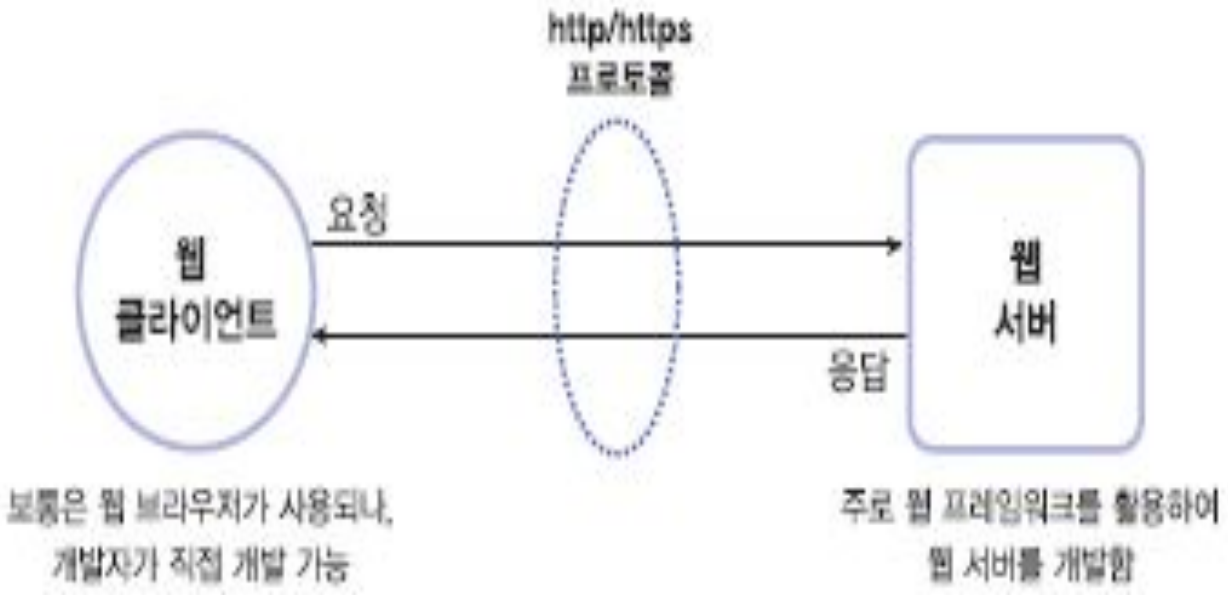
DJango

웹 프로그래밍의 이해

HTTP(S) 프로토콜로 통신하는, 클라이언트와 서버를 개발

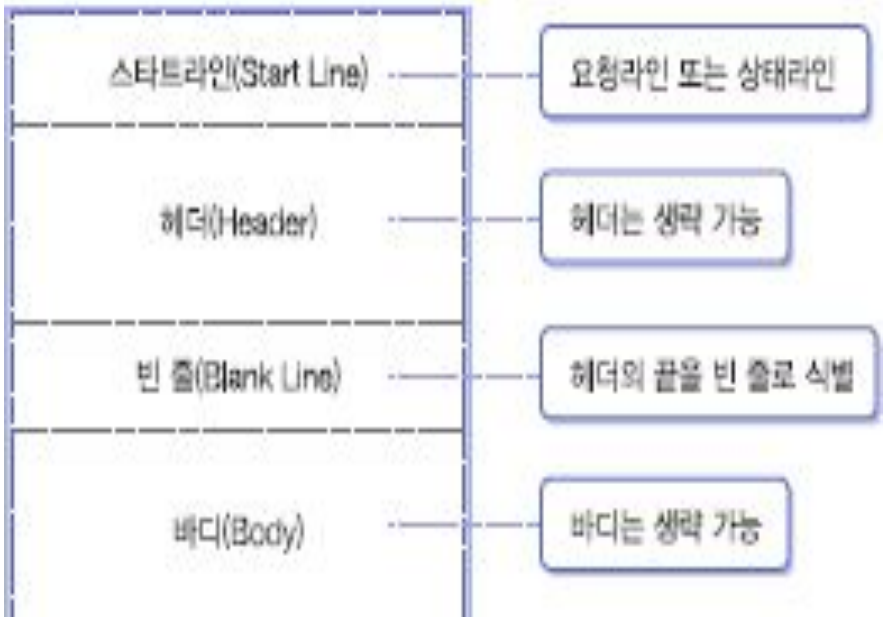
웹 클라이언트와 웹 서버를 같이 개발 및 웹 클라이언트 또는 웹 서버 하나만 개발

