DIPARTIMENTO DI INGEGNERIA INFORMATICA
AUTOMATICA E GESTIONALE ANTONIO RUBERTI

SAPIENZA
UNIVERSITÀ DI ROMA

Progetto di Applicazioni Software

Laurea in Ingegneria Informatica ed Automatica
Laurea in Ingegneria dei Sistemi Informatici
Laurea Magistrale in Ingegneria Informatica

Credits: *parte del materiale utilizzato durante il corso riprende materiale preparato negli anni precedenti dai docenti Andrea Calì, Domenico Lembo, Antonella Poggi, e dai tutor Fabio De Rosa, Luca de Santis, Ruggero Russo, Massimiliano de Leoni, Claudio Di Ciccio, che si ringraziano*

Docente

Massimo Mecella

Dipartimento di Ingegneria Informatica Automatica e Gestionale Antonio Ruberti

SAPIENZA – Università di Roma

Via Ariosto 25, 00185 Roma

II piano, stanza B209

E-mail: mecella@dis.uniroma1.it

Web: http://www.dis.uniroma1.it/~mecella/

Contattare il docente

- Durante il periodo di lezione, contattare il docente al termine delle lezioni del giovedì o del venerdì (non del lunedì) o durante il **ricevimento studenti**

- Calendario del ricevimento (costantemente aggiornato) su `http://www.dis.uniroma1.it/~mecella/ricevimento.htm`
  - Controllare sempre prima che non ci siano modifiche dell'orario di ricevimento, etc.
  - Le modifiche vengono segnalate al massimo entro 3 ore prima dell'inizio dello stesso

- Dopo il termine del ciclo di lezioni continuerà il ricevimento studenti

Contattare il docente

- Posta elettronica e telefono devono essere utilizzati ESCLUSIVAMENTE in casi eccezionali ...
- ... NON
  - per richiedere orari di ricevimento (guardare la pagina web)
  - correzioni di esercizi (orario di ricevimento)
  - domande su quando e come si fanno gli esami (guardare la pagina web)
  - domande sui contenuti del corso (orario di ricevimento)
  - prenotazioni al ricevimento (si viene di persona e si fa la fila come tutti gli altri)

Pagina del corso

http://www.dis.uniroma1.it/~mecella/didattica/2012/ProgAppSw/

(disponibile entro il 10 marzo)

- Guardare sempre la "bacheca degli avvisi" per informazioni dell'ultima ora
- Guardare le date degli appelli
- Materiale didattico disponibile appena pronto
- Informazioni varie sempre aggiornate

Organizzazione del corso

- Modulo da 6 crediti
  - ~ 60 ore di lezione/esercitazione (in aula)
  - ~ 40 ore di esercitazione (in laboratorio)

- Inizio lezioni: Lunedì 5 Marzo 2012
- Termine lezioni: Venerdì 1 Giugno 2012

- **Orario del corso**
  - Lunedì 10:15 – 11:45 & 12:00 – 13:30
    - SPV, aula 8
  - Giovedì 14.00 – 15.30
    - SPV, aula 8
  - Venerdì 14:00 – 15:30 & 15:45 – 17:15
    - Laboratorio di via Tiburtina 205, aula 15

Prerequisiti

- Fondamenti di Informatica I
- Fondamenti di Informatica II
- Progettazione del Software
  - progettazione UML
  - realizzazione in Java
- Basi di Dati
  - modello relazionale
  - il linguaggio SQL
  - progettazione concettuale
  - progettazione logica

Fortemente consigliati

- Sistemi Operativi
- Reti di Calcolatori
  - architettura Internet
  - network programming in Java

Esame / Prova finale
Modalità generali

- Al fine di sostenere l'esame o presentare la prova finale, lo studente (anche in gruppo) deve sviluppare un progetto e consegnare un elaborato che consiste in:
  - una base di dati
  - un'applicazione web che ha accesso alla base di dati (e ad eventuali risorse esterne, ad es., servizi online)
  - la documentazione relativa alla base di dati e alla applicazione

- Ulteriori requisiti dell'applicazione saranno resi noti durante lo svolgimento del corso
- Durante la discussione del progetto, il docente verifica la conoscenza generale dei contenuti del corso attraverso domande, ispirate (ma non limitate) al progetto in discussione

Modalità generali

- **Assegnazione**
  - lo studente / gruppo proporrà un'idea progettuale al docente. L'idea progettuale verrà raffinata tramite un processo di interazione tra studente / gruppo e docente fino all'approvazione del testo finale.
- Lo studente / gruppo dovrà dimostrare (**mediante una demo**) che il **progetto realizzato funzioni correttamente**

Documentazione di progetto e materiale didattico

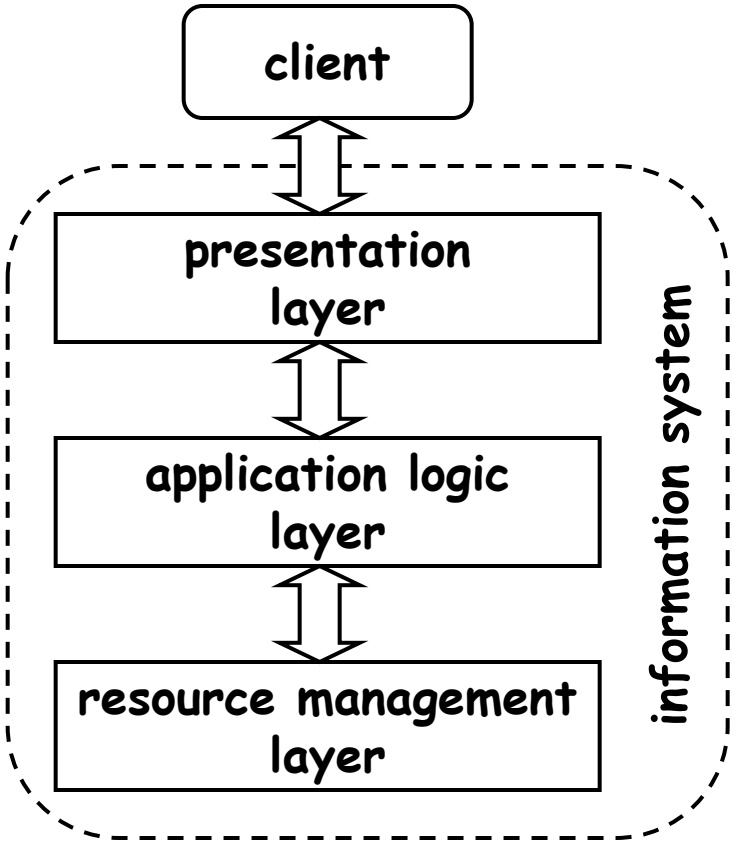
- Dettagli sulla documentazione di accompagnamento al progetto verranno forniti durante il corso e sulla pagina del corso
  - Al variare della tipologia del progetto potrà essere richiesta la consegna di materiale aggiuntivo
- Il software e la base di dati del progetto devono essere consegnati come macchina virtuale (VMWare Player or VirtualBox)

