# At end self made easy notes

# Video 47

# Code 39a

# 1) Dynamic URLs (path converters)

Django’s path() lets you capture parts of the URL and pass them as **typed** arguments to your view.

## Built-in converters

* str (default): any non-slash text → path('x/<str:slug>/', ...)
* int: digits only → path('x/<int:pk>/', ...)
* slug: letters, numbers, hyphen, underscore → path('x/<slug:title>/', ...)
* uuid: UUID strings → path('x/<uuid:uid>/', ...)
* path: **including slashes** (greedy) → path('x/<path:subpath>/', ...)

If you **omit** a converter (e.g., <my\_id>), Django treats it as **str**.

## A. Single dynamic segment

### urls.py

# student/urls.py

from django.urls import path

from student.views import home, profile

urlpatterns = [

    path('', home, name='home'),  # Home page

    # Default (str): <my\_id> is a string

    path('profile/<my\_id>/', profile, name='profile\_str'),

    # Explicit int converter: raises 404 if non-digits are passed

    path('profile/<int:my\_id>/', profile, name='profile\_int'),

    # Slug: for SEO-friendly titles

    path('profile/<slug:title>/', profile, name='profile\_slug'),

]

### views.py

# student/views.py

from django.shortcuts import render

def home(request):

    return render(request, 'student/home.html')

def profile(request, \*\*kwargs):

    """

    This single view can accept different parameters depending on the URL:

    - my\_id (str or int)

    - title (slug)

    We'll just echo back what we received.

    """

    context = kwargs  # e.g., {'my\_id': 5} or {'title': 'master-django-in-10'}

    return render(request, 'student/profile.html', context)

### templates/student/home.html

{% load static %}

<!doctype html>

<html>

<head><meta charset="utf-8"><title>Home page</title></head>

<body>

  <h1>Home page</h1>

  <ul>

    <!-- Using the int converter route -->

    <li><a href="{% url 'profile\_int' 1 %}">Student 1</a></li>

    <li><a href="{% url 'profile\_int' 2 %}">Student 2</a></li>

    <!-- Using named args also works -->

    <li><a href="{% url 'profile\_int' my\_id=3 %}">Student 3</a></li>

    <!-- Slug route example -->

    <li><a href="{% url 'profile\_slug' title='master-django-in-10' %}">

      Article: master-django-in-10

    </a></li>

  </ul>

</body>

</html>

### templates/student/profile.html

<!doctype html>

<html>

<head><meta charset="utf-8"><title>Profile page</title></head>

<body>

  <h1>Profile page</h1>

  <!-- We print whatever the view received -->

  {% if my\_id %}

    <h2>Here is student {{ my\_id }}</h2>

  {% endif %}

  {% if title %}

    <h2>Requested slug: {{ title }}</h2>

  {% endif %}

</body>

</html>

## B. Multiple dynamic segments

### urls.py

# student/urls.py

from django.urls import path

from student.views import profile\_two

urlpatterns = [

    # e.g. /student/profile/5/10/

    path('profile/<int:my\_class>/<int:my\_id>/', profile\_two, name='profile\_two'),

]

### views.py

# student/views.py

from django.shortcuts import render

def profile\_two(request, my\_class, my\_id):

    # Django already converts <int:...> to Python int

    context = {'class': my\_class, 'id': my\_id}

    return render(request, 'student/profile.html', context)

### Linking to it

<li><a href="{% url 'profile\_two' 5 10 %}">Class 5, Student 10</a></li>

<!-- or named arguments -->

<li><a href="{% url 'profile\_two' my\_class=3 my\_id=2 %}">Class 3, Student 2</a></li>

**Important:** The **order** of URL patterns matters. Put more specific patterns **before** more general ones to avoid shadowing.

# 2) Custom path converters

When built-in converters aren’t enough, define your own.

## Steps

1. Create a converter class with:
   * regex (string): used to match the path segment
   * to\_python(self, value): string → Python object (runs when resolving the URL)
   * to\_url(self, value): Python object → string (used by reverse()/{% url %})
2. Register it with register\_converter(converter\_class, "name")
3. Use it in path() like <name:param>.

## A. Example: 4-digit year converter

### student/converters.py

# student/converters.py

class FourDigitYearConverter:

    # Match exactly four digits

    regex = r'[0-9]{4}'

    def to\_python(self, value):

        """

        Convert the matched string to Python type.

        If you raise ValueError here, Django will return 404.

        """

        return int(value)

    def to\_url(self, value):

        """

        Called when reversing URLs. Must return str.

        If you want zero-padding, use :04d. For year, plain str is fine.

