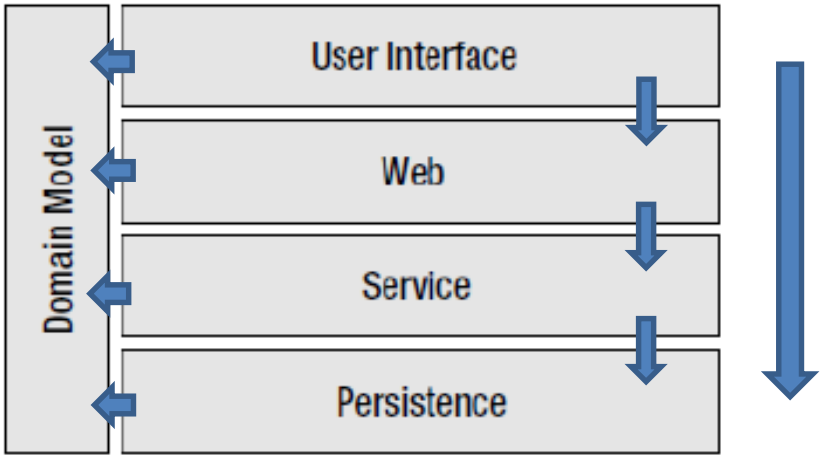
Modern Web app architecture

- Модели архитектур
- Архитектура Веб-приложений

Application Layering

- Presentation Layer
  - User interface
  - Web
- Service Layer
- Domain Layer
  - Domain object model
- Infrastructure Layer
  - Repository
  - Persistence



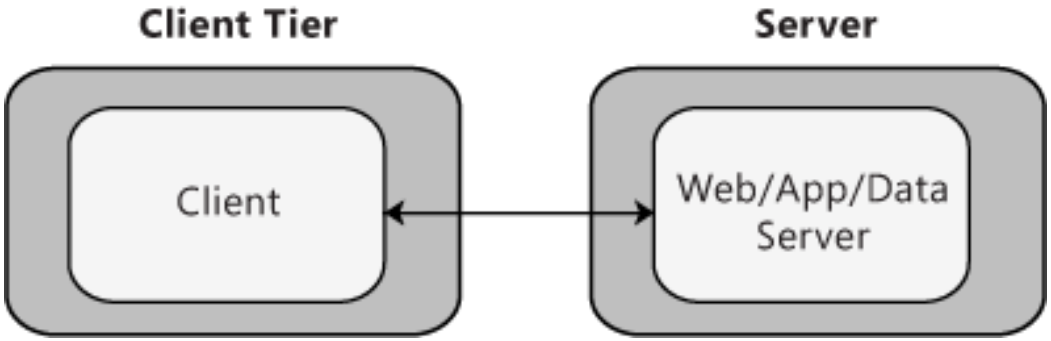
Модели архитектур приложений

- Клиентские приложения
- Клиент-серверная архитектура (тонкий и толстый клиент)
- Трех и многоуровневая архитектура
- Веб-приложения

Stand-alone Deployment

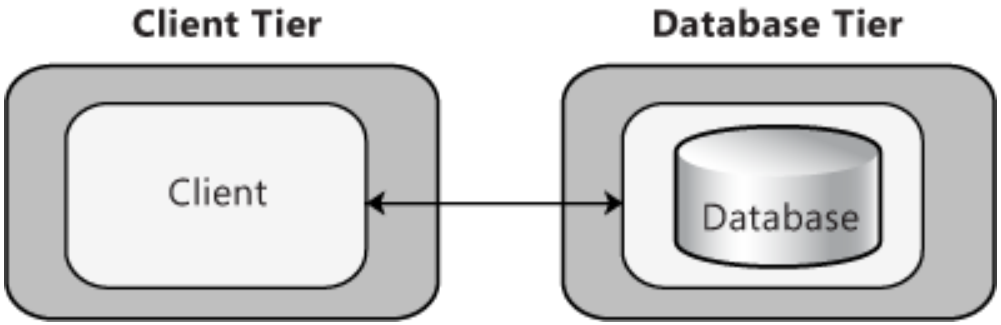
Client–server model

- The client–server model of computing is a distributed computing structure that partitions tasks or workloads between the providers of a resource or service, called servers, and service requesters, called clients
- The *client–server* characteristic describes the relationship of cooperating programs in an application. The server component provides a function or service to one or many clients, which initiate requests for such services.