Statistiche di inizio corso

- Ca. 75 discenti
  - 60 iscritti a Laurea in Ingegneria Informatica e Automatica
  - 15 iscritti a Laurea Sistemi Informatici
- Ca. 20 hanno inserito il corso nel loro piano di studi ed intendono sostenere l'esame
- I rimanenti 55 seguono il corso perchè interessati solamente all'assegnazione del progetto per la prova finale

NOZIONI DI BASE DI ARCHITETTURE SOFTWARE

Layer di un sistema informativo



A livello concettuale un sistema è progettato in termini di tre diversi componenti funzionali (layer)

1. Presentazione
2. Logica dell'applicazione
3. Gestione delle risorse

Client di un sistema informativo

- E’ un qualsiasi utente o programma software che vuole effettuare un’operazione sul sistema
- I client interagiscono con il sistema attraverso il presentation layer

Presentation Layer

- E' il livello del sistema che gestisce la comunicazione con le entità esterne al sistema stesso (client)
- Comprende le componenti che si occupano di presentare l'informazione verso i client, e che consentono ai client di interagire con il sistema per sottomettere operazioni ed ottenere risultati
- I client possono essere completamente esterni ed indipendenti dal presentation layer: nei sistemi accessibili tramite web browser che visualizzano pagine HTML, il presentation layer è costituito dai moduli del web server che concorrono a creare i documenti HTML (ad es., Java servlet), mentre il browser è il client
- In altri casi il client ed il presentation layer possono essere fusi insieme: spesso esiste un programma che assolve ad entrambi i compiti

Application Logic Layer

- E' il livello del sistema che si occupa del processamento dei dati necessario per produrre i risultati da inoltrare al livello di presentazione
- Un programma che implementa le operazioni legate ad un prelievo su un conto corrente bancario, o la sequenza di passi da compiere per effettuare un acquisto on-line sono esempi di logica applicativa di un sistema
- Il livello della logica applicativa è anche chiamato processo di business, insieme delle regole di business, o semplicemente server (in questi casi il sottostante livello è rispettivamente chiamato persistence storage, business objects, o database)

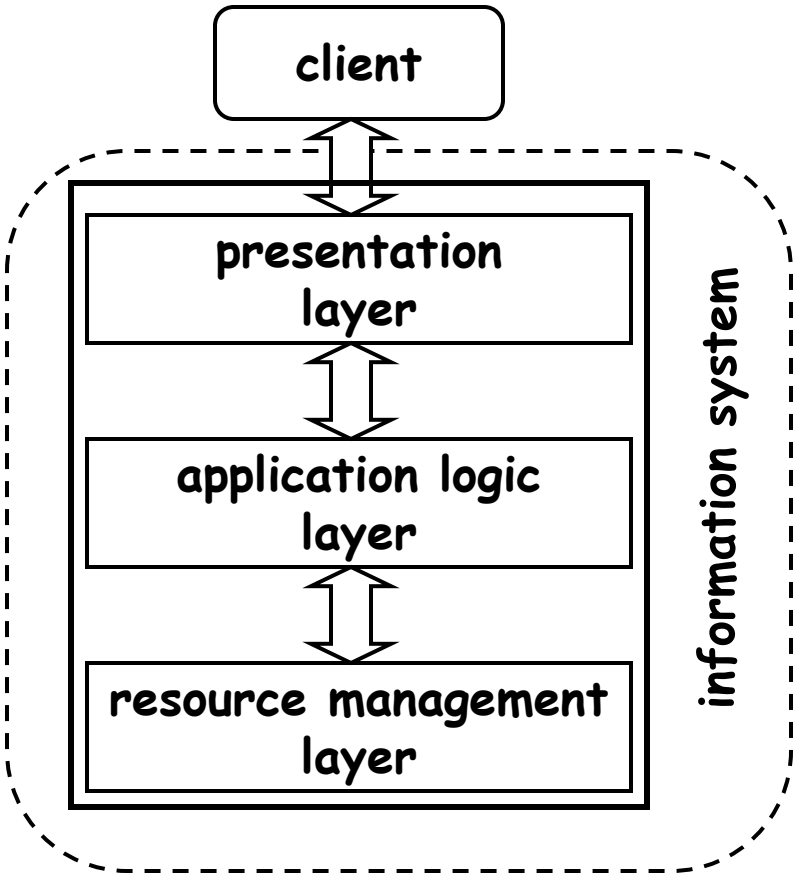
Resource Management Layer

- E' il livello che gestisce i dati che sono necessari al funzionamento dell'intero sistema
- I dati possono risiedere su una base dati, un file system, o altri contenitori di informazioni
- In linea di principio, il livello di gestione delle risorse gestisce le differenti sorgenti di dati che fanno parte del sistema informativo, indipendentemente dalla loro natura
- Nel caso in cui esso è implementato tramite un DBMS, è detto semplicemente data layer
- Secondo un'accezione più generale, questo può includere qualsiasi sistema in grado di fornire informazione

Architettura dei sistemi informativi

- Gli strati discussi finora sono astrazioni concettuali che separano le funzionalità di un sistema informativo
- Nell'implementazione dei sistemi reali, questi strati possono essere combinati e distribuiti in diversi modi
- Quando consideriamo l'implementazione ed il dispiegamento di un sistema informativo, facciamo riferimento ai livelli del sistema con il termine tier
  - Tier = modulo software che implementa uno o più layer
- Un tier non può essere realizzato su più di una macchina, ma due tier, per essere distinti, non devono necessariamente risiedere in due macchine distinte !!