웹 서버를 개발하는 경우가 많아서 파이썬 웹 프로그래밍이라고 하면 우선적으로 장고Django와 같은 웹 프레임워크를 사용하여 웹 서버를 개발



HTTP프로토콜

HTTP 메시지의 구조




```
GET /book/shakespeare HTTP/1.1
Host: www.example.com:8080
```

첫 번째 줄은 요청라인으로, 요청 방식**method**,
요청 **URL**, 프로토콜 버전으로 구성

스타트라인은 요청 메시지일 때 요청라인request line이라고 하고, 응답 메시지일 때 상태라인status line이라고함

Note
URI vs URL란?URI는 Uniform Resource Indentifier의 약자로
URL(Uniform Resource Locator)과 URN(Uniform Resource Name)을 포함하는 좀 더 넓은 의미의 표현이지만, 웹 프로그래밍에서는 **URI**와 **URL**을 동일한 의미로 사용해도 무방

HTTP프로토콜

HTTP 처리 방식

메소드명	의미	CRUD와 매핑되는 역할
GET	리소스 취득	Read(조회)
POST	리소스 생성, 리소스 데이터 추가	Create(생성)
PUT	리소스 변경	Update(변경)
DELETE	리소스 삭제	Delete(삭제)
HEAD	리소스의 헤더(메타데이터) 취득	
OPTIONS	리소스가 서포트하는 메소드 취득	
TRACE	루프백 시험에 사용	
CONNECT	프록시 동작의 터널 접속으로 변경	

HTTP 메소드 종류

GET 방식은 지정한 URL의 정보를 가져오는 메소드로, 가장 많이 사용

POST의 대표적인 기능은 리소스를 생성하는 것으로,블로그에 글을 등록하는 경우가 이에 해당.
PUT은 리소스를 변경하는 데 사용

HTTP프로토콜

GET과 POST 메소드

앞에서 8가지의 HTTP 메소드를 소개하였지만,
현실적으로 가장 많이 사용하는 메소드는 GET과 POST 2가지

```
GET  http://docs.djangoproject.com/search/?q=forms&release=1  HTTP/1.1
```

```
POST  http://docs.djangoproject.com/search/ HTTP/1.1
Content-Type: application/x-www-form-urlencoded

q=forms&release=1
```

파이썬의 장고 프레임워크에서도 폼의 데이터는
POST 방식만을 사용하고 있음

네이버의 검색 창 – GET 방식 전달

HTTP프로토콜

상태 코드

서버에서의 처리 결과는 응답 메시지의 상태라인에 있는상태 코드Status code를 보고 파악할 수 있음

메소드명	의미	CRUD와 매핑되는 역할
1xx	Informational (정보 제공)	임시적인 응답으로, 현재 클라이언트의 요청까지 처리되었으니 계속 진행하라는 의미입니다. HTTP 1.1 버전부터 추가되었습니다.
2xx	Success(성공)	클라이언트의 요청이 서버에서 성공적으로 처리되었다는 의미입니다.
3xx	Redirection (리다이렉션)	완전한 처리를 위해서 추가적인 동작을 필요로 하는 경우입니다. 주로 서버의 주소 또는 요청한 URI의 웹 문서가 이동되었으니, 그 주소로 다시 시도해보라는 의미입니다.
4xx	Client Error (클라이언트 에러)	없는 페이지를 요청하는 것처럼 클라이언트의 요청 메시지 내용이 잘못된 경우입니다.
5xx	Server Error (서버 에러)	서버 측 사정에 의해서 메시지 처리에 문제가 발생한 경우입니다. 서버의 부하, DB 처리 과정 오류, 서버에서 익셉션이 발생하는 경우가 이에 해당합니다.

