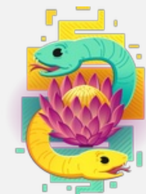
DJANGO MULTI-TENANT

BY: SHAUN DE PONTE

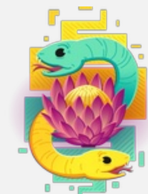
OCTOBER, 2024



PyConZA

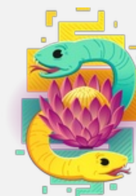
ABOUT ME

- Senior Data / Software Engineer at Calybre.
- MSc in Business Analytics from Aston University.
- I build things in my spare time.
- Co-organiser for PyData Johannesburg.



PyConZA

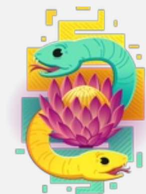
WHAT IS A TENANT?



PyConZA

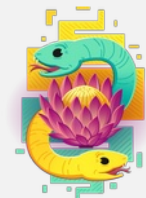
WHAT IS A TENANT

A Tenant is the customer or user of the application, operating within their own isolated portion or instance of the software.



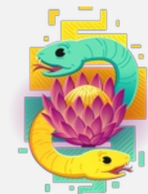
PyConZA

TYPES OF TENANT STRUCTURES



TYPES

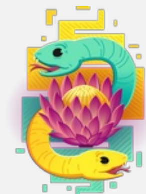
- Shared database with shared schema.
- Shared database with isolated schema.
- Isolated database with a shared app server.



PyConZA

DEMO

USING DJANGO’S ADMIN ONLY

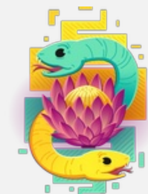


PyConZA

JAZZMIN

Jazzmin, intended as a drop-in app to jazz up your Django Admin Site, with plenty of things you can easily customise, including a built-in UI customizer

- Drop-in admin skin, all configuration optional
- Select2 drop-downs
- Bootstrap 4 & AdminLTE UI components
- Search bar for any given model admin
- Modal windows instead of popups
- Customisable side menu
- Customisable top menu
- Customisable user menu
- Responsive
- Customisable UI (via Live UI changes, or custom CSS/JS)
- Based on the latest adminlte + bootstrap



PyConZA

JAZZMIN

Polls

test@test.com

Dashboard

Polls
- Choices
- Polls
- Votes
- Make Messages

Administration
- Log entries

Authentication and Authorization
- Groups
- Users

Home | Support | Users | Polls

Search Users...

dashboard

Home / Dashboard

Polls

Choices	Add	Change
Polls	Add	Change
Votes	Add	Change
Make Messages	Go	

Authentication and Authorization

Groups	Add	Change
Users	Add	Change

Administration

Log entries	Add	Change

Recent actions

test2@test.com — 1 day, 17 hours Ago
Unknown content
Changed "test2@test.com" — Changed User permissions.

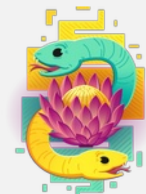
test2@test.com — 1 day, 17 hours Ago
Unknown content
Changed "test2@test.com" — Changed User permissions and Last login.

test2@test.com — 1 day, 17 hours Ago
Unknown content
Changed "test2@test.com" — Changed password.

Care week society young. — 1 week, 1 day Ago
Unknown content
Deleted "Care week society young.."

test2@test.com — 2 weeks, 6 days Ago
Unknown content
Changed "test2@test.com" — Changed User permissions.

test2@test.com — 2 weeks, 6 days Ago
Unknown content

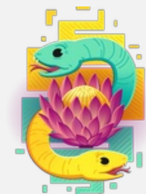


PyConZA

TENANT MODEL

system_management > models.py

```
class Tenant(models.Model):
    """
    Model representing a tenant with unique identification and associated details.

    Methods:
        __str__():
            Returns the tenant's subdomain as its string representation.

    Meta:
        verbose_name_plural (str): The plural form of the model's name is 'Instance'.
    """
    id = models.UUIDField(primary_key=True, default=uuid.uuid4, editable=False)
    name = models.CharField(max_length=255, default='Acme')
    address = models.CharField(max_length=255, default='Acme Address')
    telephone = models.CharField(max_length=255, default='9999')
    email = models.CharField(max_length=255, default='acme@explosive.com')
    image = models.ImageField(upload_to='images/', null=True, blank=True)
    subdomain = models.CharField(max_length=255, default='acme')

    created = models.DateTimeField(editable=False, auto_now_add=True)
    date_modified = models.DateTimeField(null=True, editable=False, auto_now=True)

    def __str__(self):
        return self.subdomain

    class Meta:
        verbose_name_plural = 'Instance'
```

