



# Dynamic Web Sites

COMP 8347

Usama Mir

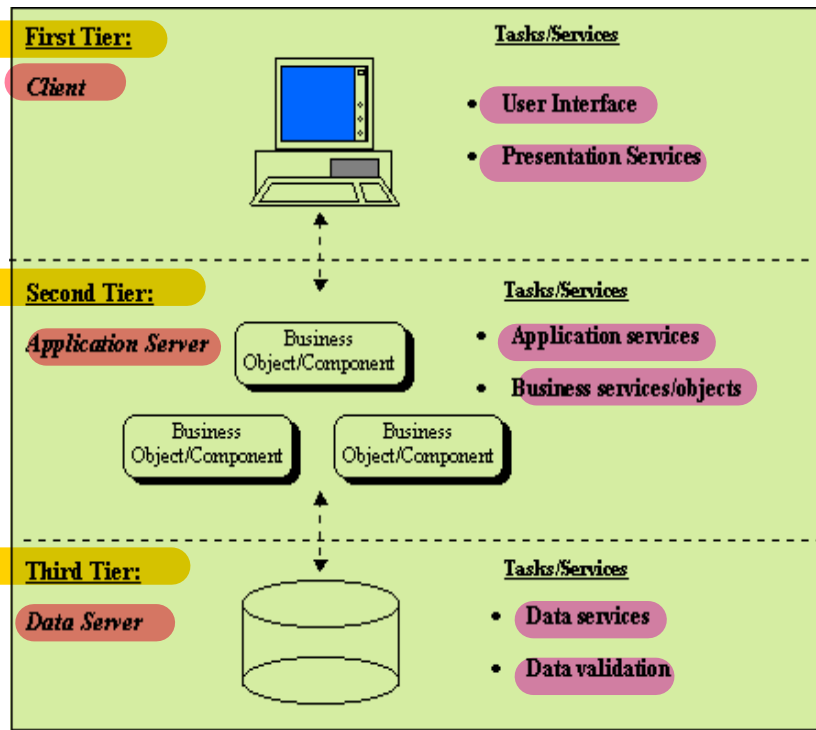
Usama.Mir@unwindsor.ca

# Dynamic Web Sites

## Topics

- ▶ Static vs. Dynamic Websites
- ▶ HTTP
- ▶ HTML
- ▶ MVC
- ▶ MTV (Django)

# Client Server Model



User requests document from the Web server.



Web server fetches and/or generates the necessary document.



The result is returned to the user's browser.



The browser renders the document.

\*Fig. taken from [1]

# Static vs. Dynamic Web Pages

- ▶ **Static web page**: requests to the same URL always return the same information.
  - ▶ Content consists of HTML text files stored on the server.
  - ▶ URL typically does not contain parameters; simply a 'path'
  - ▶ Primarily informational
  - ▶ HTML + CSS + JS
  - ▶ Server does not generate dynamic content but perform the hosting
  - ▶ Static does not mean the content cannot be changed - JS running in the browser can still change the content
  - ▶ Examples: Resume or personal websites

# Static vs. Dynamic Web Pages

- ▶ **Dynamic web page:** Data returned by a given URL can vary significantly.
  - ▶ Generates content and displays it based on actions the users make on the page
  - ▶ Functional and informational
  - ▶ HTML/XML + some server-side language like PHP or Node JS
  - ▶ Server generates dynamic HTML pages on runtime
  - ▶ Pages can still be rendered later at browser
  - ▶ Examples: Location-based sites, all others ex. Instagram

# Static vs. Dynamic Web Pages

## Advantages and Disadvantages

### ► **Static:**

+Easy creation

+Easy and faster loading

+Easy security of static content

-No flexibility

-Difficult to manage

### ► **Dynamic:**

+Easy maintenance/update

+Better user experience

+Greater functionality

-Performance issues due to large number of instructions

-Needs more resources

# HTTP

- ▶ **HTTP**: Hyper-Text Transfer Protocol
  - ▶ Encapsulates the process of serving web pages
  - ▶ Protocol for client-server communication
  - ▶ Current version is HTTP/3.
  - ▶ **A network protocol**: defines rules and conventions for communication between network devices.
- ▶ HTTP is stateless
  - ▶ Server maintains no information on past client requests.