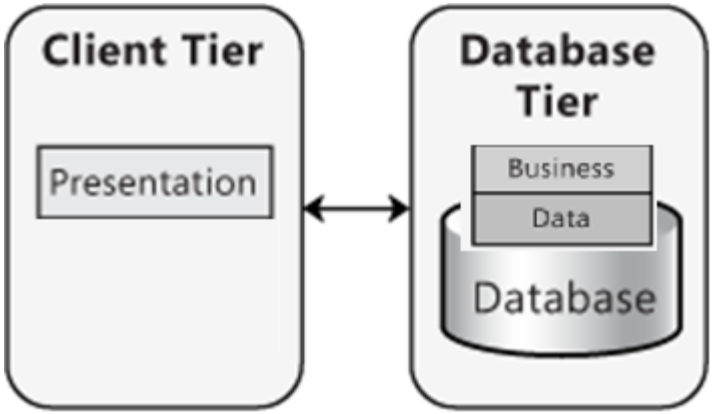


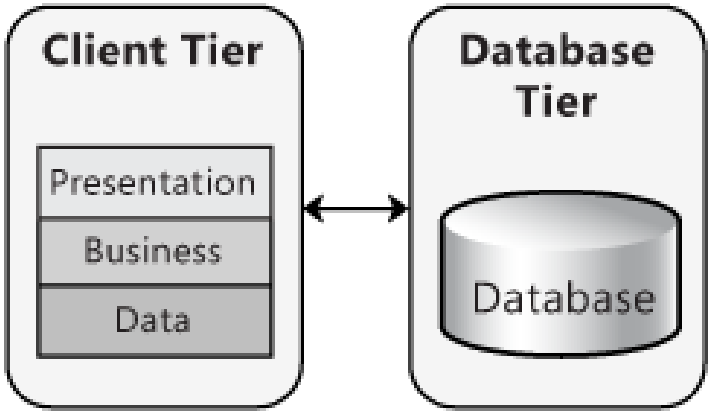
Client/Server

- Segregates the system into two applications, where the client makes requests to the server.
- In many cases, the server is a database with application logic represented as stored procedures.



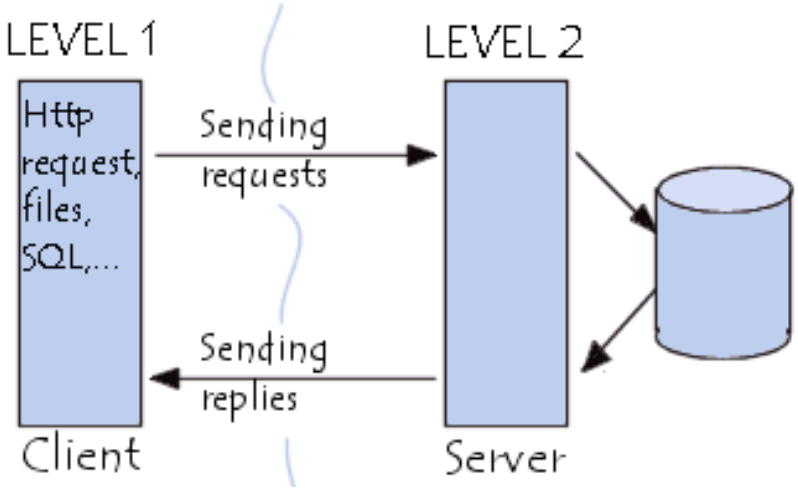
Thin Client vs Thick Client Architecture





