**Web Programlama**

Web tarayıcısı göndermek istediğimiz bilgiler ile web sunucusuna iletilir. Web sunucusu iletilen sayfayı ve diğer bilgileri bir dış ortama (**CGI(**Common Gay İnterrface**)**) aktarır. Bilgiler dış ortama ya komut satırından ya da standart girdiler aktarılır. Dış ortam(CGI) istemcinin istediği sayfayı oluşturmak üzere muhtemelen bir veri tabanı ile bağlantıya geçer ve sayfayı(**Html, resim**) üretir. **HTML** kodlarını standart çıktıya yazar. Web sunucusu standart çıktıdaki tüm kodu istemciye gönderir.

Veri Tabanı

CGI

Web Sunucusu

Web Sayfası

Web sayfasını en performanslı şekilde web sunucusu ile bütünleştirilmiş bir yorumlayıcının dış program kodlarının yorumlamasıdır.

**Django** mod\_python veya mod\_wsgi kullandığından performansı yüksektir.Django yüksek performanslı bir **python**  web framewok’üdür.

**Gereksinimler**

* Python 3.x sürümü bilinmelidir.
* Herhangi bir ortamda ya da dil ile web programcılığı hakkında bilgi edilmelidir.
* **SQL** ifadeler hakkında bilgiye sahip olunmalı.
* **HTML, CSS,** birazda **js** bilinmeli

**Django Kurulumu**

Windows için;

* Python 3.x indir ve kur
* Django istenilen sürümü kur
* Komut satırına;
  + Django (sürüm)
  + Python dizinine gir \python.exe setup.py install

Linux için;

* Python kurulu zaten
* Django indir ve kur
* Terminale;
  + Tar –txf Django-(sürüm).tar.gz
  + Cd Django(sürüm)
  + Python setup.py install

**Veri Tabanı**

Django bir çok veri tabanı ile entegreli bir şeklide çalışabilmektedir. Django kurduğumuzda bize **SQLite** sunan django bu veri tabanı ile bir çok güçlüklerin üstesinden gelir ce çok kurulum gerektirmez.

**Not: IDLE** seçimlerimizde editörünüzün python destekli olmasına dikkat edin .

**Bir Proje Oluşturma**

Windows;

* Mkdir d:\django
* Python dizini\lib\site-packages\django\django-admin.py startproject Deneme

Linux;

* Mkdir ~\django
* django-admin.py startproject Deneme

Deneme\

* **\_\_init\_\_.py :** Python paket klasörü , bir şey yazılmaz
* **Settings.py:** Django ile ilgili yapılandırma ayarlarının bulunduğu dosya
* **Urls.py:** Projedeki adreslerintutulduğu dosya
* **Wsgi.py:**  Django geliştirme sunucusu WSGI uygulamalarını içerir.

**Manage.py:** Django promizle etkileşmek için kullanılan bir dosyadır, server başlatmak için kullanılır.

**Server Çalıştırma**

Windows;

* D:\django\deneme cd \django\deneme c:\python\python.exe manage.py ruunserver

Linux;

* Python manage.py runserver

Sunucunun çalışması için hata oluşmaması için hatalar manage.py tarafından denetlenir. Sunucuyu 192.168.128.174:8080 de çalıştırmak;

- D:\django\deneme > c:\python\python.exe setup.py runserver 192.168.128.174:8080

- python manage.py runserver 192.168.128.174:8080

Tüm arayüzlerin tüm adreslerini dinletmek istiyorsak IP adresi yerine 0.0.0.0 yazabiliriz.

**Görünümler(Views)**

Bir sayfanın görüntülenebilmesi için öncelikle bu sayfayı oluşturarak görünümü hazırlamamız gerekir. Django’da görünümden basit tarayıcıda bir adres çağrıldığında istenecek olan işlevdir.

**Adresler(Urls)**

Web tarayıcıdan bir adresi istediğimizde Django hangi işlevi istediğimizi urls.py’da belirtilir. Bu dosya Django tarafından otomatik olarak oluşturulur.

url(r’^adres/’, views.adres),

adres/ : tarayıcıdan istediğimiz adres

views.adres : Bu adres çağırıldığında views.py daosyasında tanımlı işlenecek olan işlevlerdir.