        """

        return f"{int(value)}"

Note: In your original snippet you used f"{value:4d}" which pads with **spaces**. For URLs this is unsafe. Prefer f"{value:04d}" for zero-padding or just str(int(value)).

### student/urls.py

# student/urls.py

from django.urls import path, register\_converter

from student.views import year\_profile

from student.converters import FourDigitYearConverter

# Register the converter once at import time

register\_converter(FourDigitYearConverter, 'yyyy')

urlpatterns = [

    # /student/profile/2003/

    path('profile/<yyyy:year>/', year\_profile, name='profile\_year'),

]

### student/views.py

# student/views.py

from django.shortcuts import render

def year\_profile(request, year):

    # 'year' is already an int thanks to to\_python()

    return render(request, 'student/profile.html', {'year': year})

### Link / reverse examples

<!-- templates -->

<a href="{% url 'profile\_year' 2003 %}">Year 2003</a>

<a href="{% url 'profile\_year' year=2024 %}">Year 2024</a>

# in Python

from django.urls import reverse

reverse('profile\_year', args=[2003])        # '/student/profile/2003/'

reverse('profile\_year', kwargs={'year': 2024})

# 3) Extra tips and gotchas

* **Trailing slashes**: By convention, Django uses a trailing slash: path('profile/<int:id>/', ...). Be consistent or enable APPEND\_SLASH=True.
* **404 vs. 500**: If a converter’s regex matches but to\_python raises ValueError, Django returns **404**. Any other exception bubbles up as a server error.
* **Type safety**: With converters (int, uuid, custom), the view receives properly typed values—no need to cast.
* **Query string**: /profile/5/?tab=marks → tab is accessed via request.GET, **not** captured by path().
* **Slug details**: Django’s slug converter matches ASCII letters, digits, hyphens -, and underscores \_.
* **Greedy catch-all**: Use path: if you need to capture subpaths with slashes: path('files/<path:subpath>/', ...).
* **Optional segments**: path() can’t do optional parts directly. Use two patterns or re\_path() for advanced regex alternatives—**or** write a custom converter that accepts empty string and handle default in view.
* **Namespaces**: When including app URLs, set app\_name = 'student' in student/urls.py and include as path('student/', include('student.urls', namespace='student')). Then reverse as student:profile\_int.
* **Order matters**: Put **static** routes first (e.g., profile/create/) before dynamic captures like profile/<int:id>/, otherwise the dynamic one can swallow the static path.
* **Reversing safely**: Prefer {% url 'name' ... %} and reverse('name', ...) over hard-coding URLs.

# 4) Your examples rewritten “as-is” with comments

### student/urls.py

from django.urls import path, register\_converter

from student.views import home, profile, profile\_two, year\_profile

from student.converters import FourDigitYearConverter

# Register our custom converter as 'yyyy'

register\_converter(FourDigitYearConverter, 'yyyy')

urlpatterns = [

    path('', home, name='home'),

    # Default str (same as <str:my\_id>) — shows how Django treats it by default

    path('profile/<my\_id>/', profile, name='profile\_default\_str'),

    # Explicit int

    path('profile/<int:my\_id>/', profile, name='profile\_int'),

    # Slug for SEO-friendly titles

    path('profile/<slug:title>/', profile, name='profile\_slug'),

    # Two params (class and id), both ints

    path('profile/<int:my\_class>/<int:my\_id>/', profile\_two, name='profile\_two'),

    # Custom year converter. Matches exactly four digits and converts to int

    path('profile/<yyyy:year>/', year\_profile, name='profile\_year'),

]

student/views.py

from django.shortcuts import render

def home(request):

    # Simple landing page

    return render(request, 'student/home.html')

def profile(request, \*\*kwargs):

    """

    Handles routes:

      - profile/<my\_id>/   (str)

      - profile/<int:my\_id>/

      - profile/<slug:title>/

    We rely on the kwargs to print what we got.

    """

    return render(request, 'student/profile.html', kwargs)

def profile\_two(request, my\_class, my\_id):

    """

    Handles: profile/<int:my\_class>/<int:my\_id>/

    Both are already ints.

    """

    return render(request, 'student/profile.html', {'class': my\_class, 'id': my\_id})

def year\_profile(request, year):

    """

    Handles: profile/<yyyy:year>/

    'year' is an int thanks to our converter's to\_python().

