lecture 8c program design
600152 people data and the web
There are two ways of constructing a software design. One way is to make it so simple that there are obviously no deficiencies. And the other way is to make it so complicated that there are no obvious deficiencies.

-- C.A.R. Hoare
last week: mini project

This mini project provides a very high level description of how a search engine works.

Constructing a search engine can be divided into two parts: indexing and querying. In indexing, a program builds an index, which maps a set of keywords to the web pages where the keywords appears. Querying allows a user to send keyword(s) of interest to a query program that would use the index to return a list of web pages in which the keyword(s) exist.

Your task in this project can also be divided into two stages:

1. building the index
2. building a web-based query interface and its query handler.
version 4 last week: searching xhtml pages

querying program

query_title = "two"
query_body = "rabbit"

HTML form

Web Browser

Search results:
- page2.html
requested new features for version 6:

• provide css selections to the user

• persistent user data using cookie:
  to keep what a user have entered in the earlier session. what needs to be persistent:
  • query_title
  • query_body
  • css

• demo
searching xhtml pages: extension

querying program

query_title = "two"
query_body = "rabbit"

HTML form & results

simple.css
fancy.css

Web Browser

cookie <user_id>

persistent_data <user_id>.xml

index_title.pickle
index_body.pickle
input & output

Find the following terms:

one in title

in body

Search

O simple O fancy Change Style

Search results for "" in title and "" in body:

- page1.html
- page2.html
- page3.html

Find the following terms:

in title

in body

Search

O simple O fancy Change Style
separation of concerns

guest

food

servers

waiters

chefs

ingredients
separation of concerns

Imagine a cafe or a restaurant!

Separation of concerns

user

controller

view

model

persistent data (database, file)
web application: choosing paradigm

main form handler:
  main.py

main components:
  mvc_controller.py
  mvc_model.py
  mvc_view.py

modules:
  m_cookies.py
  m_index.py
  m_xxx.py
  ...

browser

html + css
manage user input
pass on the input to the model
call appropriate action in the model

controller

user

html + css

view

model

persistent data (database, file)
controller

- query_title
- query_body
- css

SEARCH or CHANGE STYLE

controller

model

- model.do_search(query_title, query_body)
- model.do_css(css)
user

html + css

controller

execute application logic
manage persistence data

model

text

view

model

persistent data (database, file)
The diagram illustrates a model-view-controller (MVC) architecture.

- **User** interacts with the **controller**.
- The controller orchestrates the flow:
  - Processes user input or requests.
  - Communicates with the **model** for data.
  - Renders the model to the **view**.

- **Model** manages the **persistent data** (database, file).
- **View** displays data to the user (manages presentation).

**View** components:
- HTML + CSS

**Controller**:

**Model**:
- Persistent data

**View**:
- Render the model display to the user.
resources

• Original paper on MVC
  http://st-www.cs.illinois.edu/users/smarch/st-docs/mvc.html
• Sun's Model View Controller
  http://java.sun.com/blueprints/patterns/MVC.html