Architettura 1-tier



L'architettura 1-tier combina tutti i livelli concettuali in un un unico tier

Rispecchia l'architettura hw basata su mainframe tipica dei primi calcolatori elettronici

Gli utenti accedevano tramite terminali non intelligenti (in genere schermo e tastiera)

Architettura 1-tier – svantaggi

- I tre livelli sono gestiti da un unico modulo, che risiede quindi in un'unica macchina
  - L'intero livello di presentazione risiede sulla stessa macchina, che si prende carico di controllare ogni aspetto dell'interazione con il client
    - e.g., calcolo centralizzato delle visualizzazioni grafiche delle interfacce

  → richiede molta potenza di calcolo

  → supporta un basso numero di utenti

- Non vengono definite application program interface (API) che possano facilitare l'interazione con altri sistemi

  → poca portabilità del sistema

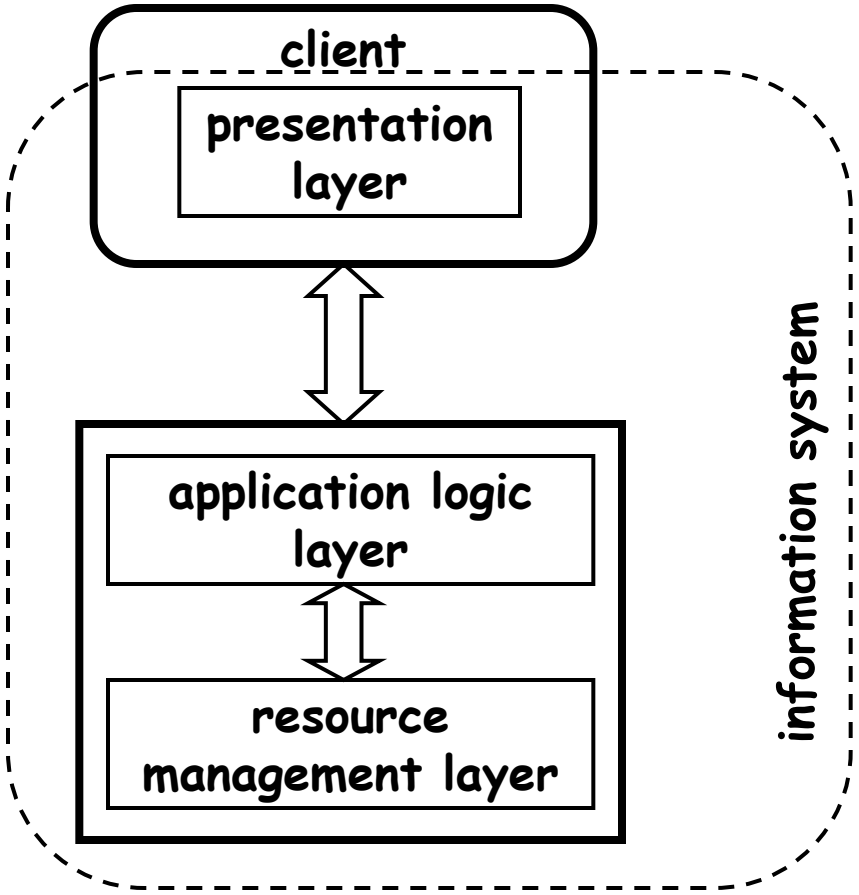
  → difficile da manutenere

  → difficile da aggiornare

Architettura 1-tier – aspetti positivi

- Tutto risiede in un'unica macchina
  - →Non ci sono costi di comunicazione
  - →Gestire e controllare le risorse è più semplice
- Client non intelligenti
  - → Bassi costi di impiego
- Il progettista è libero di fondere i livelli concettuali
  - → aumenta l'efficienza del sistema, design ottimizzato

Architettura 2-tier



- Il sistema informativo è partizionato tra due tier: i client hanno la possibilità di processare ulteriormente l'informazione fornita dal server
- Si è affermata con il diffondersi di PC e workstation e con lo sviluppo di nuove tecniche software per i sistemi distribuiti (ad es., RPC)
- Introduce il concetto di API (Application Program Interface) – interfaccia per invocare il sistema dall'esterno
- Il tier nel client
  - gestisce il livello di presentazione
  - invoca le funzionalità dell'applicazione
- I client si dividono in *thin client* (con soli compiti di presentazione) e *fat client* (inglobano parte della logica dell'applicazione)