# HTTP

## Application layer protocol

- Client sends request
- Server responds with reply
- Other application layer protocols are FTP, SMTP, POP etc.

## Almost always run over TCP

- Uses 'well known' port 80 (443 secure)
- Can support multiple request-reply exchanges over a single connection



# Uniform Resource Locators

- ▶ In the Web, functionality of pointers is provided by Uniform Resource Locators (URLs).

- ▶ URL example:

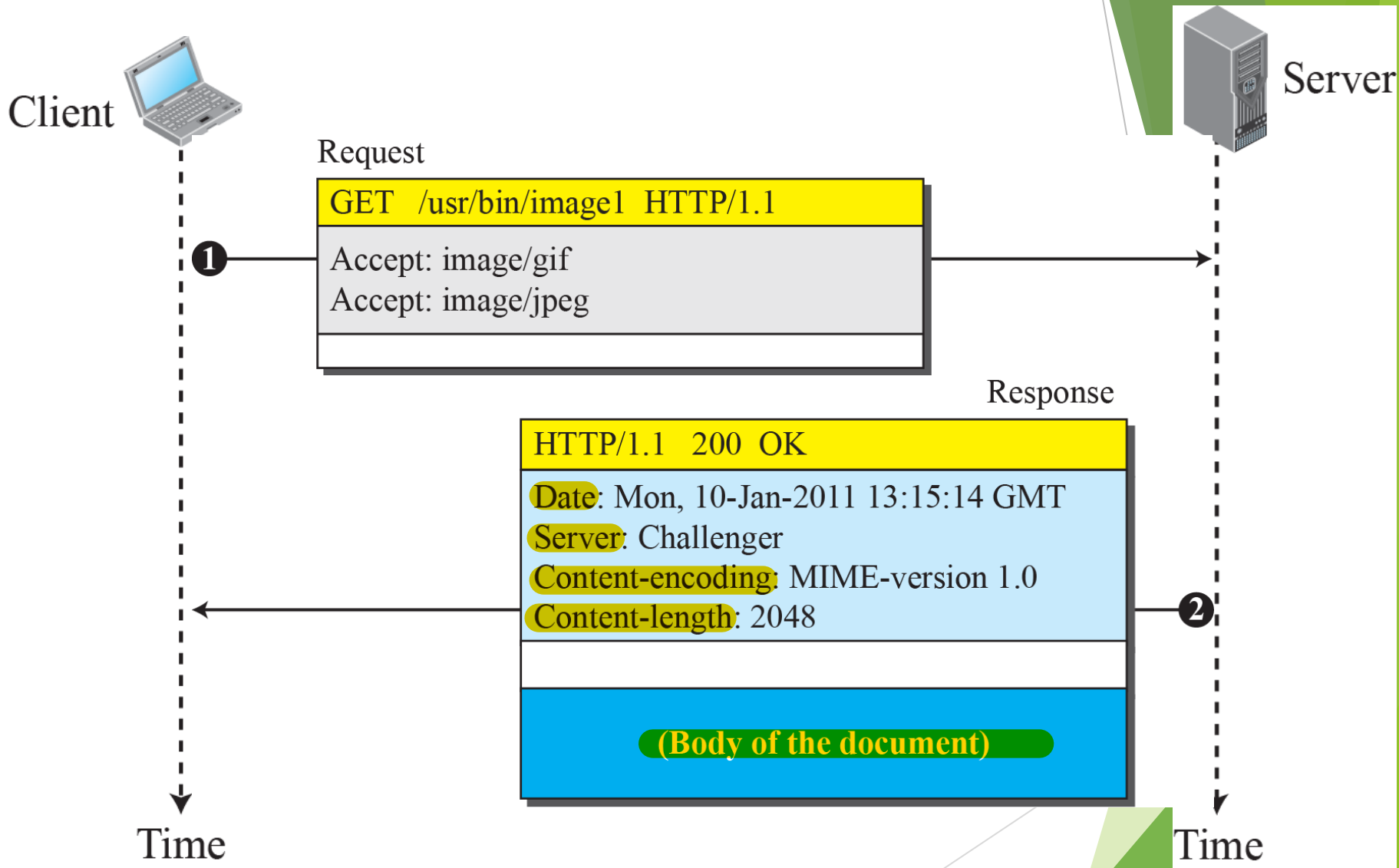
<http://www.acm.org/sigmod>

- ▶ The first part indicates how the document is to be accessed
  - ▶ “http” indicates that the document is to be accessed using the Hyper Text Transfer Protocol.
- ▶ The second part gives the unique name of a machine on the Internet.
- ▶ The rest of the URL identifies the document within the machine.
- ▶ The local identification can be:
  - ▶ The path name of a file on the machine, or
  - ▶ An identifier (path name) of a program, plus arguments to be passed to the program
    - ▶ E.g., <http://www.google.com/search?q=silberschatz>

# HTTP Methods

- ▶ **GET:** Used to retrieve information from the given server using a given URI.
  - ▶ should only retrieve data and should have no other effect on the data.
- ▶ **POST:** Used to send data to the server, e.g. customer info, using HTML forms.
- ▶ Other methods: PUT, DELETE, TRACE etc

# HTTP Requests



# HTTP Responses

- Status-Line = HTTP-Version SP Status-Code SP Reason-Phrase CRLF

HTTP/1.1 200 OK

Date: Mon, 27 Jul 2009 12:28:53 GMT

Server: Apache/2.2.14 (Win32)

Last-Modified: Wed, 22 Jul 2009 19:15:56 GMT

Content-Encoding: MIME- version 1.0

Content-Length: 88

Content-Type: text/html

Connection: Closed

<html>

<body>

<h1>Hello, World!</h1>

</body>

</html>

# Status Codes

- ▶ **1xx: Informational:** request received and continuing process.
  - ▶ Ex. 102 Processing
- ▶ **2xx: Success:** action was successfully received, understood, and accepted.