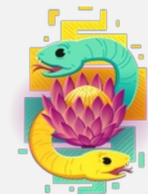


PyConZA

TENANT AWARE MODEL

system_management > models.py

```
class TenantAwareModel(models.Model):
    """
    Abstract model for tenant-aware objects that associate records with a specific tenant.

    Fields:
        tenant_aware_id (UUIDField): Unique identifier for the tenant-aware object, automatically generated.
        tenant (ForeignKey): Foreign key linking the object to a specific Tenant instance. When the linked
            Tenant is deleted, the related records will also be deleted (on_delete=models.CASCADE).

    Meta:
        verbose_name_plural (str): The plural form of the model's name is 'TenantAwareModel'.
    """

    tenant_aware_id = models.UUIDField(primary_key=True, default=uuid.uuid4, editable=False)
    tenant = models.ForeignKey(Tenant, on_delete=models.CASCADE)

    class Meta:
        verbose_name_plural = 'TenantAwareModel'
```

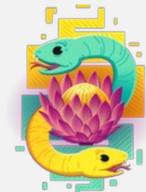


PyConZA

CUSTOM USER

system_management > models.py

```python
class CustomUser(AbstractBaseUser, PermissionsMixin):

    id = models.UUIDField(primary_key=True, default=uuid.uuid4, editable=False)
    email = LowercaseEmailField(unique=True)
    first_name = models.CharField(max_length=150, blank=True)
    last_name = models.CharField(max_length=100, blank=True)
    phone = models.CharField(max_length=500, blank=True, default=0)
    is_confirmed = models.BooleanField(default=False)
    is_staff = models.BooleanField(_('staff status'), default=True,
                                   help_text=_('Designates whether the user can log into this site.'), )
    is_active = models.BooleanField(_('active'), default=True, help_text=_(
        [illegible] of deleting accounts.'), )
    tenant = models.ForeignKey(Tenant, on_delete=models.CASCADE, null=True, blank=True)

    created = models.DateTimeField(editable=False, default=datetime.now)
    # history = HistoricalRecords()

    USERNAME_FIELD = 'email'
    REQUIRED_FIELDS = ['first_name', 'last_name']
    objects = CustomUserManager()

    def __str__(self):
        return self.email

    def get_full_name(self):
        return self.email

    def get_short_name(self):
        return self.email

    def has_perm(self, perm, obj=None):
        "Does the user have a specific permission?"
        # Simplest possible answer: Yes, always
        return True

    def has_module_perms(self, app_label):
        "Does the user have permissions to view the app `app_label`?"
        # Simplest possible answer: Yes, always
        return True

    def save(self, *args, **kwargs):
        if not self.pk and not self.tenant_id:
            self.tenant = self._state.adding and self.tenant or None
        super().save(*args, **kwargs)


    class Meta:
        verbose_name = _('Users')
        verbose_name_plural = _('Users')
```



PyConZA

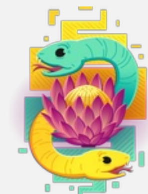
MIDDLEWARE

system_management > middleware.py

```python
class TenantMiddleware(MiddlewareMixin):
    """
    Middleware to process and assign the tenant based on the subdomain in the request's host.

    Methods:
        process_request(request):
            Extracts the subdomain from the request's HTTP host and attempts to find the corresponding tenant.
            If a subdomain is present and valid, the tenant is assigned to the request. If no valid subdomain
            is found or the tenant is not found, assigns None to the request's tenant.

    Args:
        request (HttpRequest): The incoming request object.

    Attributes:
        request.tenant (Tenant or None): The tenant object corresponding to the subdomain, or None if no
        subdomain is found or the tenant does not exist.
    """

    def process_request(self, request):
        # Extract the host parts
        host_parts = request.META['HTTP_HOST'].split('.')

        # Determine the subdomain
        if len(host_parts) > 2 and host_parts[0] != 'www':
            subdomain = host_parts[0]
        else:
            subdomain = None

        # Logic to handle subdomain or default to example.com
        if subdomain is None:
            request.tenant = None
        else:
            try:
                request.tenant = get_object_or_404(Tenant, subdomain=subdomain)
            except Http404:
                request.tenant = None  # Handle case where no tenant is found
```

settings.py

```python
MIDDLEWARE = [
    'django.middleware.security.SecurityMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.common.CommonMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
    'django.middleware.clickjacking.XFrameOptionsMiddleware',
    'system_management.middleware.TenantMiddleware',
]
```