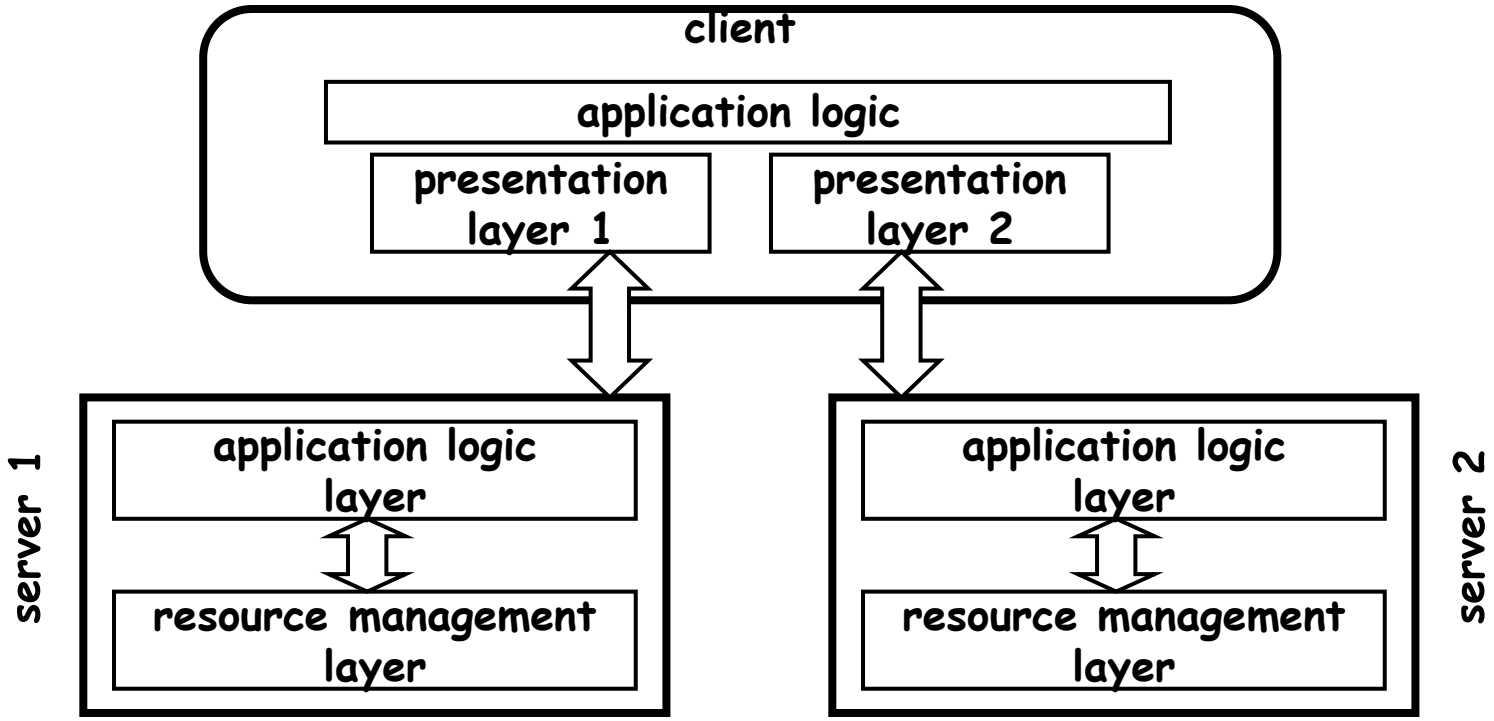
Architettura 2-tier – vantaggi

- Permette di realizzare un'architettura client/server
- Al tempo stesso, le architetture 2-tier garantiscono maggiore portabilità rispetto all'architettura 1-tier
- I client sono indipendenti gli uni dagli altri: si possono avere diversi livelli di presentazione sulla base delle esigenze di diversi client
- La capacità di calcolo del client consente di avere interfacce grafiche sofisticate ed allo stesso tempo risparmio di risorse sul server
- La possibilità di combinare sul server i livelli di logica dell'applicazione e di gestione delle risorse consente di mantenere una certa efficienza

Architettura 2-tier – svantaggi

- La contemporanea presenza sul server della logica dell'applicazione e della gestione delle risorse richiede server dalle prestazioni abbastanza elevate
- Supportano un limitato numero di client, a causa della necessità di mantenere informazioni sulla connessione e sull'autenticazione dei client
- Complessità dovuta alla pubblicazione ed al mantenimento di interfacce API, o a compatibilità fra diversi componenti
- I client si sono evoluti indipendentemente dai server ed hanno presentato nel tempo funzionalità sempre più avanzate. In particolare, hanno cercato di integrare più server, ma con l'architettura sbagliata !!

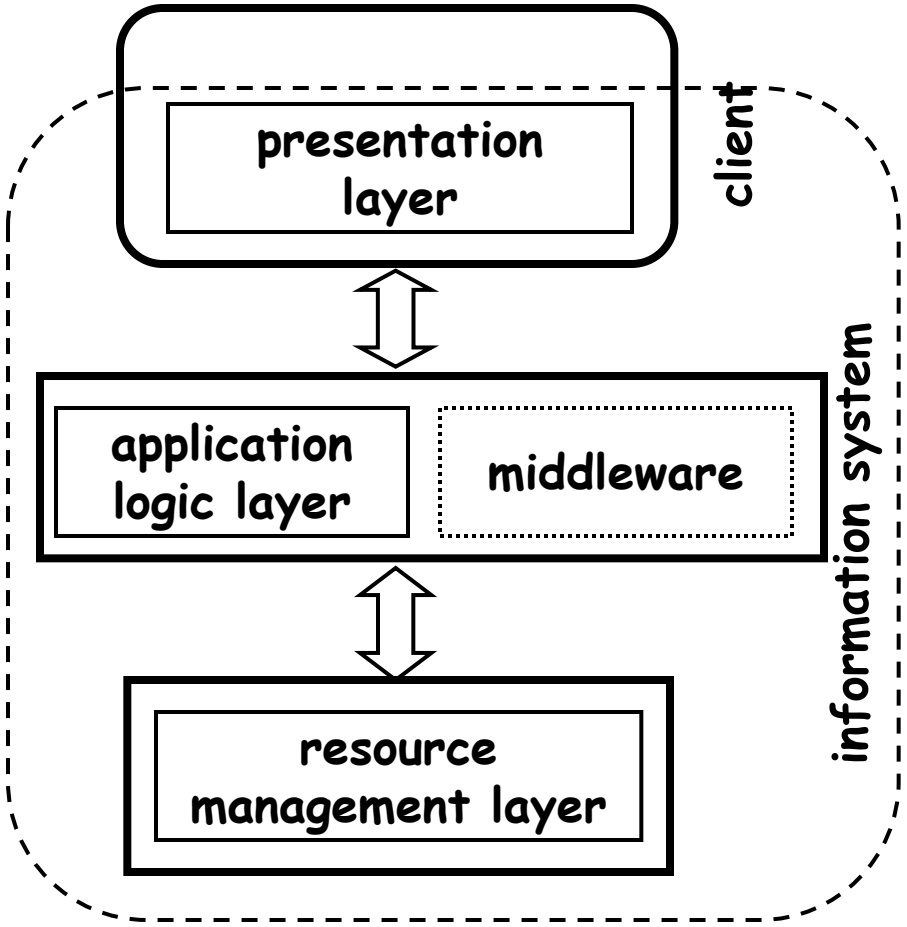
Architettura 2-tier ed integrazione



L'interazione di uno stesso client con più livelli di gestione di risorse ha fatto nascere l'esigenza di avere client via via più potenti, oltre che uno strato di logica applicativa per gestire l'integrazione delle risorse sul client stesso

→ Non scalabile !!