  - ▶ Ex. 200 OK
- ▶ **3xx: Redirection:** further action must be taken in order to complete the request.
  - ▶ Ex. 307 Temporary redirect

# Status Codes

- ▶ **4xx: Client Error:** request contains bad syntax or cannot be fulfilled
  - ▶ Ex.
  - ▶ 403 Forbidden
  - ▶ 404 Not Found
- ▶ **5xx: Server Error:** server failed to fulfill an apparently valid request
  - ▶ Ex. 505 HTTP Version Not Supported

# What Is HTML?

- ▶ HTML is a markup language used to describe webpages.
  - ▶ HTML stands for HyperText Markup Language. When a web browser displays a webpage:
    - ▶ it is reading and interpreting an HTML document.
  - ▶ Used for structuring and presenting content on the World Wide Web.
  - ▶ Some related standards include CSS3

# Basic Structure

- ▶ **DOCTYPE**: Tells browsers *how* to read your document.
  - ▶ Forces browsers to use **'standard mode'**.
  - ▶ Using standard mode, most browsers will read your document the **same way**.
- ▶ **<head>**: Contains **information** about your page.
- ▶ **<body>**: The **actual content** of your page.

```
<!DOCTYPE html>
```

```
<html>
```

```
  <head>
```

```
    <title>My first Webpage</title>
```

```
  </head>
```

```
  <body>
```

```
    <h1>This is a Heading</h1>
```

```
  <p>Hello World!</p>
```

```
  </body>
```

```
</html>
```



# Elements

- ▶ HTML **elements** are marked up using **start tags** and **end tags**.
  - ▶ Tags are delimited using **angle brackets** with the tag name in between.
    - ▶ **End tags** include a **slash** before the tag name.
    - ▶ Some elements require **only a single tag**, e.g. `<img>`
  - ▶ HTML tag names are **case insensitive**.
    - ▶ Recommended: use **lowercase**.
  - ▶ Most elements contain some **content**
    - ▶ e.g. `<p>...</p>`
  - ▶ Elements may **contain attributes**
    - ▶ Used to set **various properties** of an element.