    """

    return render(request, 'student/profile.html', {'year': year})

### student/converters.py

class FourDigitYearConverter:

    # Only match 4 digits

    regex = r'[0-9]{4}'

    def to\_python(self, value):

        # Convert '2003' -> 2003 (int). Raise ValueError for 404 behavior if needed.

        return int(value)

    def to\_url(self, value):

        # Convert Python -> path segment. Must be a string.

        # Avoid spaces; if you wanted zero-pad use f"{int(value):04d}"

        return str(int(value))

### templates/student/home.html

<!doctype html>

<html>

<head><meta charset="utf-8"><title>Home page</title></head>

<body>

  <h1>Home page</h1>

  <ul>

    <!-- Default str route -->

    <li><a href="{% url 'profile\_default\_str' 'abc' %}">Profile 'abc' (str)</a></li>

    <!-- int route -->

    <li><a href="{% url 'profile\_int' 1 %}">Student 1</a></li>

    <li><a href="{% url 'profile\_int' my\_id=2 %}">Student 2</a></li>

    <!-- slug route -->

    <li><a href="{% url 'profile\_slug' title='master-django-in-10' %}">Slug: master-django-in-10</a></li>

    <!-- two-parameter route -->

    <li><a href="{% url 'profile\_two' 5 10 %}">Class 5, Student 10</a></li>

    <li><a href="{% url 'profile\_two' my\_class=3 my\_id=2 %}">Class 3, Student 2</a></li>

    <!-- custom converter route -->

    <li><a href="{% url 'profile\_year' 2003 %}">Year 2003</a></li>

  </ul>

</body>

</html>

templates/student/profile.html

<!doctype html>

<html>

<head><meta charset="utf-8"><title>Profile page</title></head>

<body>

  <h1>Profile page</h1>

  {% if id %}<h2>Student id: {{ id }}</h2>{% endif %}

  {% if class %}<h2>Class: {{ class }}</h2>{% endif %}

  {% if my\_id %}<h2>my\_id: {{ my\_id }}</h2>{% endif %}

  {% if title %}<h2>Slug title: {{ title }}</h2>{% endif %}

  {% if year %}<h2>Year: {{ year }}</h2>{% endif %}

</body>

</html>

# 5) Quick revision bullets

* <param> defaults to **str**. Use <int:>, <slug:>, etc. for strict typing.
* Multiple parameters? Just add more segments: path('a/<int:x>/b/<slug:y>/').
* Template url tag can pass **positional** or **named** args.
* **Custom converters**: define regex, to\_python, to\_url, then register\_converter().
* Wrong type → **404**, not 500.
* Use path: converter to capture **slashes** inside one segment.
* Use **namespaces** to avoid URL name collisions across apps.
* Put **specific patterns first**.

Self:

## 1. Basic Dynamic URLs with Built-in Path Converters

In **urls.py**:

from django.urls import path

from student.views import home, profile

urlpatterns = [

    path('', home, name='home'),

    path('profile/<my\_id>', profile, name='profile'),

    # Default treats my\_id as str

    # Better to be explicit:

    path('profile/<str:my\_id>', profile, name='profile'),

    # If expecting an integer:

    path('profile/<int:my\_id>', profile, name='profile'),

    # Using slug for URLs like “master-django-in-10”:

    path('profile/<slug:title>', profile, name='profile'),

]

In **views.py**:

from django.shortcuts import render

def home(request):

    return render(request, 'student/home.html')

def profile(request, my\_id):

    student = {'id': my\_id}

    return render(request, 'student/profile.html', student)

In **home.html**:

<h1>Home page</h1>

<ul>

  <li><a href="{% url 'profile' 1 %}">Student 1</a></li>

  ...

</ul>

In **profile.html**:

<h1>Profile page</h1>

<h2>Here are student {{ id }}</h2>

## 2. Multiple Parameters in URLs

Combine two converters in one path to capture more detail:

# urls.py

urlpatterns = [

    path('profile/<int:my\_class>/<int:my\_id>/', profile, name='profile'),

]

# views.py

def profile(request, my\_class, my\_id):

    student = {'class': my\_class, 'id': my\_id}

    return render(request, 'student/profile.html', student)

And in your template:

<li><a href="{% url 'profile' 3 2 %}">Student 3</a></li>

This generates a URL like:

http://127.0.0.1:8000/student/profile/5/10/

## 3. **Custom Path Converter** Example: Four-Digit Year

When built-in converters aren’t enough, you can define your own.