Client and server communication

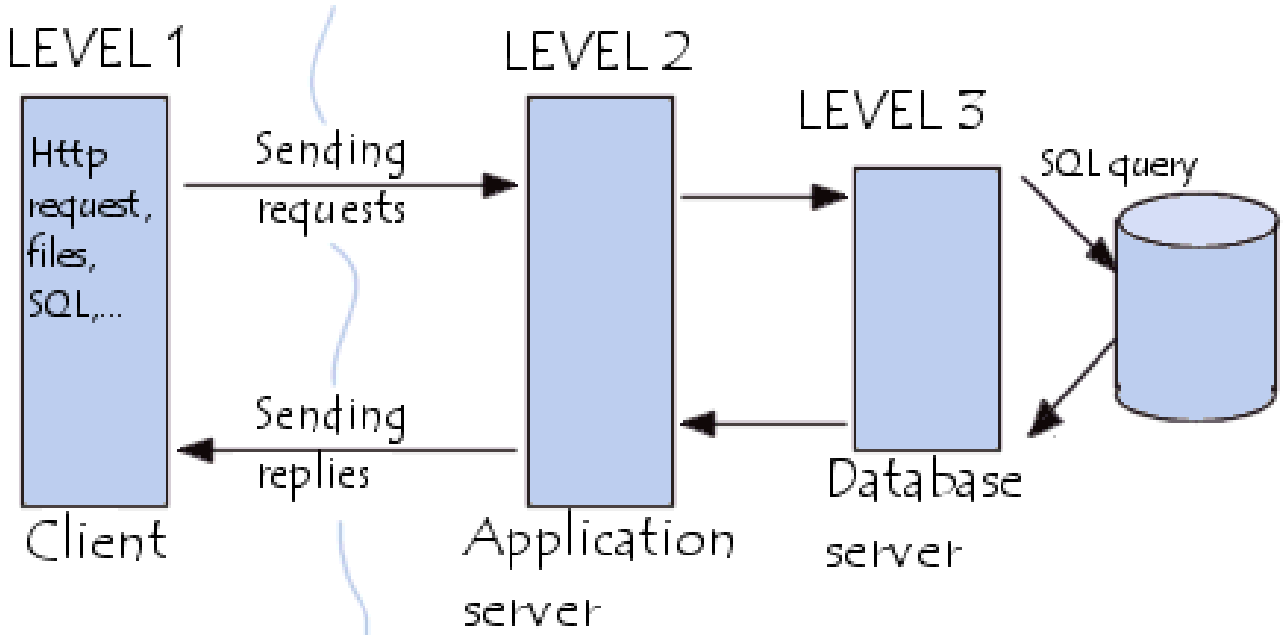
- Clients and servers exchange messages in a request-response messaging pattern: The client sends a request, and the server returns a response



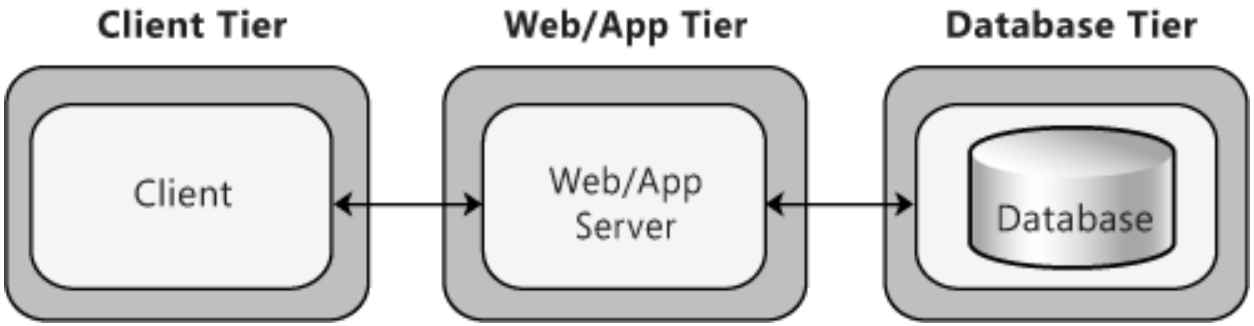
Client and server communication

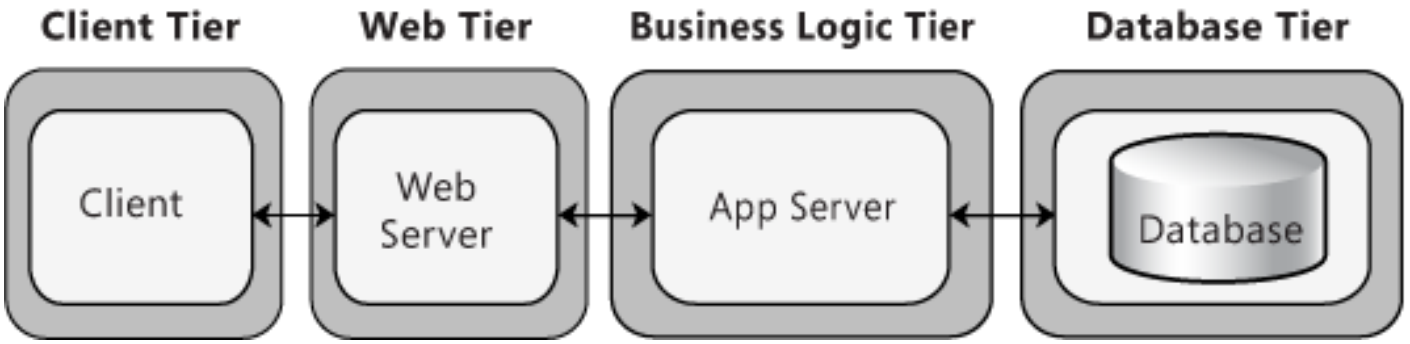
- Clients and servers exchange messages in a request-response messaging pattern: The client sends a request, and the server returns a response
  - Synchronous communication
  - Asynchronous communication
- The language and rules of communication are defined in a communications protocol.