# Attributes

- ▶ **Attributes**: provide additional information about the specific element
  - ▶ Always specified in the opening tag.
  - ▶ The pattern for writing attributes: **attribute="value"**.
  - ▶ Examples:
    - ▶ ``
    - ▶ `<div class="example">...</div>.`
    - ▶ `<a href="http://www.myurl.com">This is a link</a>`

# HTML Forms

- ▶ HTML forms are used to collect user input.
  - ▶ The **<form>** tag is used to create an HTML form.
  - ▶ HTML forms contain **form elements**.
  - ▶ The **<input>** element is the most important form element.
    - ▶ has many variations, depending on the **type** attribute.
    - ▶ **Text** Defines normal text input
      - ▶ Default width is 20 characters.
    - ▶ **Radio** Defines radio button input (for selecting one of many choices)
    - ▶ **Submit** Defines a submit button (for submitting the form)
    - ▶ **Other elements**: Reset button, checkbox, dropdown list, time, date, file, image, month, and so on.

# HTML Forms - Example

```
<form action="/url_for_processing/" method="post" >  
<label for="uname">Username:</label>  
<input type="text" name="uname"><br><br>  
<input type="radio" name="gender" value="male" >Male<br>  
<input type="radio" name="gender" value="female" >Female<br>  
<input type="submit" value="Submit now" >  
</form>
```

Username:

☐ Male  
☐ Female

# Web Framework

- ▶ **Web framework**: a software framework designed to support development of dynamic websites and services.
  - ▶ Alleviate overhead with associated activities
- ▶ Frameworks standardize the 'boilerplate' parts.
  - ▶ Provide pre-built components so you can focus on unique parts of your project.
  - ▶ Repetitive parts handled by framework.
  - ▶ Code you use will be well tested, and have less bugs than what you write from scratch.
  - ▶ Enforce good development practices.
  - ▶ Security features (login, sessions etc) often better implemented in frameworks.
- ▶ Limitations:
  - ▶ May restrict you in terms of coding paradigms.
  - ▶ Steep learning curve.

# Different Frameworks

- ▶ Many different frameworks are available:
  - ▶ ASP.NET using C#, Struts in J2EE, Ruby on Rails, other frameworks using PHP, flask, node js, react, etc
- ▶ Django is a high-level Python Web framework
  - ▶ Encourages rapid development and clean, pragmatic design.
  - ▶ Build high-performing, elegant Web applications quickly.
  - ▶ Adhere to DRY (Don't Repeat Yourself) principle.

# Django Framework

Web framework for perfectionists with deadlines

- ▶ Main focus
  - ▶ Dynamic and database driven websites
- ▶ DRY
- ▶ Rapid development
- ▶ Follow best practices
- ▶ Free
- ▶ Easy to learn
- ▶ Powerful object-relational mapper (ORM)
  - ▶ Data models defined entirely in Python
- ▶ Automatic admin interface
  - ▶ Eliminates tedious work of creating interfaces to add and update content.
- ▶ Elegant URL design
  - ▶ Flexible URLs