Architettura 3-tier



- I livelli concettuali sono completamente disaccoppiati
- Risponde all'esigenza di scalabilità
  - integrare più server
  - gestire più utenti
- Si avvale della presenza di stabili interfacce (API) sia tra il presentation layer e l'application logic layer che tra l'application logic layer e il resource management layer
- Si può avvalere della funzionalità offerte da un middleware
  - integrazione
  - replicazione

Il middleware

- Il middleware è l'insieme delle astrazioni di programmazione e delle infrastrutture che supportano lo sviluppo della logica dell'applicazione
- Come astrazione di programmazione:
  - Nasconde i dettagli dell'hardware, della rete e della distribuzione della computazione
  - Presenta sempre più potenti primitive che non cambiano concetti di base di RPC ma aumentano la flessibilità nel loro uso
  - La sua evoluzione è influenzata dalle tendenze nei linguaggi di programmazione (RPC e linguaggio C, CORBA e linguaggio C++, RMI e linguaggio Java, Web ed XML e Web service / RESTful service)
- Come infrastruttura:
  - Fornisce una piattaforma per lo sviluppo e l'esecuzione di applicazioni complesse
  - Presenta interfacce sempre più standardizzate
  - Si evolve verso un'architettura orientata ai servizi
  - L'obiettivo è l'integrazione delle piattaforme e la flessibilità nella configurazione

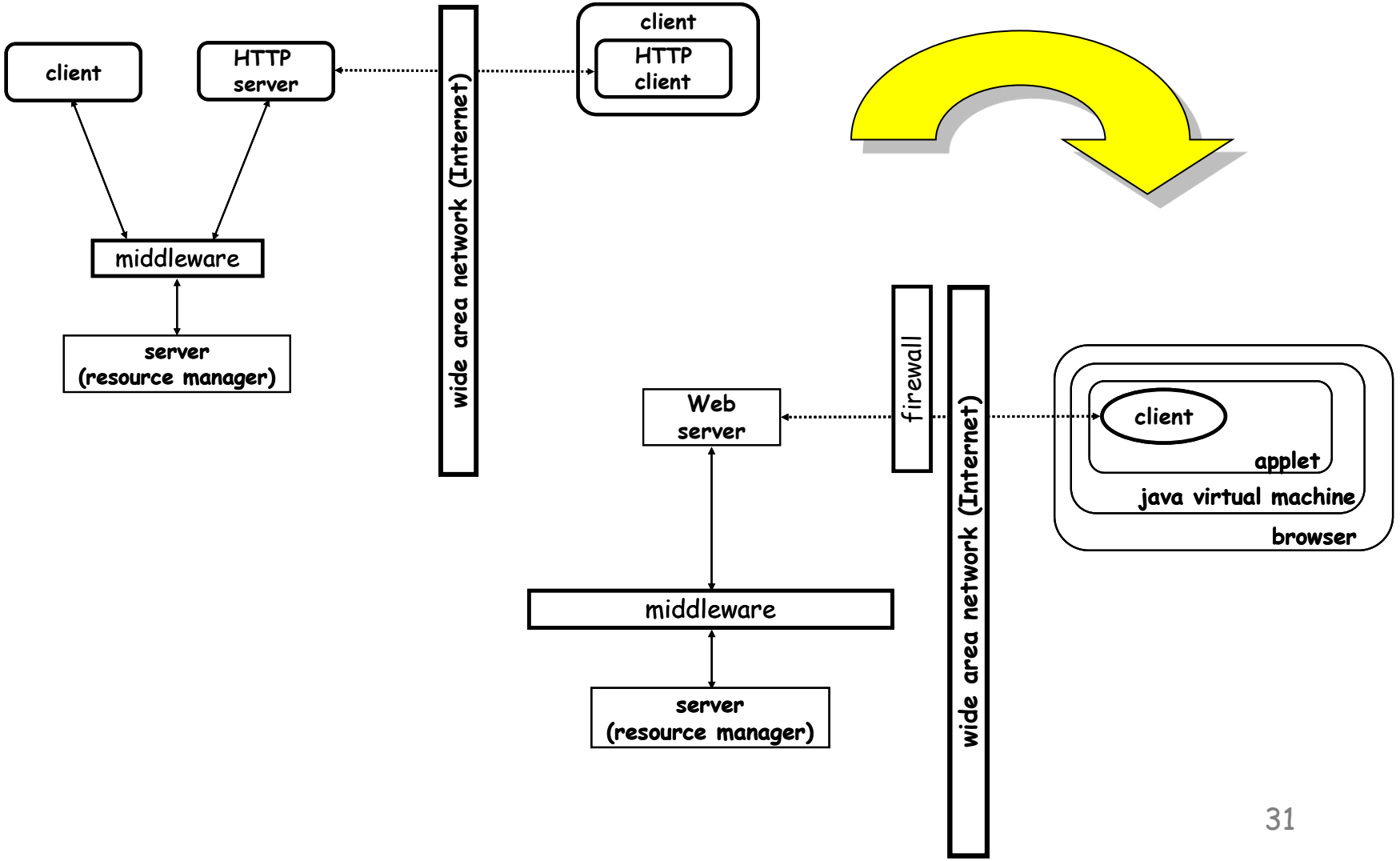
Architettura 3-tier - vantaggi

- Consente l'integrazione di sistemi differenti
- Tutta la logica dell'applicazione risiede nello strato intermedio, garantendo per l'intero sistema:
  - maggiore portabilità
  - manutenzione più semplice
  - aggiornamento di uno qualsiasi dei tier più semplice
  - maggiore flessibilità
  - maggiore scalabilità
- Il livello della logica dell'applicazione può essere distribuito su diversi server

