Django
Web Application Framework

Web Application Development

WAD

As architects we need to see through the complexities, abstract away, and design a useful solution.

Top-Down vs. Bottom Up System Architecture

Separates the low level work from the higher level abstractions

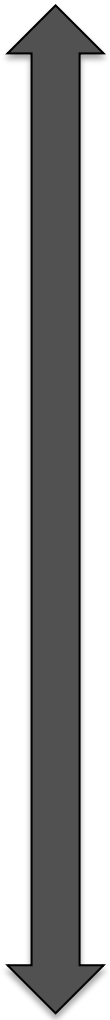
Leads to a modular design

Development can be self-contained (tiered)

Emphasizes planning and system understanding

Coding is late, and Testing is even later

Skeleton code can show how everything integrates



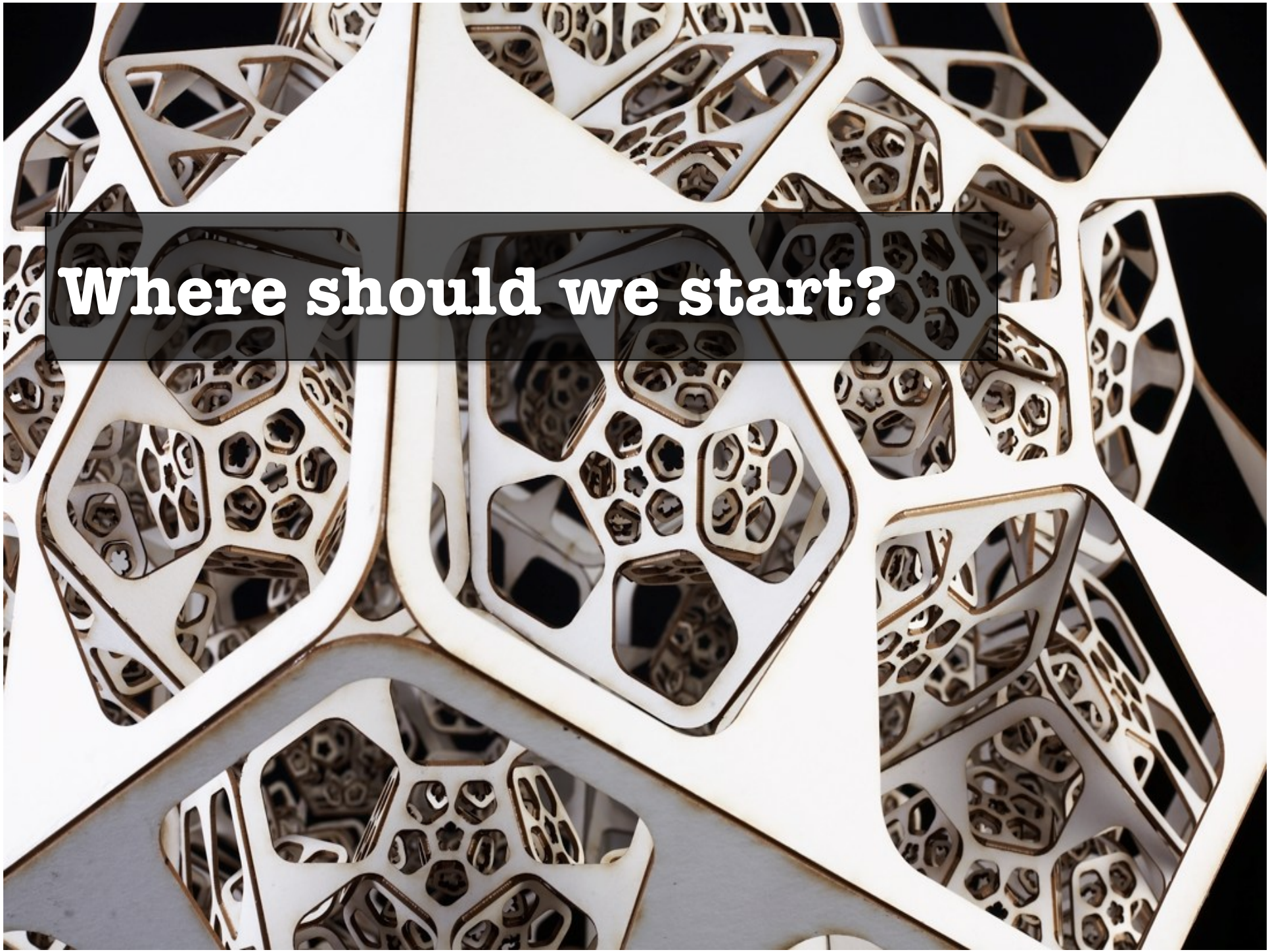
Coding begins early and so Testing can be performed early

