

3-Tier Web Architectures

Ramakrishnan & Gehrke, Chapter 7

www.w3schools.com

www.webdesign.com

...

Components of Data-Intensive Systems

■ Presentation

- Primary interface to the user
- Needs to adapt to different display devices (PC, PDA, cell phone, voice access, ...)

■ Application (“business”) logic

- Implements business logic (implements complex actions, maintains state between different steps of a workflow)
- Accesses different data management systems

■ Data management

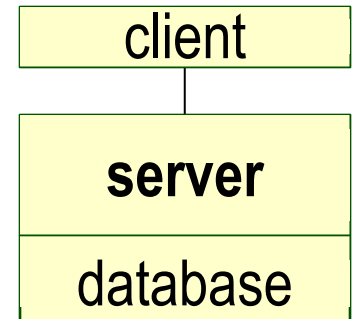
- One or more standard database management systems

- system architecture determines whether these three components reside on a single system (“tier) or are distributed across several tiers

Client-Server Work Division

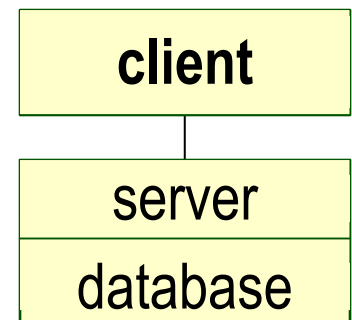
■ Thin client

- Client implements only GUI
- Server implements business logic and data management

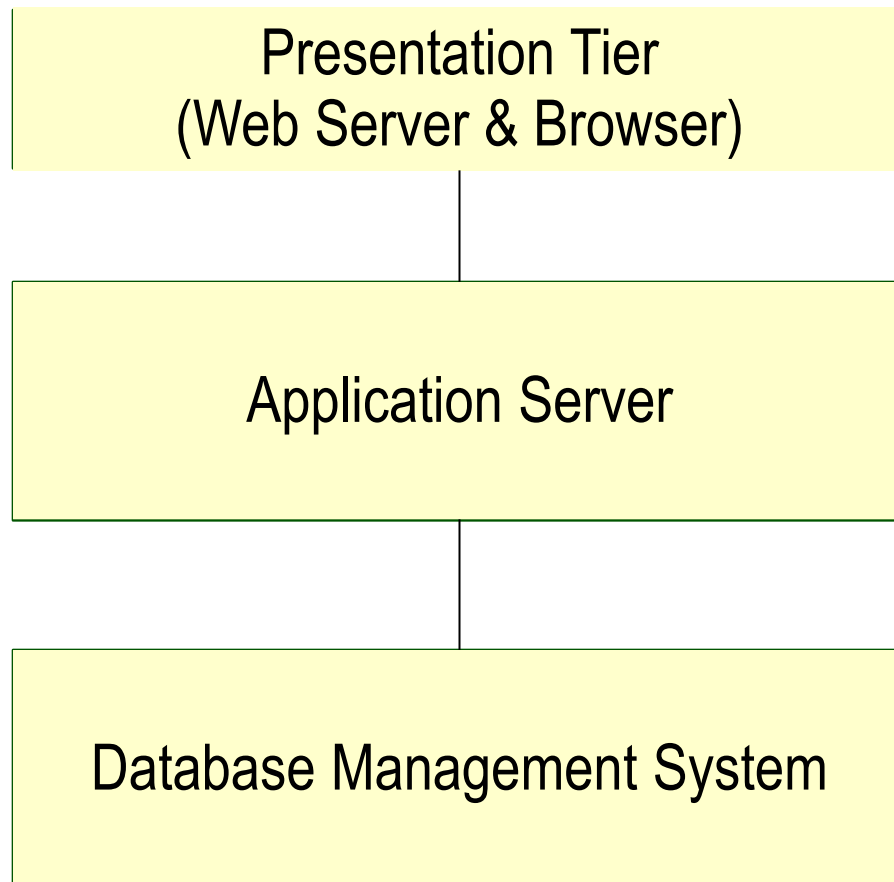


■ Thick client

- Client implements GUI & business logic
- Server implements data management



Technologies



HTML, CSS, Javascript
Ajax
Cookies

JSP, Servlets, CGI, ...

Tables, XML, JSON, ...
Stored Procedures

The Presentation Tier

- Recall: Functionality of the presentation tier
 - Primary **interface** to the user
 - Needs to adapt to different display **devices** (PC, PDA, cell phone, voice access?)
 - For efficiency, **simple functionality** (ex: input validity checking)
- Mechanisms:
 - HTML Forms
 - Dynamic HTML / JavaScript
 - CSS

JavaScript

- Goal: Add functionality to the presentation tier
- Sample applications:
 - Detect browser type and load browser-specific page
 - Browser control: Open new windows, close existing windows (example: pop-ups)
 - Client-side interaction (conditional forms elements, validation, ...)
- JavaScript optimal for Web browser because:
 - Built-in engine – always available, fast
 - Operates directly on “browser brain” = DOM

The Middle (Application) Tier

- Recall: Functionality of the middle tier
 - Encodes business logic
 - Connects to database system(s)
 - Accepts form input from the presentation tier
 - Generates output for the presentation tier
- Mechanisms:
 - CGI: Protocol for passing arguments to programs running at the middle tier
 - Application servers: Runtime environment at the middle tier
 - Servlets: Java programs at the middle tier
 - PHP: Program parts in schematic documents (see earlier)
 - How to maintain state at the middle tier

```

/**
 * return a full HTML page, as opposed to fragments
 */
private String composeFullPage() throws ConnectionFailedException, ConfigurationExcept

```

```

/**
 * return a full HTML page, as opposed to fragments
 */
private String composeFullPage() throws ConnectionFailedException, ConfigurationExcept

```

```

/**
 * return a full HTML page, as opposed to fragments
 */
private String composeFullPage() throws ConnectionFailedException, ConfigurationExcept

```

Where to Keep Application State?