HTTP프로토콜

상태 코드

상태 코드	상태 텍스트	응답 문구	서버 측면에서의 의미
2xx	Success	성공	클라이언트가 요청한 동작을 수신하여 이해했고, 승낙했으며 성공적으로 처리했다.
200	OK	성공	서버가 요청을 성공적으로 처리했다.
201	Created	생성됨	요청이 처리되어서 새로운 리소스가 생성되었다. 응답 헤더 Location에 새로운 리소스의 절대 URI를 기록합니다.
202	Accepted	허용됨	요청은 접수했지만 처리가 완료되지 않았다. 클라이언트는 응답 헤더의 Location, Retry-After를 참고하여 다시 요청을 보냅니다.
3xx	Redirection	리다이렉션	클라이언트는 요청을 마치기 위해 추가적인 동작을 취해야 한다.
301	Moved Permanently	영구 이동	지정한 리소스가 새로운 URI로 이동했다. 이동할 곳의 새로운 URI는 응답 헤더 Location에 기록합니다.

HTTP프로토콜

상태 코드

상태 코드	상태 텍스트	응답 문구	서버 측면에서의 의미
303	See Other	다른 위치 보기	**다른 위치로 요청하라.** 요청에 대한 처리 결과를 응답 헤더 Location에 표시된 URI에서 GET으로 취득할 수 있습니다. 브라우저의 폼 요청을 POST로 처리하고 그 결과 화면으로 리다이렉트시킬 때, 자주 사용하는 응답 코드입니다.
307	Temporary Redirect	임시 리다이렉션	**임시로 리다이렉션 요청이 필요하다.** 요청한 URI가 없으므로, 클라이언트는 메소드를 그대로 유지한 채 응답 헤더 Location에 표시된 다른 URI로 요청을 재송신할 필요가 있습니다. 클라이언트는 향후 요청 시 원래 위치를 계속 사용해야 합니다. 302의 의미를 정확하게 재정의해서 HTTP/1.1의 307 응답으로 추가되었습니다.
4xx	Client Error	클라이언트 에러	**클라이언트의 요청에 오류가 있다.**
400	Bad Request	잘못된 요청	**요청의 구문이 잘못되었다.** 클라이언트가 모르는 4xx 계열의 응답 코드가 반환된 경우에도 클라이언트는 400과 동일하게 처리하도록 규정하고 있습니다.
401	Unauthorized	권한 없음	**지정한 리소스에 대한 액세스 권한이 없다.** 응답 헤더 WWW-Authenticate에 필요한 인증 방식을 지정합니다.
403	Forbidden	금지됨	**지정한 리소스에 대한 액세스가 금지되었다.** 401 인증 처리 이외의 사유로 리소스에 대한 액세스가 금지되었음을 의미합니다. 리소스의 존재 자체를 은폐하고 싶은 경우는 404 응답 코드를 사용할 수 있습니다.
404	Not Found	찾을 수 없음	**지정한 리소스를 찾을 수 없다.**
5xx	Server Error	서버 에러	**클라이언트의 요청은 유효한데, 서버가 처리에 실패했다.**
500	Internal Server Error	내부 서버 오류	**서버쪽에서 에러가 발생했다.** 클라이언트가 모르는 5xx 계열의 응답 코드가 반환된 경우에도 클라이언트는 500과 동일하게 처리하도록 규정하고 있습니다.
502	Bad Gateway	불량 게이트웨이	**게이트웨이 또는 프록시 역할을 하는 서버가 그 뒷단의 서버로부터 잘못된 응답을 받았다.**
503	Service Unavailable	서비스 제공불가	**현재 서버에서 서비스를 제공할 수 없다.** 보통은 서버의 과부하나 서비스 점검 등 일시적인 상태입니다.

URL 설계

URL 구성 항목

URL 스킴 : URL에 사용된 프로토콜을 의미.

- 호스트명 : 웹 서버의 호스트명으로, 도메인명 또는 IP 주소로 표현.
- 포트번호 : 웹 서버 내의 서비스 포트번호. 생략 시에는 디폴트 포트번호로, http는 80을, https는 443을 사용
- 경로 : 파일이나 애플리케이션 경로를 의미
- 쿼리스트링 : 질의 문자열로, 앰퍼샌드(&)로 구분된 이름=값 쌍 형식으로 표현
- 프라그먼트 : 문서 내의 앵커 등 조각을 지정