Requires really good intuitions to determine functionality of modules

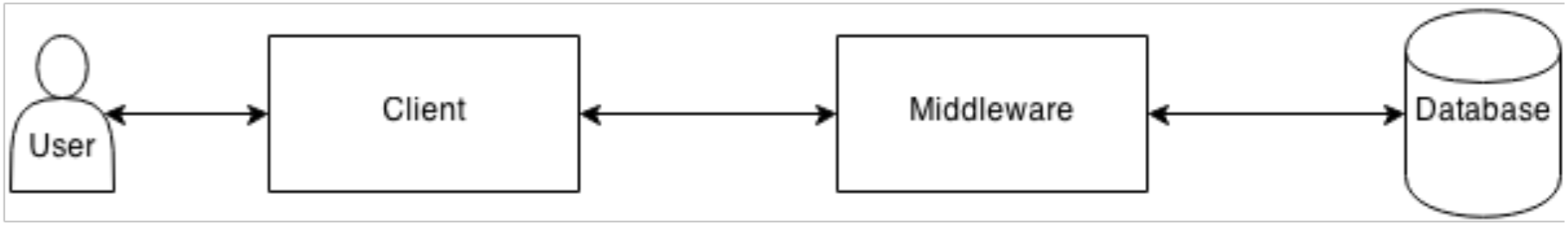
Low level design decisions can have major impact on solutions

Risks integration problems – how do components link together

Often used to add on to existing modules


Where should we start?