Statik içerikler sabit olup dinamik içerikler son oluşturulan HTML kadar **HTTPResponse()** nesnesine argüman olarak verilir.

r’adres/(ADR\d+)/’….

r : Düzenli ifade(reguler expresion) olduğunu belirtir.

‘ : adres başlangıcı olduğunu belirtir.

\d : tek bir sayı, + : bir önceki desenden bir veya daha fazla anlamına gelir.

ÖRN : ADR1, ADR2, …

r’adresler/(.+)/(.+)/$’ …..

$ : Adresin bittiğini belirtir.

ÖRN : adresler/adr/adres

**Şablonlar(Templates)**

Standart Html etiketleriyle birlikte bizim oluşturduğumuz bir takım etiketler vardır. Bu etiketler **{% etiket %}** şeklindedir.

Django etkileşimli kabuk açma

Windows;

* D:\djkodları\deneme\> cd \djkodları\deneme
* D:\djkodları\deneme> z:\python\ python.exe manage.py Shell

Linux;

* Cd djkodları\
* Manage.py Shell

**İf else etiketi**

**{% if %} … {% else %} …. {% endif %}**

**{% ifequal %}**

Eğer bir değeri başka bir değer ile karşılaştıracaksak bu etiketi kullanırız

**{% for %}**

**{% for %} …. {% endfor %}**

Sayaç indislerine **forloop.counter**  ile erişilir.

Sayaç tersten almak için **forloop.revcounter** bağlam değişkeni kullanılır. Ters sayacı 0’da sonlandırmak istiyorsak **forloop.revcounter0**  değişkeni işimizi görür.

Başlangıcı ve sonunu bilmek istersek döngünün **forloop.first, forloop.last**  kullanırız.

**Filtrelerden Bazıları**

**Add :** toplama işlemi yapar

**Copfirst, title :** cümlenin ilk harfini büyük harfe çevirmek için copfirst, başlık için title kullanılır.

**Default :**  eğer bağlanan değerinin değeri False olarak dönerse ön tanıml değer basılır.

**Join:** Python’daki Join() işlevi ile aynıdır.

**Length :**  Uzunluğu verir.

**Ljust, rjust, center :** Sırası ile kelimeyi sola, sağa yaslar ve ortalar.

**Lower, upper :** Küçük, büyük harf dönüşümü yapar.

**Pluralize :**  s takısı ile yazılır.

**Random :**  Listeden random eleman seçer.

**Time:** datetime biçiminde zamanları n gösterimek için kullanılır.

**Unorder\_list:** listelerden sırasız Html listesi oluşturur.

**Urlize:** Bir metindeki adresleri url biçiminde ifade eder.

**Templates Views ile kullanılması**

**Urls.py :**  url(r’link/’, views.fonk),

**Views.py :** def fonk():

Django’nun template yükleme sistemi **settings.py’da TEMPLATES\_DIRS** değişkenine projemize ait şablonları kaydettiğimiz klasörü eklememiz gerekir.

**Render\_to\_response :**  Django bu kadar kolaylık sunuyorken şablon içerip veriyi bağlam haline getirip daha sonra bunu yorumlatmak zor bunun yerine bu fonksiyon kullanılır.

**Locals():** Bir görünüm altında tanımlanmış değişken(nesne) isimlerinden oluşan bir sözlük döndürür.

**Djando Veri Toplama Yapılandırması**

Django ön tanımlı olarak **PostreSQL**, **MYSQL** ve **Oracle** desteği ile gelir. Veri tabanı yapılandırması **settings.py**  **DATABASE**  bölümüdür. Hangi veritabanı kullanıyorsak **ENGINE** parametresinde belirtiyoruz. Bu parametreler; **postresql.pycep2, mysql, sqlite3, oracle**  olabilir ve bunlardan önce **django.db.backend.**  kullanırız.

