Django 1.11

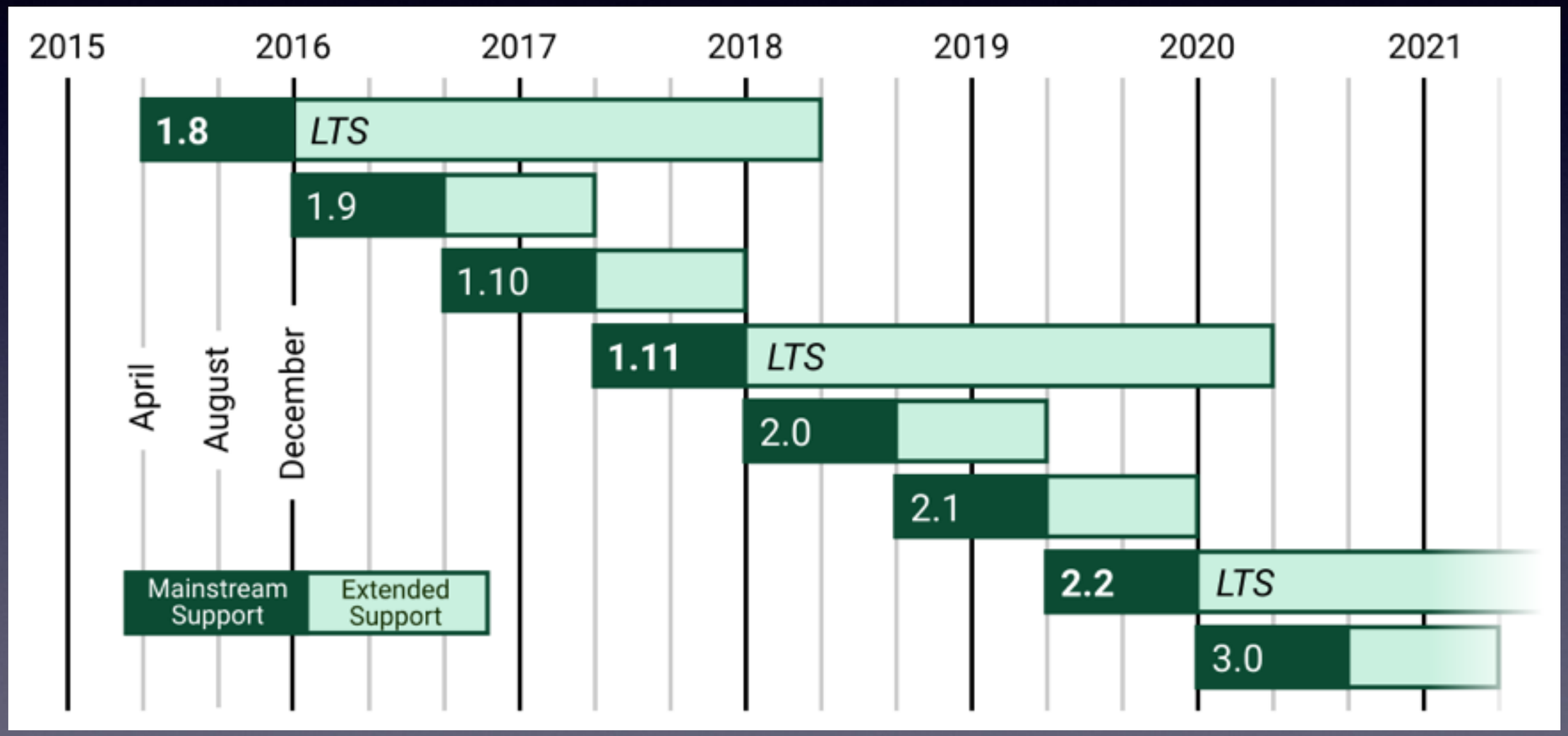
Django User Group Hamburg 10.05.2017

Arne Brodowski
www.arnebrodowski.de

Motivation to use 1.11

- First release with Python 3.6 support
- Last release with Python 2 support (2.7 only)
  - Django 2.0 will only support Python 3.5+
- LTS (Long Term Support) release

Release Timeline



What's new?