URL 설계

파이썬의 우아한 URL

URL을 정의하 기 위해 정규표현식Regular Expression을 추가적으로 사용할 수 있음.

```
urlpatterns = [
    path('articles/2003/', views.special_case_2003),
    path('articles/<int:year>/', views.year_archive),
    path('articles/<int:year>/<int:month>/', views.month_archive),
    path('articles/<int:year>/<int:month>/<slug:slug>/', views.article_detail),
]
```

```
urlpatterns = [
    path('articles/2003/', views.special_case_2003),
    re_path(r'^articles/(?P<year>[0-9]{4})/$', views.year_archive),
    re_path(r'^articles/(?P<year>[0-9]{4})/(?P<month>[0-9]{2})/$', views.month_archive),
    re_path(r'^articles/(?P<year>[0-9]{4})/(?P<month>[0-9]{2})/(?P<slug>[\w-]+)/$', views.
article_detail),
]
```

Django 일반적인 특징

| MVC 패턴 기반 MVT |

장고는 MVC(Model-View-Controller)를 기반으로 한 프레임워크입니다. 하지만 장고에서는 View를 Template, Controller를 View라고 함

| 객체 관계 매핑 |

장고의 객체 관계 매핑(ORM, Object-Relational Mapping)은 데이터베이스 시스템과 데이터 모델 클래스를 연 결시키는 다리와 같은 역할

자동으로 구성되는 관리자 화면 |

장고는 웹 서버의 콘텐츠, 즉 데이터베이스에 대한 관리 기능을 위하여 프로젝트를 시작하는 시점 에 기본 기능으로 관리자 화면을 제공합니다

| 우아한 URL 설계 |

웹 프로그래밍에서 URL 디자인은 필수인데, 장고에서는 유연하면서도 강력한 기능을 제공. 장고에서는 우아한(Elegant) URL 방식을 채택하여 URL을 직관적이고 쉽게 표현

| 자체 템플릿 시스템 |

장고는 내부적으로 확장이 가능하고 디자인이 쉬운 강력한 템플릿 시스템을 갖고 있음. 이를 통해 화면 디자인과 로직에 대한 코딩을 분리하여 독립적으로 개발 진행

일반적인 특징

| 캐시 시스템 |

동적인 페이지를 만들기 위해서 데이터베이스 쿼리를 수행하고 템플릿을 해석하며, 관련 로직을 실행해서 페이지를 생성하는 일은 서버에 엄청난 부하를 주는 작업

다국어 지원 |

장고는 동일한 소스코드를 다른 나라에서도 사용할 수 있도록 텍스트의 번역, 날짜/시간/숫자의 포맷, 타임존의 지정 등과 같은 다국어 환경을 제공

풍부한 개발 환경 |

장고는 개발에 도움이 될 수 있는 여러 가지 기능을 제공

소스 변경사항 자동 반영 |

장고에서는 *.py 파일의 변경 여부를 감시하고 있다가 변경이 되면 실행 파일에 변경 내역을 바로 반영

장고 프로그램 설치

윈도우에서 장고 설치

```
C:\Users\shkim>pip install Django
```

장고는 파이썬 환경에서 동작하는 패키지이므로, 장고가 정상적으로 설치되었는지 확인하기 위해서 아래와 같이 명령을 입력

```
C:\Users\shkim>python -m django --version
2.0.5
```

장고 프로젝트 설치

1. anaconda prompt
   - 장고설치
     pip install Django
   - 장고버전확인
     python -m django --version
   - 설정폴더로 위치 변경
     cd D:₩20220221₩python₩django
     cd d: => 드라이브가 다른경우
   - 프로젝트생성
     django-admin startproject study1
   - 프로젝트 폴더로 이동 : D:₩20220221₩python₩django ₩study1
     cd study1
   - application 생성
     python manage.py startapp member

장고 프로젝트 설치

2. 파일 탐색기 실행
   d:₩20220221₩python₩django₩study1₩templates 폴더 생성

3. spyder의 폴더를 d:₩20220221₩python₩django₩study1 폴더로 변경
   - study1 폴더
     settings.py 파일을 spyder에서 열기

settings.py 수정

```
INSTALLED_APPS = [
        .....
            'member',    추가
]


TEMPLATES = [
    'DIRS': [BASE_DIR/'templates'], =>
수정
]
```