**Uygulama Oluşturma**

Windows**;**

* D:\djkodlari > cd Proje
* D:\djkodları\proje > c:\Python\python.exe manage.py startapp App\_name

Linux**;**

* cd djkodlari
* ./manage.py startapp App\_name

Settings.py **INSTALLED\_APPS** gidip **‘app\_name’**  ekliyoruz

**Modelimiz(models.py)**

Django’da veri tabanları models.py içinde yazılır.

Class Tablo\_Adı(models.Model):

Özellikler

Herhangi bir alanın seçeneğe bağlı olarak girilmesini sağlamak için , o alanı tanımlarken **blank=True**  argümanını eklememiz gerekir Şayet veri girilmesini zorunlu kılmayacaksak. Veri tabanı değişikliklerini uygulatmak ve veri tabanını oluşturmak

Windows;

* C:\Python\python.exe manage.py migrate
* C:\Python\python.exe manage.py makemigrations

Linux;

* ./manage.py migrate
* ./manage.py makemigrations

Admin paneline giriş yapabilmek için

Windows;

* C:\Python\python.exe manage.py createsuperuser

Linux;

* ./manage.py createsuperuser

**GET ve POST**

Kullanıcı veriyi bizlere iki yöntemle gönderebilmektedir; GET ve POST iki yöntemle de kullanıcıdan veri almamız mümkün. Ancak farkları vardır; Get ile veri alırsak tüm veri web tarayıcısının adres satırında göründüğünden başkalarının tarayacının geçmişine kullanıcıların hangi verilerin gönderildiğini izleyebilir. Post ise form verileri tarayıcıya doğrudan gönderildiğinden adres satırında görünmez boyutu istediğimiz kadar olabilir. Böyle baktığımızda sanki her zaman Post yöntemini kullanmak daha akıllıcadır.

Kullanıcıdan gelen veri üzerinde işlem yapacaksak ve bunu sunucuda bir şekilde saklayacaksak( veri tabanında ya da dosya olarak) Post yöntemini aksi halde Get yöntemini kullanmalıyız.

**Formlar(forms.py)**

**Csrf\_token(cross site request forgery)** : csrf saldıralarından korumak için form içerisinde her oturum için rasgele bir değer üretilip bu değerin kullanıcıdan gelen ile sunucudaki karşılaştırıldıktan sonra işlemin yapılmasının sağlar.

**Çerezler(Cookies)**

HTTP'de istemci ile sunucu arasındaki bağlantıların takibi için çerezler kullanılır. Çerezler küçük bilgi parçacıklarıdır.

Sunucuya istemciye bu bilgi parçacığını verir, istemcide sürekli bağlantıalrın da bu bilgi parçacığını sunucuya tekrar iletir. Böylece hangi kullanıcı ile muhattap olduğunu anlar. Django çerezleri **request.COOKİES** nesnesinde barındırır Bu nesne bir sözlük gibi davranır.

Çerezlerin ömrü normalde kapandığında son bulur. Ancak daha uzun süreli saklamak istersek çrez ataması yaparken **expires**  parametresi verilir.

**Oturumlar(Sessions)**

Django oturumları çerezleri kullanarak sağlar. Django oturum bilgilerini bu çerezi anahtar kabul eden bir sözlükte tutar ve veri tabanındaki **django\_session** isimli tabloya kaydeder. Oturum bilgileri **request.session** nesnesinde saklanır.

**Kullanıcı Yetkileri**

Bir kullanıcı **superuser**  ve **staff** ise Django yönetim(admin) panelinden giriş yapabilir ve tüm yetkilere sahiptir.Kullanıcıların izinleri vardır. Bunları yönetim panelinden yapabiliriz. Mevcut izinlerin haricinde kullanıcılara izin atamasıda yapılabilir.(**hos\_perm, Permision ile)**

**Model İzinleri**

Özel izinleri model de tanımlayabiliriz.