High Level System Architecture



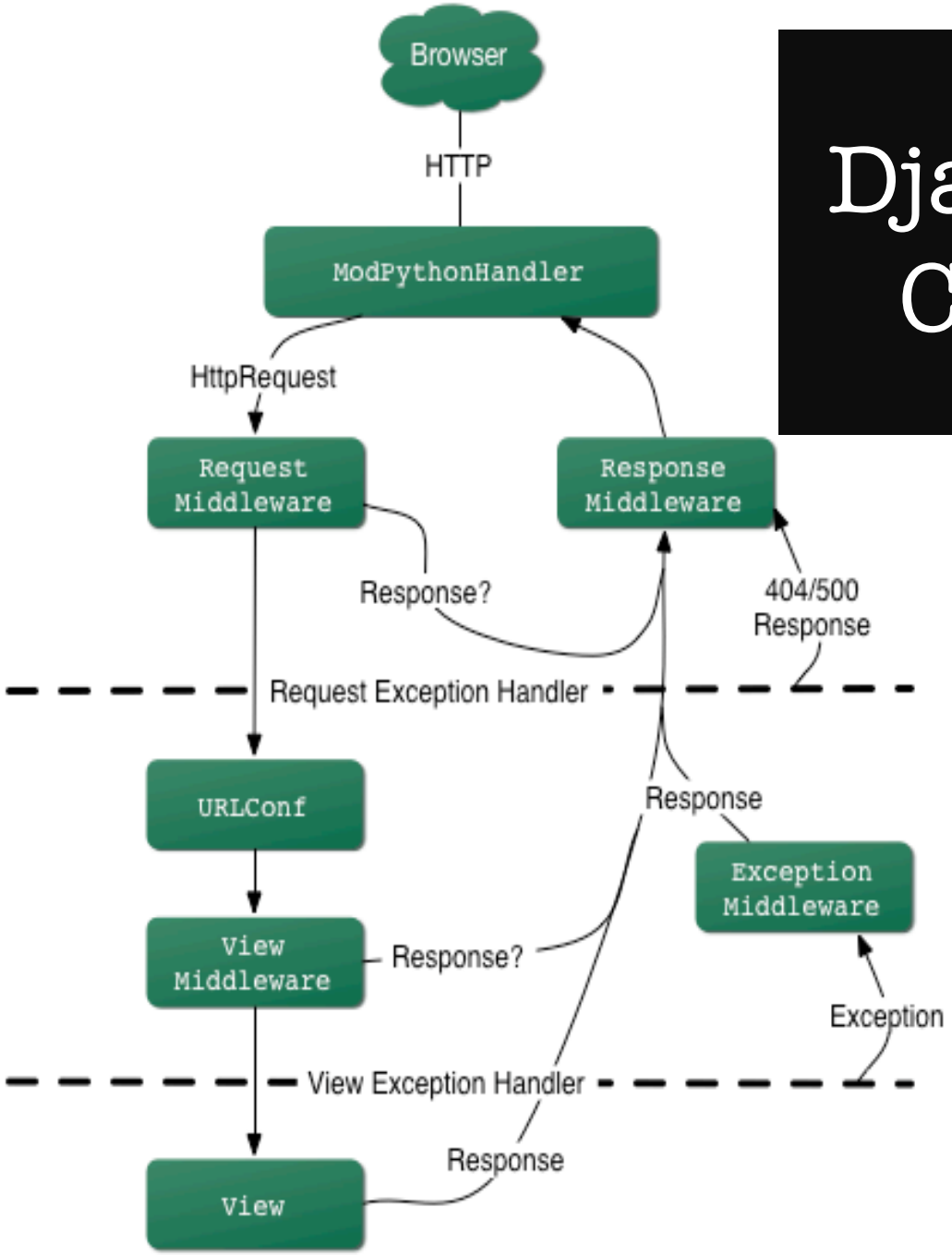
- We need to work out what we are going to build
- And we need to decide what technologies will be used in each box.
- For the middleware, we will be using Django as the WAF for building the application server.


django

Overall Design Philosophy

- **Loose Coupling**
- Less Code
- Quick Development
- **Don't Repeat yourself (DRY)**
- **Explicit is better than implicit**
  - A core Python principle
- Consistency
- See http://docs.djangoproject.com/en/dev/misc/design-philosophies/ for more details and more philosophies