장고 프로젝트 설치

settings.py 수정

```
DATABASES = {    => 수정
    'default' : {
        'ENGINE': 'django.db.backends.mysql',
        'NAME' : 'kicdb',
        'USER' : 'kic' ,
        'PASSWORD' : '1234',
        'HOST' : 'localhost',
        'PORT':'3306'
    }
}
 LANGUAGE_CODE = 'ko-kr'

 TIME_ZONE = 'Asia/Seoul'
```

장고 프로젝트 설치

4. 애플리케이션 개발하기 - spyder
   (1)study1/urls.py : path('member/',include('member.urls')) ->추가.
admin 삭제
   (2)member/urls.py 파일 생성
      from django.urls import path
      from . import views

      urlpatterns = [
         path('login/',views.login,name='login')
      ]
   (3) member/views.py
      def login(request): => 추가
         return render(request, 'member/login.html')

5. 파일 탐색기 실행
   (1) templates/member 폴더 생성
   (2) login.html 파일 생성

장고 프로젝트 설치

6. anaconda prompt
   -db설정하기
    python manage.py migrate
   - server 연결하기 (기본 8000번포트 설정됨)
    python manage.py runserver

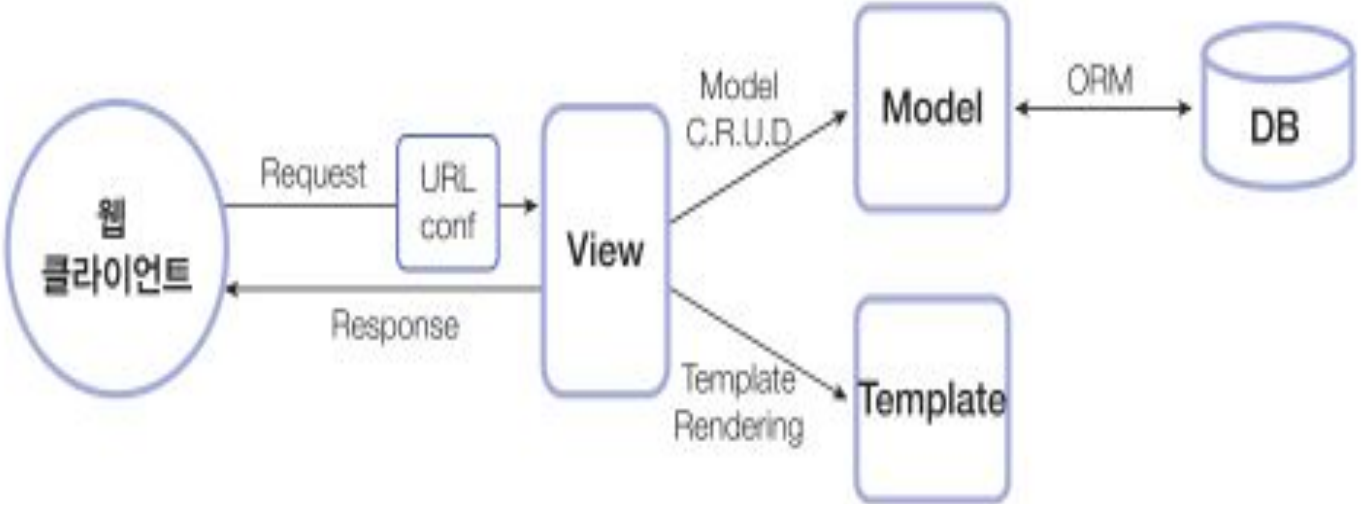
7. 브라우저띄우기
   http://localhost:8000

8. maria db 추가하기
   - mysql 설치
     pip install mysqlclient

   - db 수정 후
    python manage.py makemigrations => db변경내용 적용.
    python manage.py migrate

장고에서의 애플리케이션 개발 방식

개발 시 일반적으로 언급되는 MVC(Model-View-Controller) 패턴이란 데이터(Model), 사용자 인터페이스(View), 데이터를 처리하는 로직(Controller)을 구분해서 한 요소가 다른 요소들에 영향을 주 지 않도록 설계하는 방식



장고의 MVT 패턴

장고에서의 애플리케이션 개발 방식

Model – 데이터베이스 정의

모델이란 사용될 데이터에 대한 정의를 담고 있는 장고의 클래스입니다. 장고는 ORM 기법을 사용하여 애플리케이션에서 사용할 데이터베이스를 클래스로 매핑해서 코딩할 수있음

```
from django.db import models

class Person(models.Model):
    first_name = models.CharField(max_length=30)
    last_name = models.CharField(max_length=30)
```

```
CREATE TABLE myapp_person (
    "id" serial NOT NULL PRIMARY KEY,
    "first_name" varchar(30) NOT NULL,
    "last_name" varchar(30) NOT NULL
);
```

장고는 테이블 및 컬럼을 자동으로 생성하기 위해 필요한 많은 규칙을 갖고 있습니다. 위의 예제는 그중에서 다음과 같은 규칙이 적용.