**Class Meta:**

**Permissions = (….)**

Deneysel e-posta sunucusu komutu

**Python –m smtpd –n –c DebuggingServer localhost:port\_numarası**

Django ile standart kullanıcı ayarlamaları yapılmaktadır. Kullanıcı oluşturma, düzenleme,silme,parola değiştirme mail doğrulama vs. Web site adını **.site** methodundan yapabiliriz.

**CAPTHA ile İnsanlık Testi**

Web sitemize kullanıcı kaydının insan tarafından yapıldığına emin olmak için üzerindeki harf ile doğrulama yapılır. **DJANGO CAPTCHA**  kullanarak bu işlemi basit bir şekilde yapar.

Windows için CAPTCHA (setuptools gerekli), PIL kütüphanesine ihtiyaç var.

Linux için CAPTCHA kurulur.

Projeye eklemek için **INSTALL\_APPS’**e ‘captcha’ eklenir. Çalıştırmak için **python manage.py test captcha**  yapılır. Urls.py’a **url(r’captcha/’, include(captcha.urls)),** eklenir ayrıca kullanıcı formunada import edilmelidir. Kullanıcı resimlerini kaldırmak için değiştirdikten sonra **django\_cleanup**  kullanabiliriz.

**Caching**

Dinamik web sitelerinin, bir istek geldiğinde cevaplanabilmesi için bir çok işlem yapılır. Veri tabanı bağlantısı, çeşitli sorgulamalar ve bir takım hesaplamalardan sonra nihayet sayfa oluşturulur ve istemciye gönderilir.

Bir istemci bir sayfayı istediğinde (aslında bir adres, url) önce cachle bakacağız, cache’de var ise hiç hesap kitap yapmadan doğrudan bunu göndereceğiz. Yoksa ise sayfa oluşturulup, istemciye gönderilecek ve cache kaydedilecek

Django Cache Yöntemleri ;

* Dosya Sistemi
* Veri Tabanı
* Bellek

**Dosya Sistemi Cachleme**

En kolay yöntemdir. İstanilen sayfa oluşturulduktan sonra belirli bir anahtarlama şifresiyle dosya sistemine kaydedilir, daha sonra aynı sayfa istenildiğinde dosya sisteminden okunarak istemciye gönderilir (Ağ için NTFS). Settings.py’da **CACHES**  ayarları yapılır(File Based Cache)

**Veri Tabanı Cachleme**

Önce -> /manage.py createcachetable table, sonra settings.py CACHES Backend: DatabaseCache;

**Bellek Cachleme**

En hızlı ve en pahalı yöntemdir. Dezavantajı sistem yeniden başlatıldığında cache’ler gider iki çeşiti vardır

1 – yerel bellek : Local bellekte tutulur. Settings.py Backend: LocMemCache yapaılr. En çok memcached sunucu ile yapılır.

2 – Dağıtık Sistem bellekleri : Cachleme sunucusu kurulur. Başlangıç türü otomatik yapılır. Pythn için memcached kurulmalıdır. Settings.py Backend: memCachedCache eklenir

Cache’leme de internete doğrudan erişim makinelerde herkes erişmesi için memcached sunucusu hiç çakıştırmamak gerekir.

**MiddleWare\_Chaces’a** UpdateCacheMiddliware, CommonMiddliware, FetchfromCacheMiddliware eklenir.

Anonim kullanıcılar cache’li sayfa görmek için **CACHE\_MIDDLEWARE\_ONLY =True**  yapılmalıdır.

Proje geliştirildikten sonra hata ayıklama kipi olan **DEBUG = False** olmalı çünküğü bilgiler başkasına görülmemeli . Hata ayıklama kapatıldığı zaman CSS dosyaları gitmesi için : **./manage.py runserver –insecure** yapılmalıdır.

**404 Hatası Şablonu**

Request nesnesi şablon içerisinden erişmek için settings.py **TEMPLATE\_CONTEXT\_PROCESSURS ={**

**….request, …. auth }**

**500 Hatası Şablonu**

Settings.py **ADMINS = ( “ad”, ”…@mail.com”)**

Hataların admine gitmesi için  **SERVER\_EMAİL = ‘mesaj’ ‘<email>’**