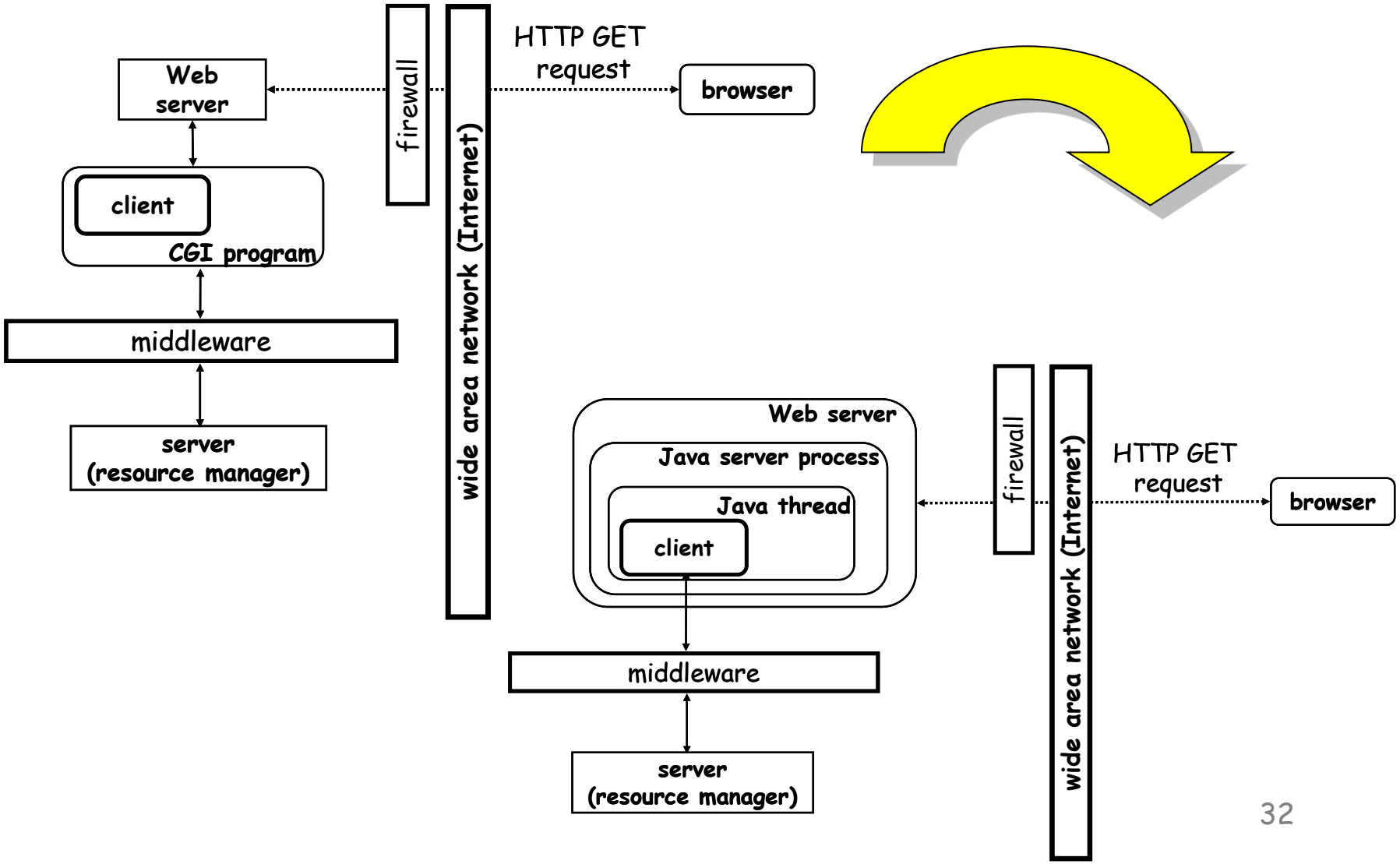
Architettura 3-tier – svantaggi

- Lo svantaggio principale è nel decadimento delle performance dovute ai problemi di comunicazione fra i diversi nodi del sistema
- Costo aggiuntivo per implementare le interfacce standard

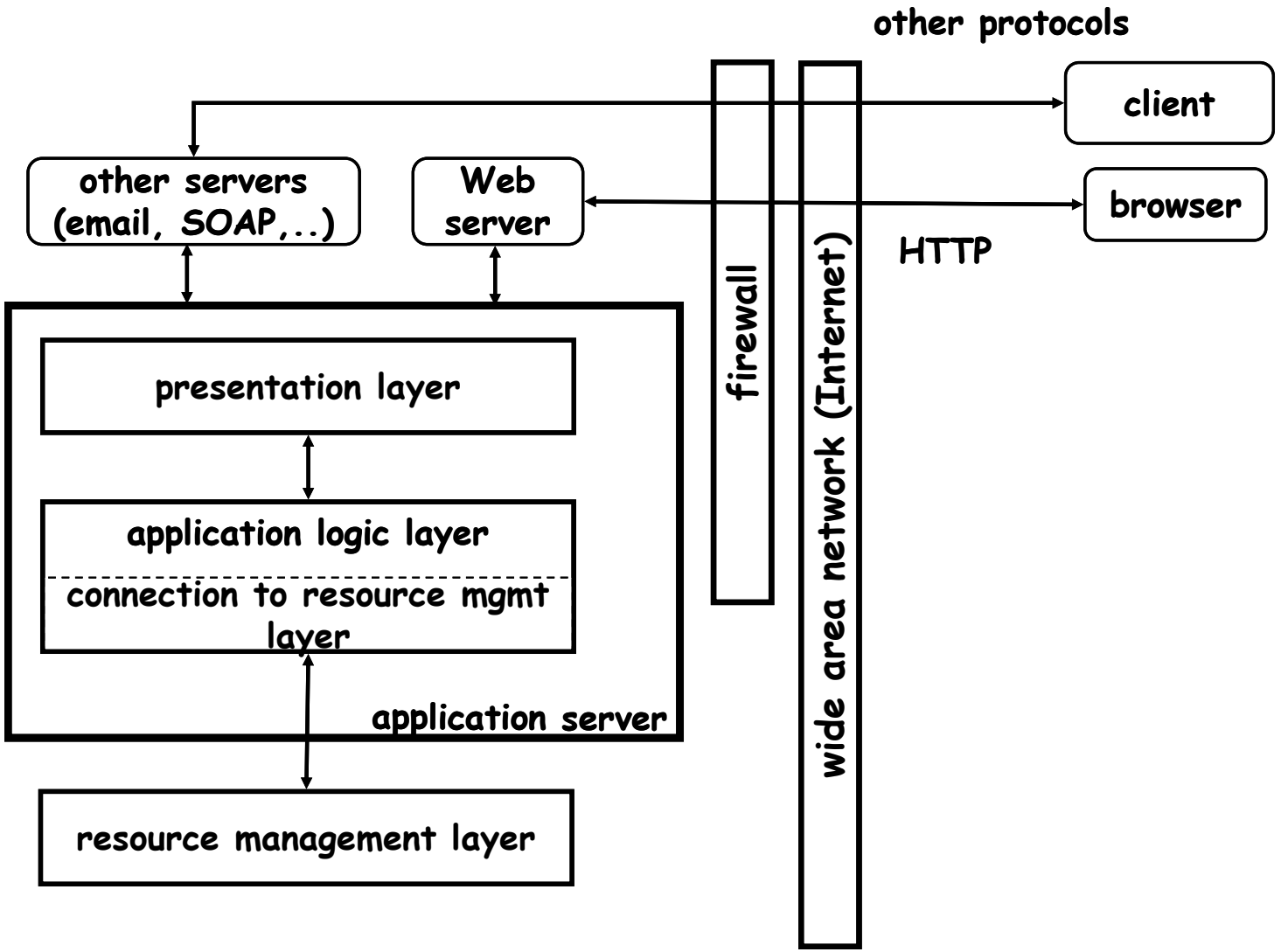
3-tier on the Web: Support for Remote Clients