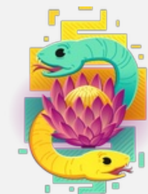


PyConZA

AUTHENTICATION

system_management > authentication.py

```python
class TenantBackend(ModelBackend):

    def authenticate(self, request, username=None, password=None, **kwargs):
        """
        Authenticate a user based on email and password within the context of a tenant.

        Args:
            request (HttpRequest): The current request object, which contains the tenant.
            username (str, optional): The email address of the user attempting to authenticate.
            password (str, optional): The password of the user attempting to authenticate.
            **kwargs: Additional keyword arguments.

        Returns:
            User or None: The authenticated user object if authentication is successful;
            otherwise, None.
        """

        UserModel = get_user_model()
        tenant = getattr(request, 'tenant', None)
        if tenant is None:
            return None
        try:
            user = UserModel.objects.get(email=username, tenant=tenant)
        except UserModel.DoesNotExist:
            return None
        if user.check_password(password) and self.user_can_authenticate(user):
            return user
        return None
```

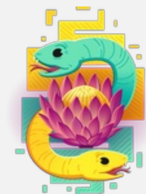


PyConZA

CUSTOM ADMIN SITE

system_management > custom_admin.py

```
from django.contrib.admin import AdminSite
from django.contrib.auth.models import Group

class TenantAdminSite(AdminSite):
    """
    Custom admin site for tenant-based administration.

    Methods:
        has_permission(request):
            Determines whether the current user has access to the admin site.

        get_queryset(request):
            Returns the queryset filtered by the tenant associated with the request,
            if a tenant is present. Otherwise, returns the default queryset.

    Attributes:
        tenant_admin_site (TenantAdminSite): Instance of the custom tenant admin site with
        the name 'tenant_admin'.
    """

    def has_permission(self, request):
        return request.user.is_active

    def get_queryset(self, request):
        queryset = super().get_queryset(request)
        if hasattr(request, 'tenant'):
            return queryset.filter(tenant=request.tenant)
        return queryset

tenant_admin_site = TenantAdminSite(name='tenant_admin')
```

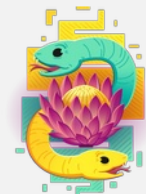


PyConZA

VIEWS

system_management > views.py

```python
class TenantLoginView(LoginView):

    """
    Custom login view for tenant-specific authentication.

    Attributes:
        form_class (CustomAuthenticationForm): The custom authentication form used for login.

    Methods:
        form_valid(form):
            Validates the user and ensures the tenant from the login form matches the tenant in the request.
            If the tenant does not match, an error is added, and the form is marked as invalid.
            On success, logs the user in and redirects to the success URL.

        form_invalid(form):
            Renders the login form again with validation errors if the form is invalid.

    Views:
        access_denied(request):
            Renders the 'access_denied.html' template when access is denied.

        index(request):
            Renders the landing page (home.html) for the site.
    """
    form_class = CustomAuthenticationForm

    def form_valid(self, form):
        user = form.get_user()
        if user.tenant != self.request.tenant:
            form.add_error(None, "Invalid login for this tenant.")
            return self.form_invalid(form)
        login(self.request, user)
        return redirect(self.get_success_url())

    def form_invalid(self, form):
        return render(self.request, 'registration/login.html', {'form': form})
```

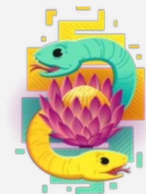


PyConZA

REGISTER APP MODEL

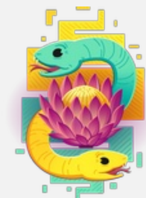
system_management > admin.py

```python
    def get_queryset(self, request):
        qs = super().get_queryset(request)
        if hasattr(request, 'tenant'):
            return qs.filter(tenant=request.tenant)
        return qs


    def save_model(self, request, obj, form, change):
        if hasattr(request, 'tenant'):
            obj.tenant = request.tenant
        super().save_model(request, obj, form, change)


    def get_form(self, request, obj=None, **kwargs):
        # Set the tenant attribute when the form is requested
        self.tenant = request.tenant
        return super().get_form(request, obj, **kwargs)

tenant_admin_site.register(Customer, CustomerAdmin)
```



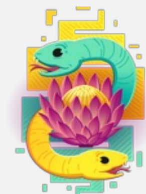
PyConZA

MORE DEMO’S



PyConZA

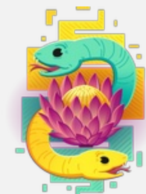
USE CASES



PyConZA

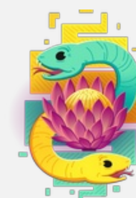
USE CASES

- Quick proof of concept.
- Building internal apps for your org.
- Scalable into a robust multi-tenant app.
- Database agnostic, easy to setup and deploy, no dependencies.



PyConZA

THANK YOU



PyConZA