3.3.3 URLconf – URL 정의

파이썬의 URL 정의 방식은 전통적인 자바나 PHP 계열의 URL보다 직관적이고 이해하기가 쉬움.
그래서 이런 방식을 우아한Elegant URL이라고 부르는 것.
URL을 정의하기 위해서는 다음 예제처럼 **urls.py** 파일에 URL과 처리 함수(뷰View라고 부름)를 매핑하는 파이썬 코드를 작성하면 됨

예제 3-1 URLconf 예시

```
from django.urls import path

from . import views

urlpatterns = [
    path('articles/2003/', views.special_case_2003),
    path('articles/<int:year>/', views.year_archive),
    path('articles/<int:year>/<int:month>/', views.month_archive),
    path('articles/<int:year>/<int:month>/<slug:slug>/', views.article_detail),
]
```

장고에서의 애플리케이션 개발 방식

View – 로직 정의

뷰는 웹 요청을 받아서 데이터베이스 접속 등 해당 애플리케이션의 로직에 맞는 처리 를 하고, 그 결과 데이터를 HTML로 변환하기 위하여 템플릿 처리를 한 후에, 최종 HTML로 된 응답 데이터를 웹 클라이언트로 반환하는 역할

```
from django.http import HttpResponse
import datetime

def current_datetime(request):
    now = datetime.datetime.now()
    html = "<html><body>It is now %s.</body></html>" % now
    return HttpResponse(html)
```

장고에서의 애플리케이션 개발 방식

Template – 화면 UI 정의

개발자가 응답에 사용할 *.html 파일을 작성하면, 장고는 이를 해석해서 최종 HTML 텍스트 응답을 생성하고, 이를 클라 이언트에게 보내줍니다. 클라이언트(보통 웹 브라우저)는 응답으로 받은 HTML 텍스트를 해석해 서 우리가 보는 웹 브라우저 화면에 UI를 보여주는 것

템플릿 파일은 *.**html** 확장자를 가지며, 장고의 템플릿 시스템 문법에 맞게 작성

장고에서 템플릿 파일을 찾을 때는 TEMPLATES 및 INSTALLED_APPS에서 지정된 앱의 디렉 토리를 검색.
이 항목들은 프로젝트 설정 파일인 **settings.py** 파일에 정의되어 있음

템플릿 시스템

템플릿 변수

```
{{ variable }}
```

템플릿 시스템은 변수를 평가해서 변수값으로 출력.
변수명은 일반 프로그래밍의 변수명처럼 문자, 숫자, 밑줄(_)을 사용하여 이름을 정의

```
{{ name|lower }}
```

아래의 예시처럼 파이프(|) 문자를 사용.
name 변수값의 모든 문자를 소문자로 바꿔주는 필터

```
{{ text|escape|linebreaks }}
```

필터를 체인으로 연결할 수도 있음

```
{{ bio|truncatewords:30 }}
```

다음은 bio 변수값 중에서 앞에 30개의 단어만 보여 주고,
줄 바꿈 문자는 모두 없애줌.

템플릿 시스템

템플릿 필터

```
{{ list|join:" // " }}
```

필터의 인자에 빈칸이 있는 경우는 따옴표로 묶어줌

```
{{ value|default:"nothing" }}
```

value 변수값이 False이거나 없는 경우, “nothing”으로 보여줌

```
{{ value|length }}
```

value 변수값의 길이를 반환

```
{{ value|striptags }}
```

value 변수값에서 HTML 태그를 모두 없애줌

템플릿 시스템

템플릿 태그

```
<ul>
{% for athlete in athlete_list %}
    <li>{{ athlete.name }}</li>
{% endfor %}
</ul>
```

for 태그에 사용되는 변수들

변수명	설명
forloop.counter	현재까지 루프를 실행한 루프 카운트(1부터 카운트함)
forloop.counter0	현재까지 루프를 실행한 루프 카운트(0부터 카운트함)
forloop.revcounter	루프 끝에서 현재가 몇 번째인지 카운트한 숫자(1부터 카운트함)
forloop.revcounter0	루프 끝에서 현재가 몇 번째인지 카운트한 숫자(0부터 카운트함)
forloop.first	루프에서 첫 번째 실행이면 True 값을 가짐
forloop.last	루프에서 마지막 실행이면 True 값을 가짐
forloop.parentloop	중첩된 루프에서 현재의 루프 바로 상위의 루프를 의미함