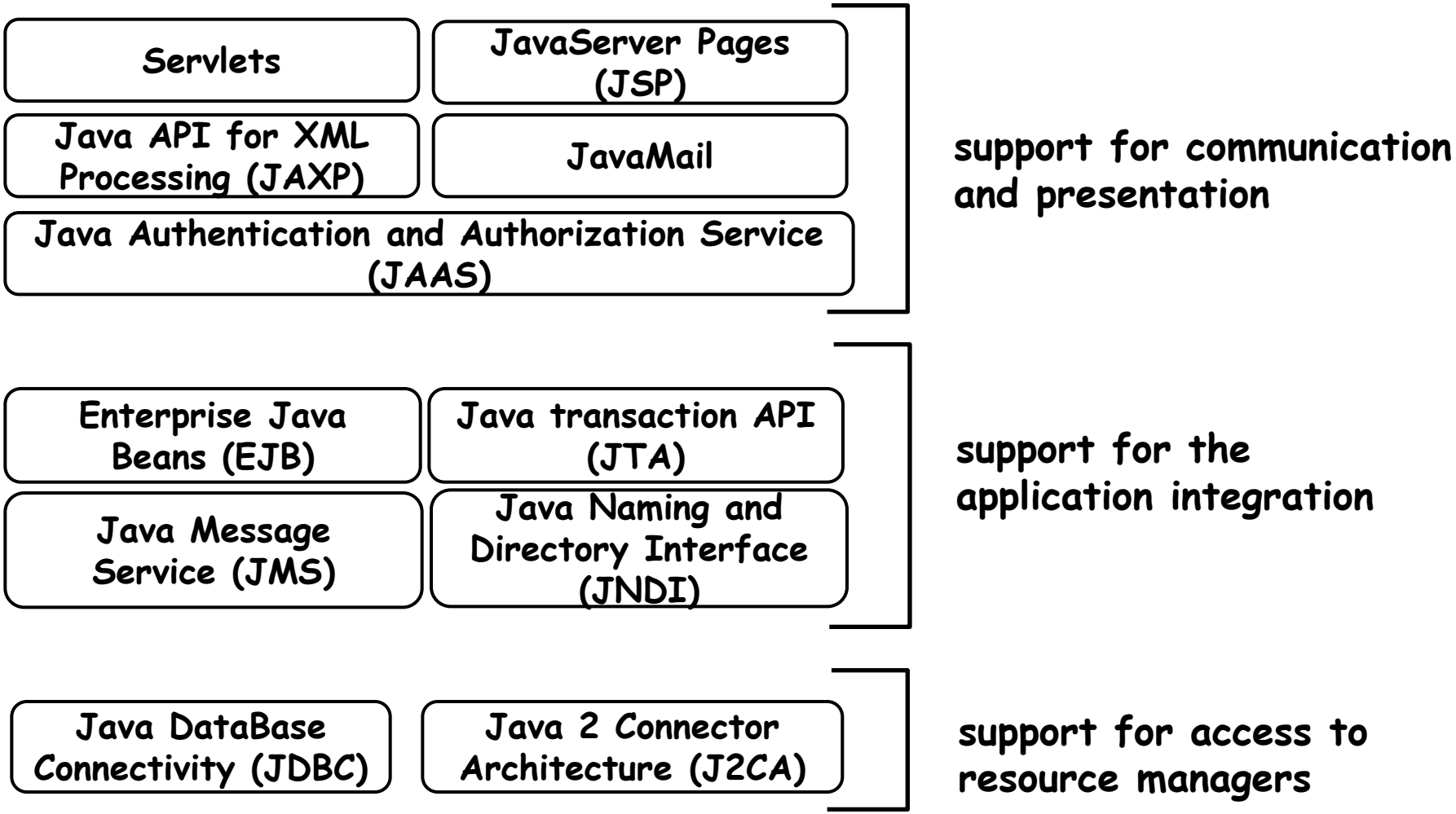
3-tier on the Web: Server Pages



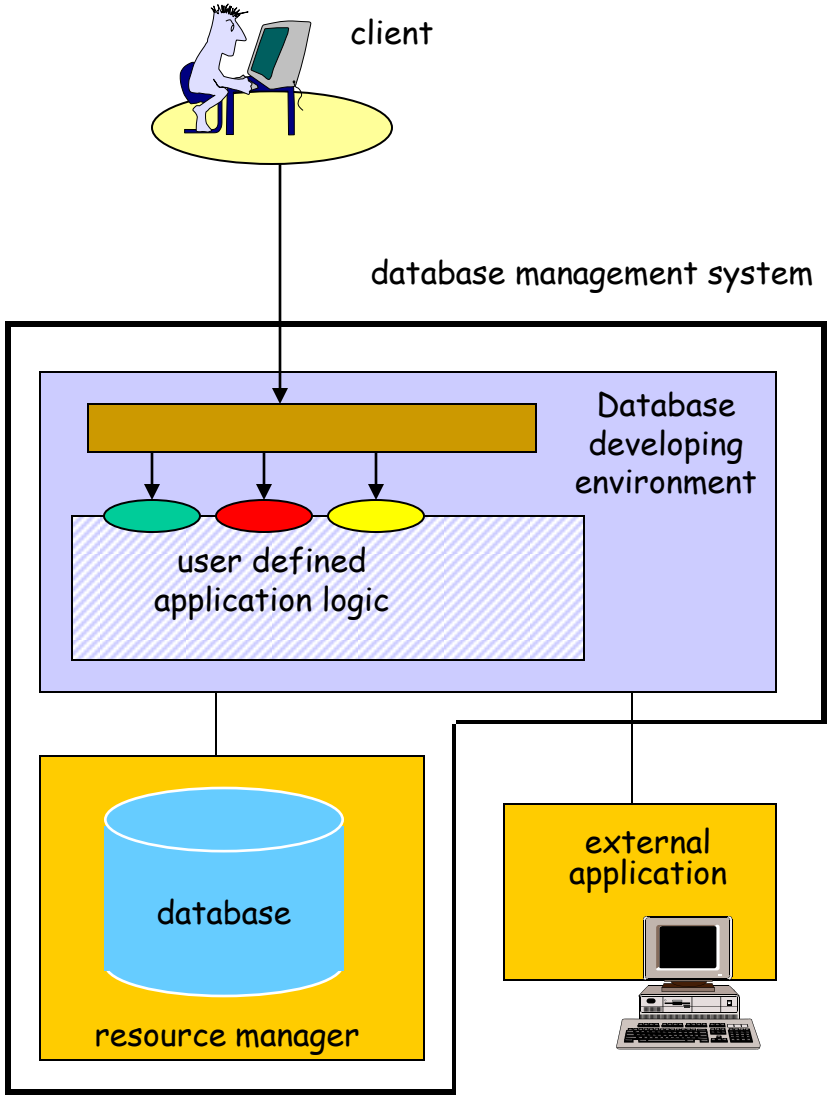
Putting All Together: Application Servers



An Example: J2EE



DBMS per la logica dell'applicazione



- DBMS sono usati tradizionalmente per gestire i dati
- Forniscono molti strumenti per realizzare parte della logica dell'applicazione
  - triggers
  - replicazione
  - stored procedures
  - queuing systems

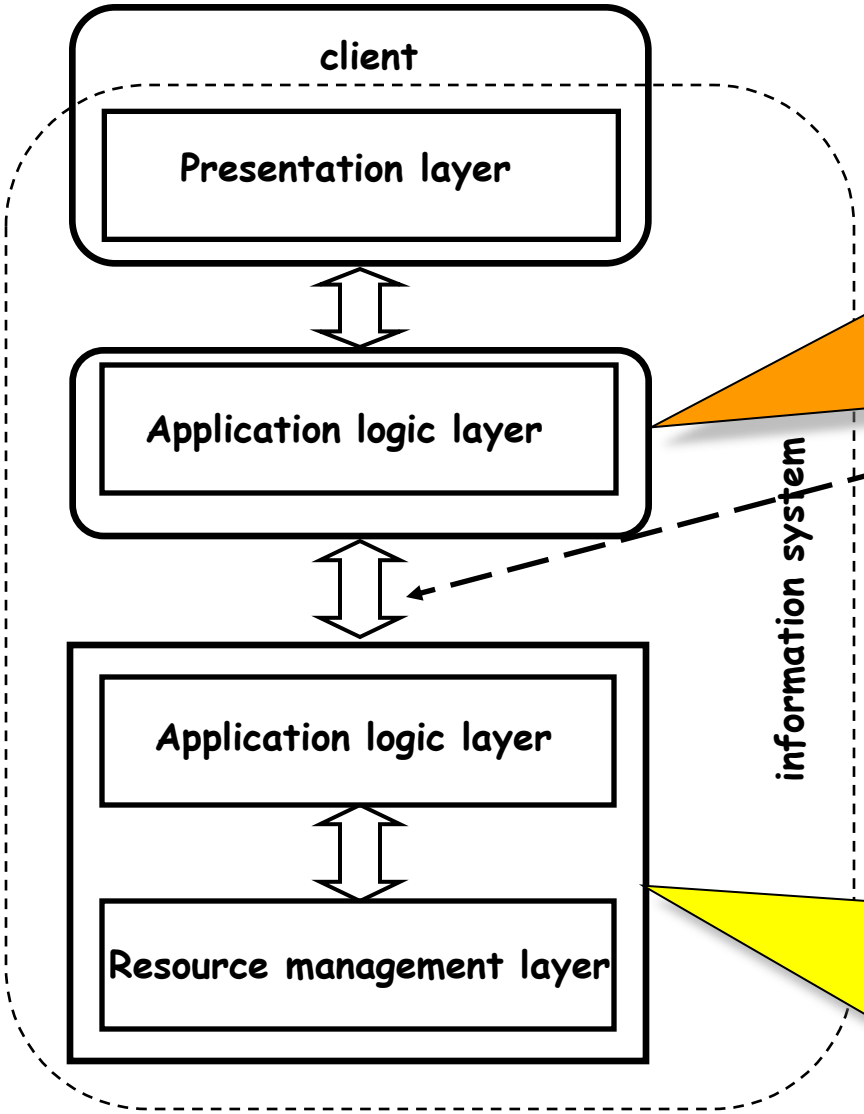
→ funzioni estremamente efficienti

→ problemi: scalabilità, modularità, manutenzione e aggiornamento

In questo corso (1)

- Vogliamo realizzare sistemi N-tier, in cui i livelli concettuali di presentazione, logica dell'applicazione e gestione delle risorse siano disaccoppiati
- Ci rifacciamo alle architetture client/server (anche quando il nostro sistema risiede su una singola macchina)
- Vogliamo che la logica dell'applicazione non risieda completamente nel DBMS
- Ci concentriamo sul livello della logica dell'applicazione, sul data layer e sulla interazione fra i due (JDBC e PHP)
- Il livello di presentazione deve essere separato, e possibilmente orientato al Web
- Uso di un middleware se necessario

In questo corso (2)



Insegnamo a realizzare un'applicazione web che si connette ad un DBMS attraverso diverse modalità (Java/JDBC, PHP)

A livello di deployment sceglieremo un DBMS relazionale con supporto per stored procedure e trigger (e quindi possibilità di gestire parte della logica applicativa lato server)