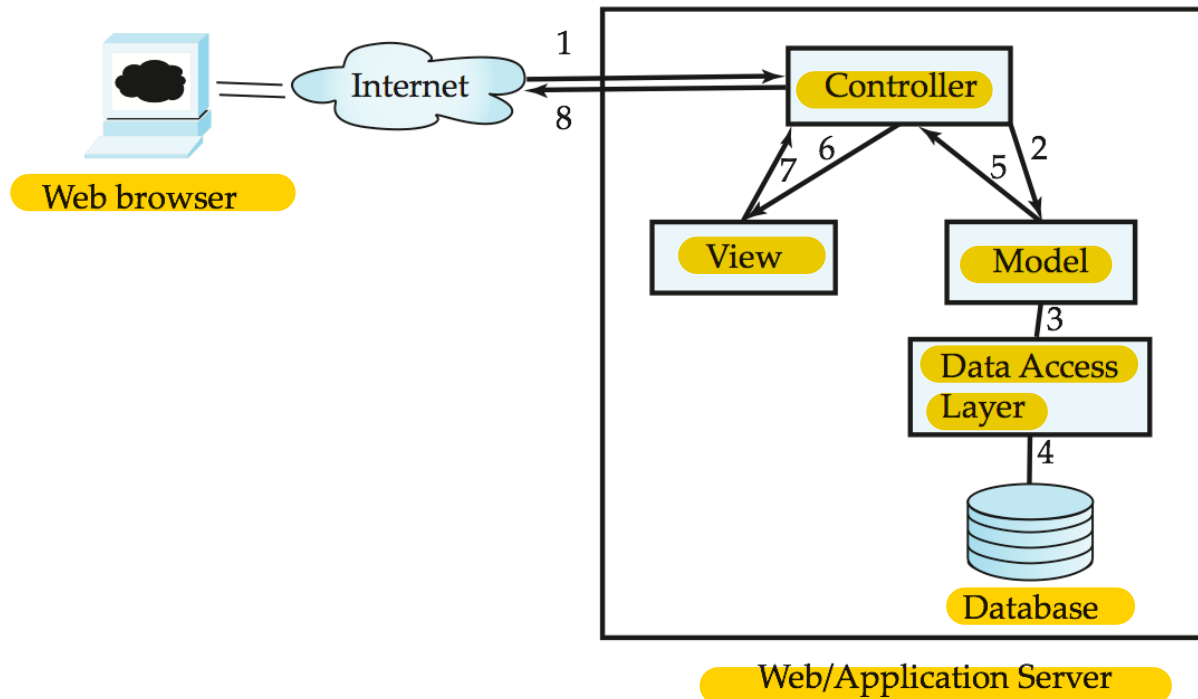
# Sites Using Django

- ▶ Youtube
- ▶ Instagram
- ▶ Spotify
- ▶ Mozilla Firefox
- ▶ National Geographic
- ▶ Pinterest



# MVC

- ▶ **Model-View-Controller (MVC)** architecture
  - ▶ **model:** business logic
  - ▶ **view:** presentation of data, depends on display device
  - ▶ **controller:** receives events, executes actions, and returns a view to the user
- ▶ **business-logic** layer
- ▶ **data access** layer
  - ▶ **interfaces** between **business logic layer** and the **underlying database**
  - ▶ **provides mapping** from **object model of business layer** to **relational model of database**



# Application Architecture

# Django's MTV Architecture

► MVC → MTV

► **Model:**

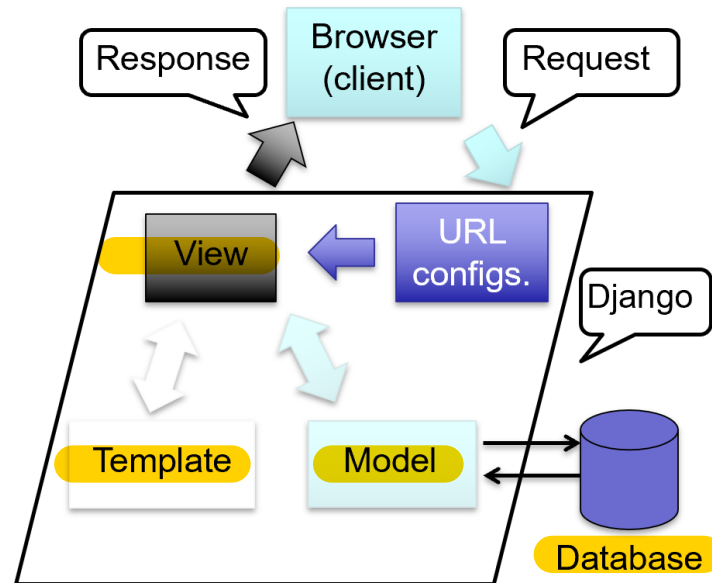
- Deals with data representation/access.