템플릿 시스템

템플릿 태그

{% if %} 태그

```
{% if athlete_list %}
    Number of athletes: {{ athlete_list|length }}
{% elif athlete_in_locker_room_list %}
    Athletes should be out of the locker room soon!
{% else %}
    No athletes.
{% endif %}
```

| {% csrf_token %} 태그 |

```
<form action="." method="post">{% csrf_token %}
```

| {% url %} 태그 |

```
<form action="{% url 'polls:vote' question.id %}" method="post">
```

템플릿 시스템

템플릿 태그

| {% with %} 태그 |

```
{% with total=business.employees.count %}
    {{ total }} people works at business
{% endwith %}
```

| {% load %} 태그 |

```
{% load somelibrary package.otherlibrary %}
```

템플릿 시스템

템플릿 주석

첫번째는 한 줄 주석문으로, 아래처럼 **{# #}** 형식을
따름. 한 문장의 전부 또는 일부를 주석 처 리하는 방법

```
{# greeting #}hello
```

{# #} 주석문 내에 템플릿 코드가 들어있어도
정상적으로 주석 처리됨

```
{# {% if foo %}bar{% else %} #}
```

여러 줄의 주석문으로, 아래처럼 **{%** comment **%}** 태그를
사용합니다

```
{% comment "Optional note" %}
<p>Commented out text here</p>
{% endcomment %}
```

템플릿 시스템

HTML 이스케이프

이처럼 사용자가 입력한 데이터를 그대로 렌더링하는 것은 위험
장고는 앞 서와 같은 결과를 방지하기 위해서 자동 이스케이프 기능을 제공.
즉 장고는 디폴트 로 HTML에 사용되는 예약 문자들을 아래처럼 예약 의미를 제거한 문자로 변경해주는 기능을 제공

< (less than) 문자는 < 로 변경함
> (greater than) 문자는 > 로 변경함
' (single quote) 문자는 ' 로 변경함
" (double quote) 문자는 " 로 변경함
& (ampersand) 문자는 & 로 변경함

템플릿 시스템

템플릿 상속

```
<!DOCTYPE html>
<html lang="en">
<head>
    <link rel="stylesheet" href="style.css" />
    <title>{% block title %}My amazing site{% endblock %}</title>
</head>

<body>
    <div id="sidebar">
        {% block sidebar %}
        <ul>
            <li><a href="/">Home</a></li>
            <li><a href="/blog/">Blog</a></li>
        </ul>
        {% endblock %}
    </div>

    <div id="content">
        {% block content %}{% endblock %}
    </div>
</body>
</html>
```

폼처리하기

폼 클래스를 템플릿으로 변환

- {{ form.as_table }} : <tr> 태그로 감싸서 테이블 셀로 렌더링됩니다. {{form}}과 동일함
- {{ form.as_p }} : <p> 태그로 감싸도록 렌더링됩니다.
- {{ form.as_ul }} : <li> 태그로 감싸도록 렌더링됩니다.

```
from django import forms


class ContactForm(forms.Form):
    subject = forms.CharField(max_length=100)
    message = forms.CharField(widget=forms.Textarea)
    sender = forms.EmailField()
    cc_myself = forms.BooleanField(required=False)
```

```
<p><label for="id_subject">Subject:</label>
    <input id="id_subject" type="text" name="subject" maxlength="100" /></p>
<p><label for="id_message">Message:</label>
    <input type="text" name="message" id="id_message" /></p>
<p><label for="id_sender">Sender:</label>
    <input type="email" name="sender" id="id_sender" /></p>
<p><label for="id_cc_myself">Cc myself:</label>
    <input type="checkbox" name="cc_myself" id="id_cc_myself" /></p>
```