Django Internal Control Flow



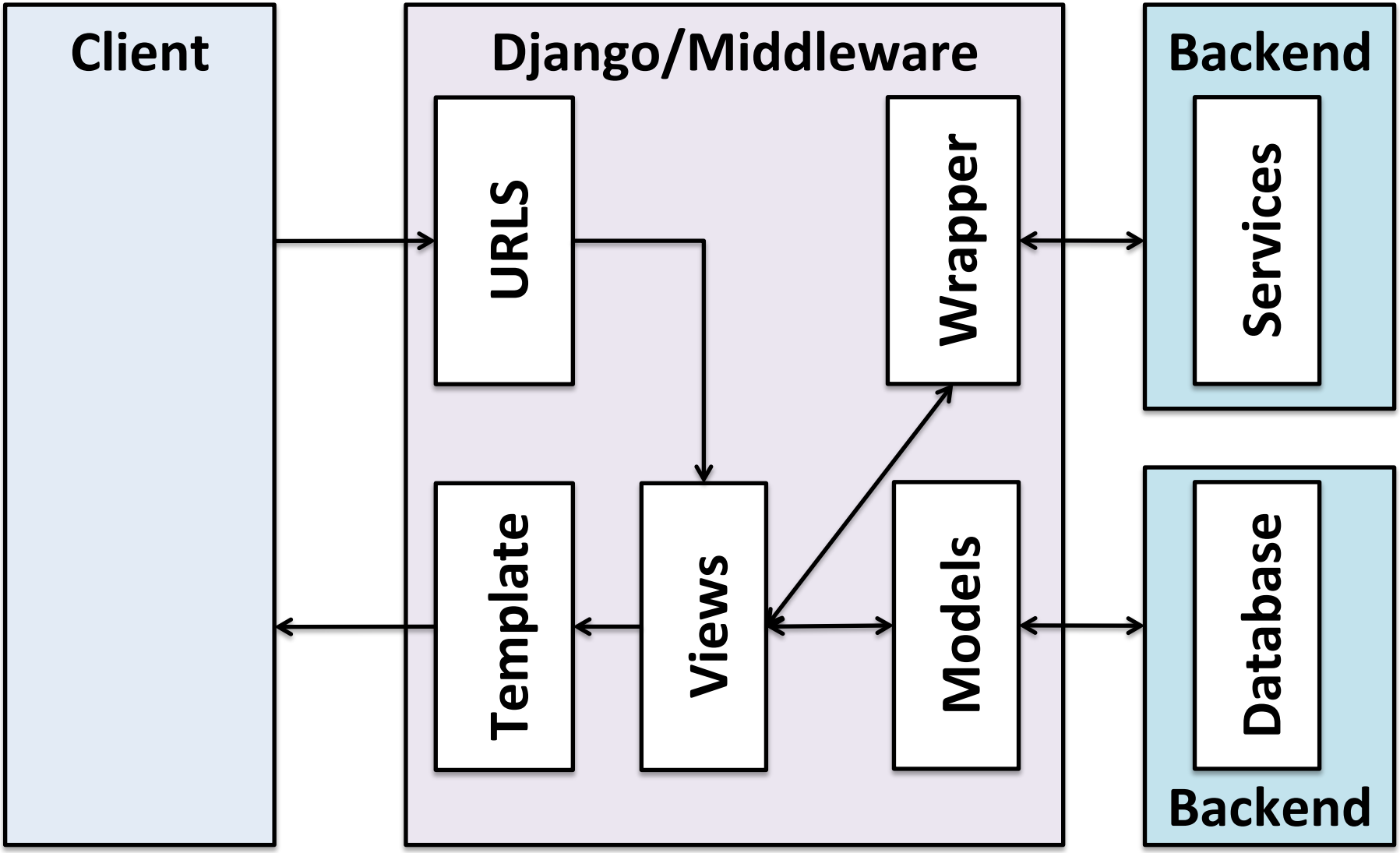
Model Template View

- MVC and MTV are similar…. however,
  - MVC Model maps to MTV Model ☺
  - MVC Controller kind of maps to MTV Views
    - The Django Framework also acts as the controller
    - The MTV View provides the application logic and supplies the data
    - i.e. describes the data that gets presented to the user – but not how it looks…
- MVC View kind of maps to MTV Template
  - MTV Templates presents formats and presents the data

Model Template View

- Separates content from presentation – is where the template comes in.
- MTV Template describes how the data should be presented.
- The machinery associated with the Django framework is essentially the global or over-aching controller that receives and dispatches messages between the key components..

Simplified Internal Flow



Internal Sections/Components

- **Controlling flow (usually in urls.py):**
  - To specify what view function should handled a particular URL (or part of), URL patterns are used to find matches with the URL, and to route this request to the appropriate view.
  - The use of pattern matching means that different instantiations can be handled by a common pattern.
- **Defining Views (usually in views.py):**
  - Views are responsible for handling and processing the specific request, collating the data from databases/ external services, then selecting the template, for the response to be generated.

Internal Sections/Components

- **Providing Templates**
  - The templates mean the response format (html,xml,etc) is decoupled from the data to be presented.
- **Building Data Models (usually in models.py):**
  - The models specify the entities and relationships in the database – these provide an Object Relational Mapping to the actual database tables
  - The framework constructs the database given the models defined.

DJANGO WALKTHROUGH

ADDITIONAL HOMEWORK

Django Tutorial – Polling App

- **Part One**
  - How to create a project and applications
  - How to setup a database and data models via ORM
  - How to use the data models via the API
- **Part Two**
  - How to use and customize the admin site
- **Part Three**
  - How do design URL mappings and working with URLConf
  - How to create a view
  - How to create a template
- **Part Four**
  - Working with Forms
- **Part Five**
  - Test Driven Development