Template approach -> embed the java inside HTML
MVC approach(Model View Controller) -> separate all file namely html, css etc for good programmer

CS144: Model-View-Controller (MVC)

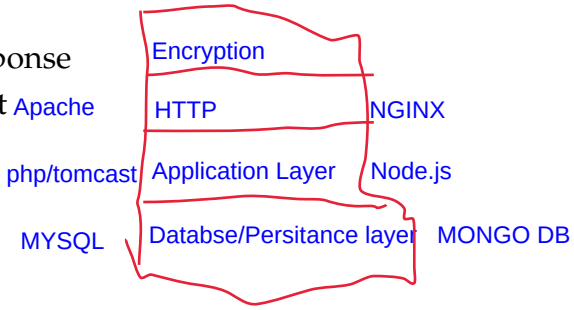
Types of Web servers

- Static site
  - Web was originally about downloading static content from Web servers
    - Q: What should a Web site do to serve a static page for a request?
  - HTTP server (say, Apache) + Filesystem
    - Configure the mapping from a URL to a file
      - E.g., "DocumentRoot /var/www/html/" in Apache
- Dynamic site
  - A large part of Web content is dynamically generated now
    - Q: What happens when Google gets a query? What does it do?
    - Q: What happens when you log in Amazon?
    - Q: How does a server identify which process to run to generate the corresponding content given a request?
      - e.g., web.xml file on Tomcat

Four layers of Web site

4 layer of HTTP need to generate content dynamically

- Encryption layer: encrypt transport
- HTTP layer: interpret request and serve response
- Application layer: generate dynamic content
- Storage/Data layer: store and retrieve data



Generating dynamic pages

How can we generate a dynamic Web page?

- Example: Hello, John! at http://oak.cs.ucla.edu/classes/cs144/examples/hello.html

1. Programmatic approach: Write a program that prints out the HTML page!

   Example: Java Servlet for "Hello, John!"

```
protected void doGet(HttpServletRequest request,
                     HttpServletResponse response)
    throws ServletException, IOException
{
    PrintWriter out = response.getWriter();
    out.println("<html>");
    out.println("<head><title>Hello</title></head>");
    out.println("<body>Hello, " + request.getParameter("
       first_name") + "!");
    out.println("</body>");
    out.println("</html>");
    out.close();
}
```

2. Template approach: Write an HTML page that allows simple "variable substitution"!

   Example: Java ServerPages (JSP)

```
<html>
<head><title>Hello</title></head>
<body>Hello, <%= request.getParameter("first_name") %>!
</body>
</html>
```

- Q: What are the problems of the two approaches?
- Notes:
  - Even for template approach, once complex code gets embedded inside, the page gets ugly and becomes difficult to maintain
  - Code "ownership"
    - Often, page design is done by designers, while app coding is done by developers.
    - Who "owns" the above pages?
    - When multiple people "own" the same page, "conflicts" arise
      - Can we separate page design from programming logic?

Model-View-Controller (MVC) Pattern

- Most programs have to deal with data, application logic, and final result presentation
  - Data may be stored in a file or database engine, and locally or remotely.
  - Application logic is often independently of where and how data is stored and retrieved
  - The "result" from the application may be presented in different ways depending on the device and/or user
  - Mixing these three "independent" components into one gigantic spaghetti code is often not a good idea
- Develop application into three modular components!
  - *Model*: deals with data storage and access
  - *View*: deals with result presentation
  - *Controller*: deals with "application logic"
  - "Code" for each component may be "owned" by different people
    - e.g., model: DB engineer, controller: app developer, view: UI designer
- Example

  In Java servlet:

```
/*  CONTROLLER */
protected void doGet(HttpServletRequest request,
                     HttpServletResponse response)
    throws ServletException, IOException
{
    // retrieve, update, process data
    ...

    // construct data model
    request.setAttribute("data1", XXX);

    // dispatch data model to the view
    request.getRequestDispatcher("/index.jsp").forward(request
       , response);
}
```

```
/*  MODEL  */
User getUser(int userid)
{
    // retrieve and return the user
}
```

In index.jsp:

```
/* VIEW */
<html>
<head><title>Demo</title></head>
<body>Your data: <%= request.getAttribute("data1") %></body>
</html>
```

- Specialized "tags" exist to add simple logical constructs, such as loop, to the view
  - Example: Java Standard Tag Libraries (JSTL)

```
<%@ taglib uri="http://java.sun.com/jstl/core" prefix="c"
   %>
<html>
<title>Contacts</title>
<body>
    <table>
        <c:forEach var="contact" items="${contacts}">
            <tr>
                <td>${contact.name}</td>
                <td>${contact.email}</td>
            </tr>
        <c:forEach>
    </table>
</body>
</html>
```

- Most frameworks supports separation of Model, View, and Controller
  - Java Struts, Python Django, ASP .NET, Ruby on Rails, ...