In converters.py:

class FourDigitYearConverter:

    regex = "[0-9]{4}"

    def to\_python(self, value):

        return int(value)

    def to\_url(self, value):

        return "%04d" % value

### In urls.py:

from django.urls import path, register\_converter

from student.converters import FourDigitYearConverter

from student.views import profile

register\_converter(FourDigitYearConverter, 'yyyy')

urlpatterns = [

    path('profile/<yyyy:year>/', profile, name='profile'),

]

In views.py:

def profile(request, year):

    student = {'year': year}

    return render(request, 'student/profile.html', student)

Now you can visit:

http://127.0.0.1:8000/profile/2003/

## 4. **Why and How Custom Path Converters Work**

* **Purpose**: Converts let you define strict URL patterns and transform the captured value for use in views.
* **Core mechanics** (as documented by Django):
  + The class must define a regex attribute for matching.
  + to\_python(self, value) converts the matched string into a Python type (e.g., int).
  + to\_url(self, value) defines how to convert the Python type back into a string during URL reversing (e.g., used by reverse() or {% url %}).
* •   class FourDigitYearConverter:
* •       regex = "[0-9]{4}"
* •
* •       def to\_python(self, value):
* •           return int(value)
* •
* •       def to\_url(self, value):
* •           return "%04d" % value

Then register with:

register\_converter(FourDigitYearConverter, 'yyyy')

[Django Project](https://docs.djangoproject.com/fr/4.2/topics/http/urls/?utm_source=chatgpt.com)

* **Community Q&A** confirms:
  + regex defines valid matches (e.g., exactly four digits).
  + to\_python gets called first—if it raises ValueError, Django returns a 404.
  + to\_url defines how model values get formatted during reversing, e.g.:

“The converter will accept any 1-4 digit number (for less than 4 digit it will prepend leading zero(s))” [Stack Overflow](https://stackoverflow.com/questions/76842781/registering-custom-path-converters-in-django?utm_source=chatgpt.com)

* **Advanced usage tips**:
  + Clean, reusable converters: e.g., year and month converters:
* o   class TwoDigitMonthConverter:
* o       regex = "[0-9]{2}"
* o       def to\_python(self, value): return int(value)
* o       def to\_url(self, value): return "%02d" % value

[DjangoTricks](https://www.djangotricks.com/tricks/7gNEJU2GfCH9/?utm_source=chatgpt.com)

* + Model converters for lookups, e.g.:
* o   class QuestionConverter(IntConverter):
* o       def to\_python(self, value):
* o           return Question.objects.get(pk=int(value))
* o       def to\_url(self, obj):
* o           return str(obj.pk)

Then register and use <question:question> to pass the actual model instance to your view. [Considerate Code](https://consideratecode.com/2018/05/11/the-hidden-powers-of-custom-django-2-0-path-converters/?utm_source=chatgpt.com)

* + Using enums for strict string matching:
* o   class RoleConverter:
* o       regex = "chef|gourmand|foodie"
* o       def to\_python(self, value): return value
* o       def to\_url(self, value): return value

More advanced: use an Enum to convert into typed values inside the view. [adamj.eu](https://adamj.eu/tech/2025/08/01/django-custom-url-converter-string/?utm_source=chatgpt.com)

## 5. **Summary Table**

| **Feature** | **Syntax / Code Sample** |
| --- | --- |
| Default dynamic URL | path('profile/<int:my\_id>/', profile, name='profile') |
| Multiple parameters | path('profile/<int:my\_class>/<int:my\_id>/', profile, name='profile') |
| Custom path converter | Define converter class + register\_converter in urls.py |
| Built-in converters | str, int, slug, uuid, path [Django Project](https://docs.djangoproject.com/fr/4.2/topics/http/urls/?utm_source=chatgpt.com)[Hostinger](https://www.hostinger.com/tutorials/django-url-patterns?utm_source=chatgpt.com) |
| Custom converter methods | regex, to\_python(), to\_url() |
| Error handling | to\_python() can raise ValueError → 404 |
| Advanced use cases | Model instance conversion, enum-based routing [Considerate Code](https://consideratecode.com/2018/05/11/the-hidden-powers-of-custom-django-2-0-path-converters/?utm_source=chatgpt.com)[adamj.eu](https://adamj.eu/tech/2025/08/01/django-custom-url-converter-string/?utm_source=chatgpt.com) |

## 6. **Your Example, Polished**

Here’s your example, refined and complete:

# converters.py

class FourDigitYearConverter:

    regex = "[0-9]{4}"

    def to\_python(self, value):

        return int(value)

    def to\_url(self, value):

        return "%04d" % value

# urls.py

from django.urls import path, register\_converter

from student.converters import FourDigitYearConverter

from student.views import home, profile

register\_converter(FourDigitYearConverter, 'yyyy')

urlpatterns = [

    path('', home, name='home'),

    path('profile/<yyyy:year>/', profile, name='profile'),

]

# views.py

def profile(request, year):

    student = {'year': year}

    return render(request, 'student/profile.html', student)

<!-- profile.html -->

<h1>Profile page</h1>

<h2>Here are student from year {{ year }}</h2>