Principles of Information and Database Management
198:336
Week 9 – Apr 4

Matthew Stone

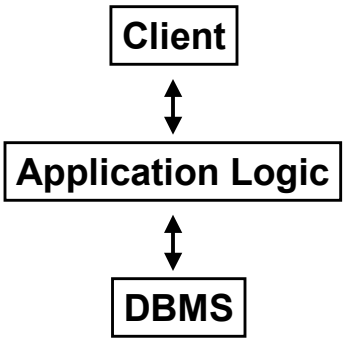
Data over the Web

Three-tier architectures

Ingredients of three-tier architecture

Information retrieval: text as data

Design of Network Apps



Client

Presentation layer

- Allows users to make requests
- Allows users to provide input
- Allows users to see results

Application Logic

Layer for control

- What should happen with user input?
- How does control execute across steps?
- What data should be accessed, recorded, and presented?
- How should interaction proceed?

DBMS

Database layer

- The stuff we've been learning about!

Example of Breakdown

User “authentication”

- User is challenged for login and password
- System checks whether this is OK
- Grants the user access or gives suitable error

Client Layer

Entering information

- Prompts the user for login and password
- Gives the user places to specify them
- Gives the user a place to hit OK

DBMS

Stores login information as a table

- Valid login names
- Encrypted passwords

Application layer

Requires login in client layer

Gets login information from client

- Encrypts password

Checks if login, encryped password in DB

Decides what to do next

Splitting up the Design

Client

- Runs on a web browser
- Generic, lightweight interface mechanism
- Gets (X)HTML description of interaction
  - Using HTTP(S) protocol
- Carries out that interaction with user

Splitting up the design

Application layer

- Part of a web server
  - Accepts and responds to HTTP(S) requests
- Implemented in generic language
  - Java servlets, Javascript, PHP, Perl
- Connects to DBMS however it likes

Splitting up the design

DBMS

- Handles generic information functionality
- Storage, backup, concurrency, scale, security…

Example, idealized

Step 1:

- User at machine

  home.isp.net

  asks in web browser for page

  http://buy.mystuff.com
- Client sends HTTP request to server

Example, idealized

Step 2:

- Application logic runs as part of web server running on the machine buy.mystuff.com

  This happens by running a file for the root of this interaction
- Application logic decides user needs to log in
- Application logic sends login page back to machine at home.isp.net

Example, idealized

Data now comes back to home.isp.net

```
<html>
<form action="https://buy.mystuff.com/secret"  method="post">
Account:
<input type="text" name="account" />
Key:
<input type="password" name="key" />
<input type="submit" />
</form>
</html>
```

Example, idealized

Home.isp.net creates the interaction described by this data in a browser

The user types, edits, clicks, etc.

The result is a new request that goes back to buy.mystuff.com

Example, idealized

Now the login logic runs at buy.mystuff.com

- We get the values the user typed as parameters – call them A and K
- We open a connection to the database, which is a server running at dbms.mystuff.com
- We create a safe SQL query asking whether an entry of (A, encrypt(K)) exists in table authorized
- We get an answer, yes or no.

Example, idealized

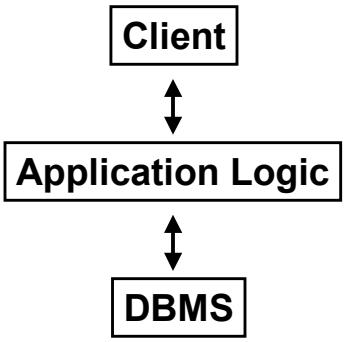
If authorized, we send back one interaction
- We construct a new SQL query using A to access secret information
- We format it as HTML

If unauthorized, we send back another
- We construct a new HTML page
- Explaining error
- Offering another chance to log in?

Example, idealized

Finally, the user's browser at home.isp.net carries out the last step of the interaction

Design of Network Apps



Design issues

User experience

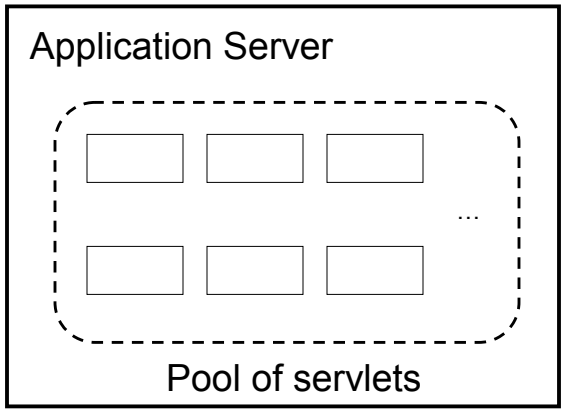
- Latency
- Richness
- Adaptivity

Design issues

Infrastructure effectiveness

- Trust
- Data Integration
- Scalability
- Modularity

Middle Tier – Servlets



Java Servlets for Tomcat

Overview:

- Define new class with either of two methods: doGet and doPost
- Get parameters from request
- Check they're safe
- Prepare an SQL query
- Set the ? elements in the prepared query
- Execute the query
- Write out the results through response

Java Servlets for Tomcat

Implement class HttpServlet

```
public class ReadUserName extends HttpServlet {
   public void doGet(HttpServletRequest rq,
                     HttpServletResponse rs) throws
   ServletException, IOException {
        …
   }
   public void doPost(…) { … }
}
```

Useful methods

Finding stuff out from request rq

String rq.getParameter(String)

Eg.

String account = rq.getParameter(“account”);

JDBC Stuff

```
String query = “SELECT R.cash “ +
                “FROM Relationship R “ +
                “WHERE R.account = ?”;
PreparedStatement ps =
  conn.prepareStatement(query);
ps.setString(1, account);
ResultSet r = ps.executeQuery();
```

Finally

Writing stuff out to a response rs

```
PrintWriter out = rs.getWriter();
out.println(String);
```

Why not this?

```
String query = “SELECT R.cash” +
                “ FROM Relationship R” +
                “ WHERE R.account = “ +
                account;

Statement s = conn.createStatement();
ResultSet r = s.executeQuery(query);
```

Get and URL Encoding

When you type v1 as the value of n1
and v2 as the value of n2
the browser makes a load request for:
http://request.com?n1=v1&n2=v2

this is a URL, and it requires us to
“encode” n1,v1,n2 and v2

Encoding

```
import java.net.URLEncoder;
import java.net.URLDecoder;

String s’ = URLDecoder.decode(s, “UTF-8”);
String s = URLEncoder.encode(s’, “UTF-8”);
```

Same encoding happens with post

But you don't construct a URL

You pass data "silently" as part of the http header.