3-Tier Architecture

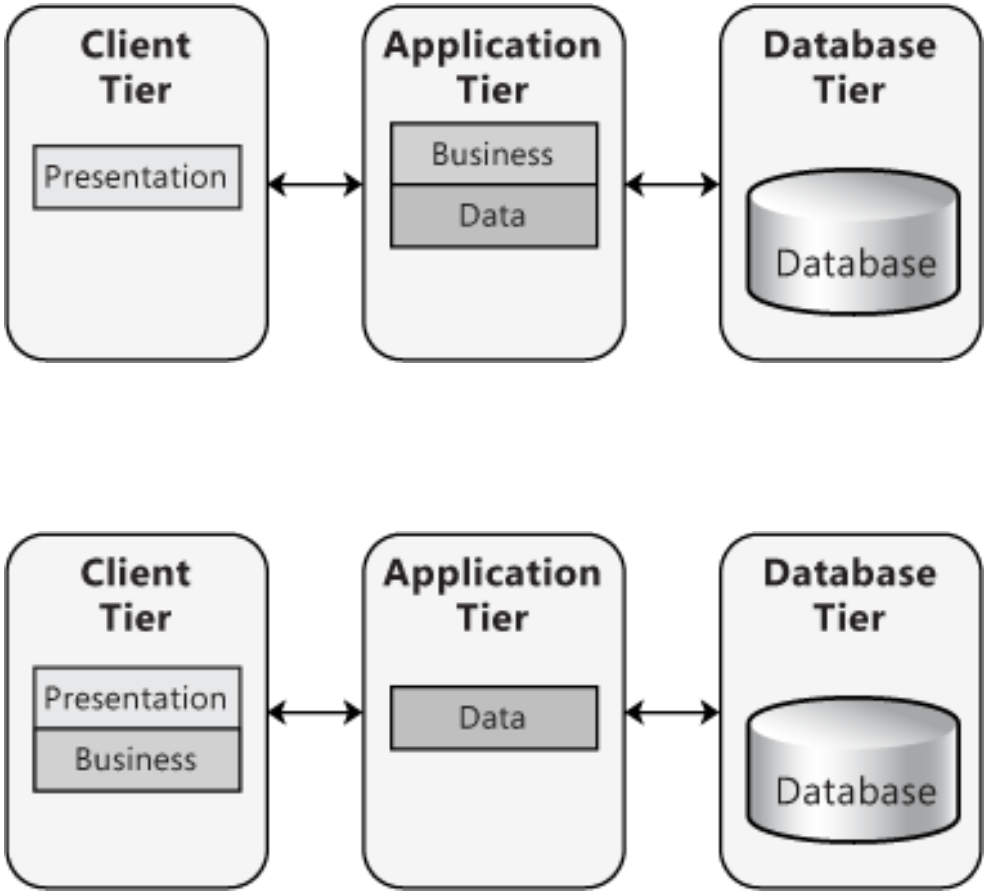


N-Tier Architecture

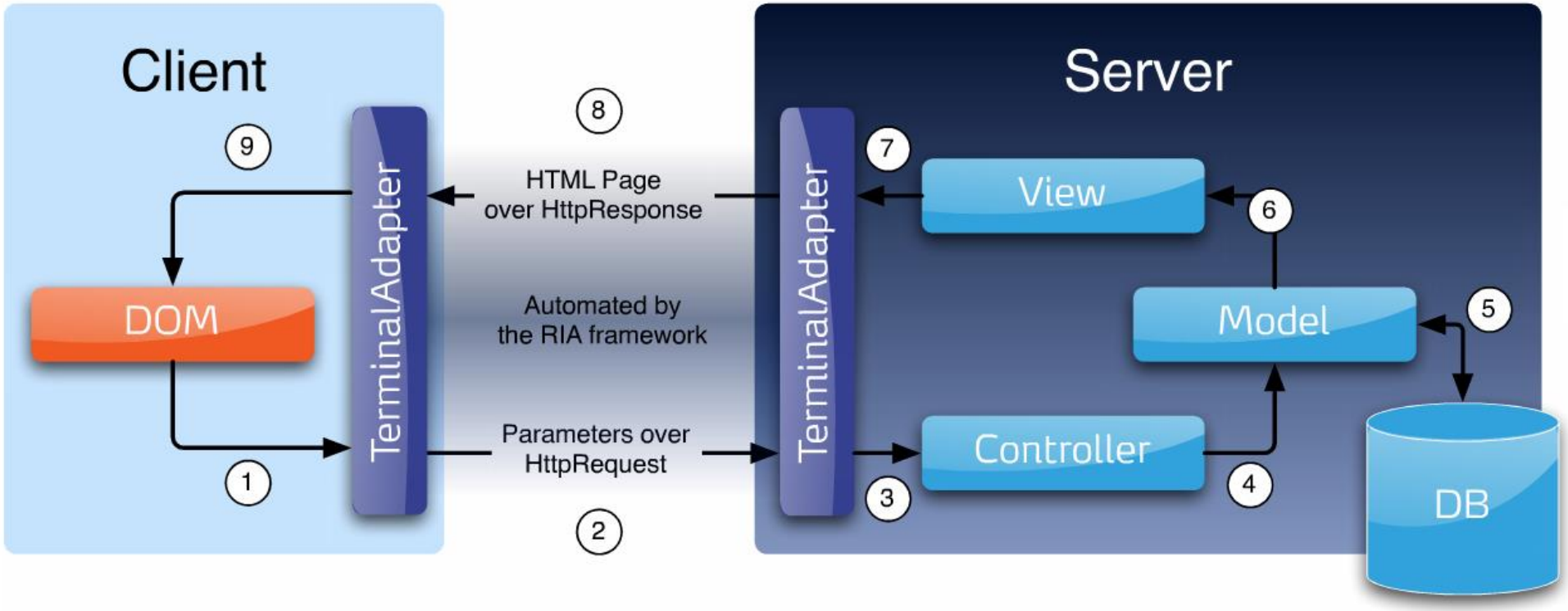




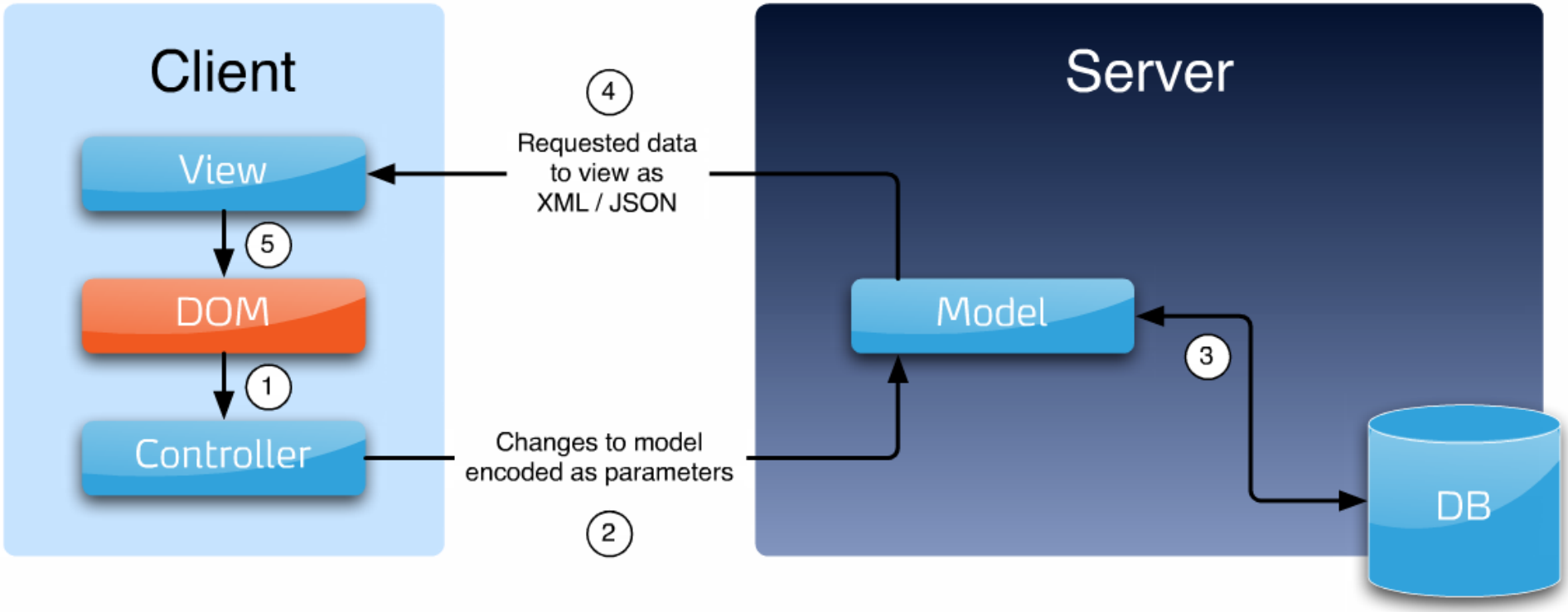
Thin vs Rich client

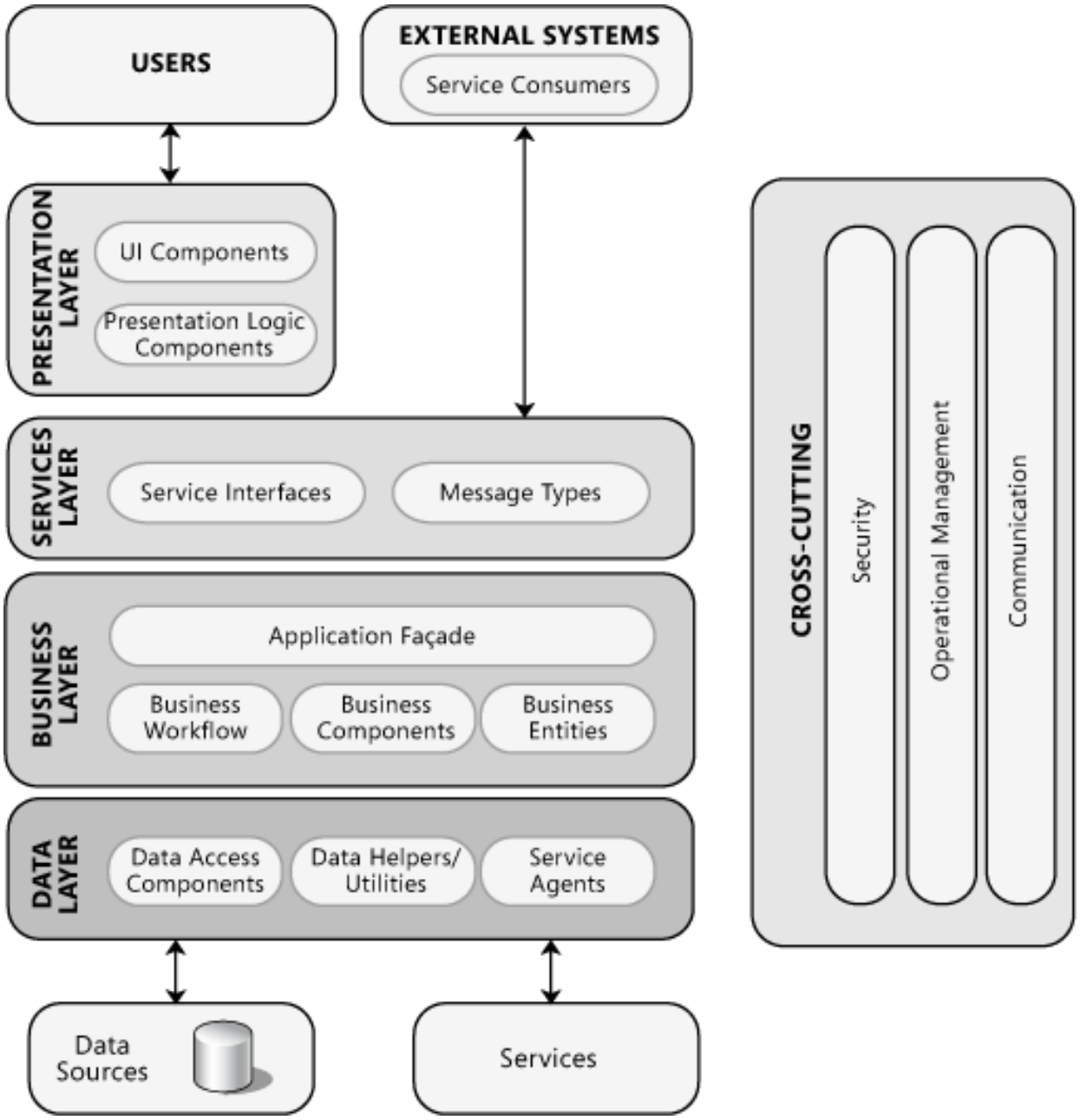


Server-side


Client
Server
8
9
7
HTML Page