► **Template:**

- Describes how data is represented.
- Same as 'view' in MVC

► **View:**

- Describes which data is presented.
- Same as 'controller' in MVC.



# Project Directory

## Create a new Django project:

### outer mysite/

- container for project; can be renamed.

### manage.py

- command-line utility to interact with your project.

### inner mysite/

- actual python package for project

### \_\_init.py\_\_

- empty file, indicates this dir is a package

### settings.py

- settings/configuration for the project

### urls.py

- URL declarations for the project

### wsgi.py

- entry-point for WSGI-compatible web servers to serve your project

# Settings

- ▶ **Settings.py**: Python module with variables for Django settings.
  - ▶ update DATABASES **'default'** item
  - ▶ **'ENGINE'** : `'django.db.backends.sqlite3'`
    - ▶ `'django.db.backends.postgresql_psycopg2'`,
    - ▶ `'django.db.backends.mysql'`, or
    - ▶ `'django.db.backends.oracle'`
- ▶ By default, following apps are installed
  - ▶ **django.contrib.admin** - The admin site.
  - ▶ **django.contrib.auth** - An authentication system.
  - ▶ **django.contrib.contenttypes** - A framework for content types.
  - ▶ **django.contrib.sessions** - A session framework.
  - ▶ **django.contrib.messages** - A messaging framework.
  - ▶ **django.contrib.staticfiles** - A framework for managing static files.

# References

---

[1] <http://edn.embarcadero.com/article/10343>

---

[2] [www.tutorialspoint.com/http/](http://www.tutorialspoint.com/http/)

---

[3] Python web development with Django by Jeff Forcier, Paul Bissex and Wesley Chun. Addison Wesley 2009.

---

[4] <https://flatworldbusiness.wordpress.com/flat-education/previously/web-1-0-vs-web-2-0-vs-web-3-0-a-bird-eye-on-the-definition/>

---

[5] Database Systems Concepts, 6<sup>th</sup> Edition

---

[6] Data Communications and Networking, 5<sup>th</sup> Edition

---

[7] <https://www.w3schools.com/html/default.asp>

---

[8] <https://techvidvan.com/tutorials/django-project-structure-layout/>

---

[9] [https://www.w3schools.com/html/html\\_forms.asp](https://www.w3schools.com/html/html_forms.asp)

---

[10] Lectures from Dr. Arunita and Dr. Saja

# Extra Slides for Lab 3

## Extra Slides for the Lab: Field Type - Textual Data

Field	Example Values
CharField	"Product Name"
TextField	"To elaborate on my point..."
EmailField	<u>george@site.com</u>
URLField	<u>www.example.com</u>



## Extra Slides for the Lab: Field Type - Numeric and Miscellaneous Data

Field	Example Values
IntegerField	-1, 0, 1, 20
DecimalField	0.5, 3.14

Field	Example Values
BooleanField	True, False
DateTimeField	datetime(1960, 1, 1, 8, 0, 0)

## Extra Slides for the Lab:

### Field Type - Null, Blank, and Many to Many

#### null

If **True**, Django will store empty values as **NULL** in the database.  
Default is **False**.

#### blank

If **True**, the field is allowed to be blank. Default is **False**.

```
models.CharField(max_length=10, blank=True)
```

► For Many to Many relationships, visit:

[https://docs.djangoproject.com/en/4.1/topics/db/examples/many\\_to\\_one/](https://docs.djangoproject.com/en/4.1/topics/db/examples/many_to_one/)