- Class-based model indexes for creating database indexes
- Template-based widget rendering to ease customizing form widgets
- Subquery expressions to create explicit subqueries using the ORM

Class-based model indexes

- Indexes are added to models using the Meta.indexes option.
- django.db.models.indexes module contains classes which ease creating database indexes
- Index options: fields, name
  - index names cannot be longer than 30 characters and shouldn’t start with a number (0-9) or underscore (_), default: auto-generated

Class-based model indexes

- The models.Index class creates a b-tree index, as if you used db_index on the model field or index_together on the model Meta class.
- It can be subclassed to support different index types, such as GinIndex.
- It also allows defining the order (ASC/DESC) for the columns of the index.
  - ASC: 'field_name' / DESC: '-field_name'

Class-based model indexes

```
class Customer(models.Model):
    first_name = models.CharField(max_length=100)
    last_name = models.CharField(max_length=100)

    class Meta:
        indexes = [
            models.Index(fields=['last_name', 'first_name']),
            models.Index(
                fields=['first_name'],
                name='first_name_idx'
            ),
        ]
```

Template-based widget rendering

- Django's form widgets are rendered using Django's template engines system.
- Widgets can specify custom template names.
- Forms and widgets can specify custom renderer classes.
- A widget's template can be overridden by a project.

Template-based widget rendering

- Built-in-template form renderers
  - "DjangoTemplates": Standalone template engine with it's own settings
  - "Jinja2": Same as above, but for Jinja2 templates
  - "TemplatesSetting": uses settings.TEMPLATES

Template-based widget rendering

- Context available in widget templates
  - Widget.get_context()
    - widget = {"name", "value", "attrs", "is_hidden", "template_name"}

Template-based widget rendering

django/forms/templates/django/forms/widgets/input.html

```
<input
    type="{{ widget.type }}"
    name="{{ widget.name }}"
    {% if widget.value != None and widget.value != "" %}
        value="{{ widget.value }}"
    {% endif %}
{% include "django/forms/widgets/attrs.html" %}
/>
```

Template-based widget rendering

- View all built-in widget templates:
  - https://github.com/django/django/tree/master/django/forms/templates/django/forms/widgets
  - https://github.com/django/django/tree/master/django/contrib/admin/templates/admin/widgets

Template-based widget rendering

- Backwards incompatible
    - Some undocumented classes in django.forms.widgets are removed: SubWidget, RendererMixin, ChoiceFieldRenderer, RadioFieldRenderer, CheckboxFieldRenderer, ChoiceInput, RadioChoiceInput, CheckboxChoiceInput
    - The Widget.format_output() method is removed. Use a custom widget template instead.

Template-based widget rendering

- Backwards incompatible
  - Some widget values, such as <select> options, are now localized if settings.USE_L10N=True. You could revert to the old behavior with custom widget templates that uses the localize template tag to turn off localization.

Subquery expressions

- The new **Subquery** and **Exists** database expressions allow creating explicit subqueries.
- Subqueries may refer to fields from the outer queryset using the **OuterRef** class.

Subquery expressions

Example: Annotate each post with the email address of the author of the newest comment on that post:

```
from django.db.models import OuterRef, Subquery

newest = Comment.objects.filter(
    post=OuterRef('pk')
).order_by('-created_at')

Post.objects.annotate(
    newest_commenter_email=Subquery(
        newest.values('email')[:1]
    )
)
```

Subquery expressions

Example: Annotate each post with the email address of the author of the newest comment on that post:

```
SELECT "post"."id", (
    SELECT U0."email"
    FROM "comment" U0
    WHERE U0."post_id" = ("post"."id")
    ORDER BY U0."created_at" DESC LIMIT 1
) AS "newest_commenter_email" FROM "post"
```

Subquery expressions

- OuterRef can be used with nested Subquery instances.

```
Book.objects.filter(id=OuterRef(OuterRef('pk')))
```

Subquery expressions

- Limiting the subquery to a single row

```
subquery = Subquery(newest.values('email')[:1])
```

- Limiting a subquery to a single column

```
Comment.objects.filter(
    post__in=Subquery(posts.values('pk'))
)
```

Minor changes

Serialization

- The encoder used by the JSON serializer can now be customized by passing a cls keyword argument to the serializers.serialize() function.