- Client-side state
 - Information is stored on the client's computer in the form of a cookie
- Hidden state
 - Information is hidden within dynamically created web pages
- Server-side state
 - Information is stored in a database, or in the application layer's local memory

Server-Side State

- Various types of server-side state, such as:
 - 1. Store information in a database
 - Data will be safe in the database
 - BUT: requires a database access to query or update the information
 - 2. Use application layer's local memory
 - Can map the user's IP address to some state
 - BUT: this information is volatile and takes up lots of server main memory

Client-side State: Cookies

- Cookie = (Name, Value) pair
- Text stored on client, passed to the application with every HTTP request
 - Lifetime can be preset (eg, 1 hour)
 - Can be disabled by client
 - wrongfully perceived as "dangerous", therefore will scare away potential site visitors if asked to enable cookies
- Advantages
 - Easy to use in Java Servlets / PHP
 - simple way to persist non-essential data on client even when browser has closed
- Disadvantages
 - Limit of 4 kilobytes
 - Users can (and often will) disable them
- Usage: store interactive state
 - current user's login information
 - current shopping basket
 - Any non-permanent choices user has made

Hidden State

- overcome cookie disabling
- Can “hide” data in two places:
 - Hidden fields within a form
 - path information
- Requires no client or server “storage” of information
 - state information passed inside of each web page – “on the wire”

Hidden State: Hidden Fields

- Declare hidden fields within a form:

```
<form method='GET' action='http://.../input.php'>  
  <input type='hidden' name='basketid' value='PJyACJt4eYmWrcp' />  
  <input name='wordKey' type='text'>  
  <input type='submit' value='Go'>  
</form>
```

- Advantages

- Users will not see information (unless they view HTML source!)

- Disadvantages

- If used prolifically, it's a performance killer
- Works only in presence of forms

Hidden State: KVP Information

- Information stored in URL GET request:
 - `http://server.com/index.htm?user=jeffd`
 - `http://server.com/index.htm?user=jeffd&preference=pepsi`
- Parsing field in Java:
 - `javax.servlet.http.HttpUtils.parserQueryString()`
- Advantages
 - Independent from forms
- Disadvantages
 - Limited to URL size (some kB)

Multiple state methods

- Typically all methods of state maintenance are used:
 - User logs in and this information is stored in a **cookie**
 - User issues a query which is stored in the **URL** information
 - User places an item in a shopping basket **cookie**
 - User purchases items and credit-card information is stored/retrieved from a **database**
 - User leaves a click-stream which is kept in a **log** on the web server (which can later be analyzed)

Some Web Service Security Hints

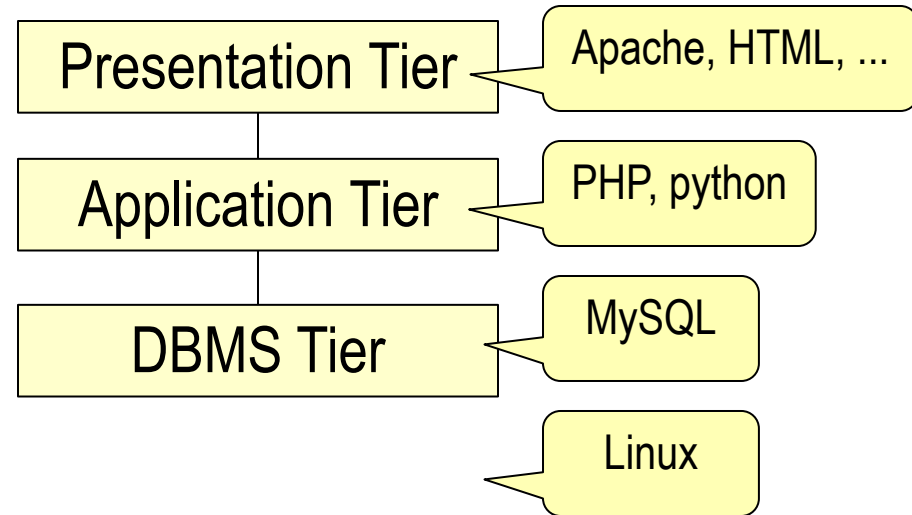
- Never use anything blindly that comes from client side
 - don't assume that JavaScript code has been executed
 - double check cookies on server
 - don't trust hidden fields contents
- never assume anything!
 - set defaults (define in a central place!)
- Clear state after request response
- as with any API: clean, defensive programming
 - perform standard plausi checks:
admissible number ranges, empty strings, max string lengths!
- *Be paranoid !!!*

Summary: 3-Tier Architectures

- Web services commonly architected as having 3 components
 - Presentation / application / data management tier
- Application tier needs most implementation flexibility
 - Rich choice of platforms (Java servlets, PHP, ...), each with tool support
- To maintain state, use:
 - Hidden form fields, hidden paths, cookies, server store, ...
- *For every aspect & component, security is an issue!*

DB Module Relevance

- In the project: LAMP stack
 - Linux, Apache, MySQL, PHP/Python



- Alternatives:
 - MERN stack:
 - *MongoDB: A document database*
 - *Express: web framework for Node.js*
 - *React: JavaScript front-end library*
 - *Node.js: JavaScript runtime bringing JavaScript to the server*
 - MEAN stack
 - *MongoDB, Express.js, AngularJS, Node.js*