise user will not have a link for that endpoint.

By default every registered endpoint is available for every user. But anonymous users do not have permission to reach endpoints. This behavior is provided by default `permission_func` which is following:

```
def grant_access(user, endpoint):
    """default permission function for endpoints."""
    return not user.is_anonymous()
```

This behavior can be changed by providing custom permission functions. Lets change *number of users* endpoint example to support anonymous users:

```
from djangonumeric import NumberResponse
from djangonumeric import register

def total_users(user):
    """Return total number of users."""
    user_count = User.objects.filter(is_active=True).count()
    return NumberResponse(user_count, 'Total number of users')

def grant_all(user):
    """Grant access to everybody."""
    return True

# register endpoint to django-numeric
# we are providing a new permission function to grant access to everybody.
register('total-users', total_users, NumberResponse,
        permission_func=grant_all)
```

Lets also change *employee of the month* example to grant access for only certain number of users.

```
from djangonumeric import LabelResponse
from djangonumeric import register
from django.conf import settings

def calculate_eom(user):
    """Return employee of the month."""
```

```
user = User.objects.by_month().order('-success_rate')[0]
return LabelResponse(user.username, 'Employee Of The Month')

def private_access(user):
    """Grant access for only given usernames on settings.

    Make sure that GRANTED_USERNAMES are in settings file.
    """
    username_access_list = settings.GRANTED_USERNAMES
    return user.username in username_access_list

# register endpoint to django-numeric
register('employee-of-the-month', calculate_eom, LabelResponse,
        cache_timeout=1*24*60*60, permission_func=private_access)
```

---

## Serializers

---

To create an endpoint url, user and endpoint is serialized and unique url is created for each user, endpoint tuple. Created url structure can be change by changing the serializer from settings. Serializer is changed from `DJANGO_NUMERICS_SERIALIZER_BACKEND`. By default this value is set to `django_numerics.serializers.DebugSerializer` which creates very readable serializer. I recommend to change this to `CryptoSerializer` to get better urls. Following are serializers that is provided by `django_numerics` by default.

### 11.1 `django_numerics.serializers.DebugSerializer` (default)

This serializer creates easy to read url structures for endpoints.

For total-users examples that was given earlier, this endpoint creates following url for user *huseyin*. Each user will have their own username on url.

```
http://localhost:8000/numerics/huseyin/total-users
```

This serializer is very helpful for development and has no dependency to external packages.

### 11.2 `django_numerics.serializers.BasicSerializer`

This serializer creates more cryptic urls. It uses user's database id and md5 of endpoint name to form a url. Again for the same user and endpoint this serializer creates following url:

```
http://localhost:8000/numerics/21/30495cd73bfabef15d781e531d4f9685
```

This can be used on production for small projects that do not want to add extra dependencies to project and if project give permission only small number of users. If everybody has access please use `CryptoSerializer` since people can access other people's endpoint by changes user id from url.

### 11.3 `django_numerics.serializers.CryptoSerializer`

This is most "secure" serializer. It creates cryptic urls. For the same endpoint and user following url will be created:

```
http://localhost:8000/numerics/gAAABBBVO-Z_U8bedooJKqXSW_eN-EzpCWJZQBQD9tn22UyyJDSxhRj7BDgk39PS2s149
```

Only downside for this serializer is, it has dependency to cryptography package which can be installed by

```
pip install cryptography
```