- DjangoJSONEncoder now serializes timedelta objects (used by DurationField).

Minor changes

django.contrib.postgres

- JSONField accepts a new encoder parameter to specify a custom class to encode data types not supported by the standard encoder.

Minor changes

Request & Response

- CommonMiddleware now sets the Content-Length response header for non-streaming responses. Was done by ConditionalGetMiddleware before.

- ConditionalGetMiddleware now adds the ETag header to responses.

Minor changes

Templates

- mark_safe() can now be used as a decorator.

- The Jinja2 template backend now supports context processors by setting the 'context_processors' option in OPTIONS.

- The regroup tag now returns namedtuples instead of dictionaries so you can unpack the group object directly in a loop, e.g. {% for grouper, list in regrouped %}.

Minor changes

Tests

- Added the test --debug-mode option to help troubleshoot test failures by setting the DEBUG setting to True.

- DiscoverRunner now runs the system checks at the start of a test run. Override the DiscoverRunner.run_checks() method if you want to disable that.

Minor changes

Migrations

- Added support for serialization of uuid.UUID objects.

Minor changes

CSRF

- Added the CSRF_USE_SESSIONS setting to allow storing the CSRF token in the user's session rather than in a cookie.

Minor changes

Memcached

- Memcached backends now pass the contents of OPTIONS as keyword arguments to the client constructors.

- Memcached backends now allow defining multiple servers as a comma-delimited string in LOCATION, for convenience with third-party services that use such strings in environment variables.

Minor changes

Database Backends

- QuerySet.iterator() now uses server-side cursors on PostgreSQL.

- Added the skip_locked argument to QuerySet.select_for_update() on PostgreSQL 9.5+ and Oracle to execute queries with FOR UPDATE SKIP LOCKED.

- Added MySQL support for the 'isolation_level' option in OPTIONS to allow specifying the transaction isolation level.

Minor changes

Models / Querysets

- ImageFieId now has a default validate_image_file_extension validator.
- Added support for expressions in QuerySet.values() and values_list().
- Added the nulls_first and nulls_last parameters to Expression.asc() and desc() to control the ordering of null values.

Minor changes

Models / Querysets

- Added QuerySet.union(), intersection(), and difference().
  - Uses SQL's UNION, INTERSECT or EXCEPT operator
  - return model instances of the type of the first QuerySet even if the arguments are QuerySets of other models. Passing different models works as long as the SELECT list is the same in all QuerySets (at least the types, the names don't matter as long as the types in the same order).

BACKWARDS INCOMPATIBLE

- django.template.backends.django.Template.render() prohibits non-dict context
- In model forms, CharField with null=True now saves NULL for blank values instead of empty strings.
- If you subclass AbstractUser and override clean(), be sure it calls super().
- The undocumented DateTimeAwareJSONEncoder alias for DjangoJSONEncoder (renamed in Django 1.0) is removed.

BACKWARDS INCOMPATIBLE

- The cached template loader is now enabled if DEBUG is False and OPTIONS['loaders'] isn’t specified. This could be backwards-incompatible if you have some template tags that aren’t thread safe.

- The prompt for stale content type deletion no longer occurs after running the migrate command. Use the new remove_stale_contenttypes command instead.

- get_model() and get_models() now raise AppRegistryNotReady if they’re called before models of all applications have been loaded. Previously they only required the target application’s models to be loaded. If you need the old behavior of get_model(), set the require_ready argument to False.

Django 1.11

- Full release notes:
    - https://docs.djangoproject.com/en/1.11/releases/1.11/