over HttpResponse
View
6
TerminalAdapter
TerminalAdapter
DOM
Automated by
the RIA framework
Model
5
Parameters over
HttpRequest
Controller
DB
1
3
4
2

Client-side


Client
View
5
DOM
1
Controller
4
Requested data
to view as
XML / JSON
Changes to model
encoded as parameters
2
Server
Model
3
DB

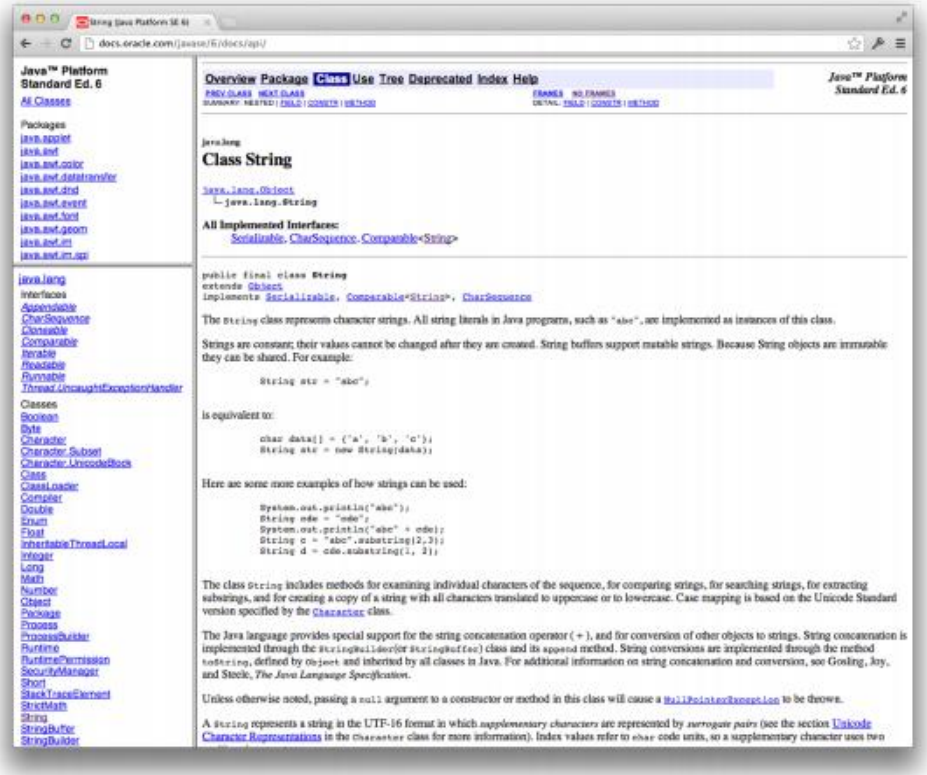

USERS
EXTERNAL SYSTEMS
Service Consumers
PRESENTATION LAYER
UI Components
Presentation Logic Components
SERVICES LAYER
Service Interfaces
Message Types
BUSINESS LAYER
Application Façade
Business Workflow
Business Components
Business Entities
DATA LAYER
Data Access Components
Data Helpers/ Utilities
Service Agents
Data Sources
Services
CROSS-CUTTING
Security
Operational Management
Communication

Архитектура современных Веб-приложений

История развития, от и до …

In the beginning...

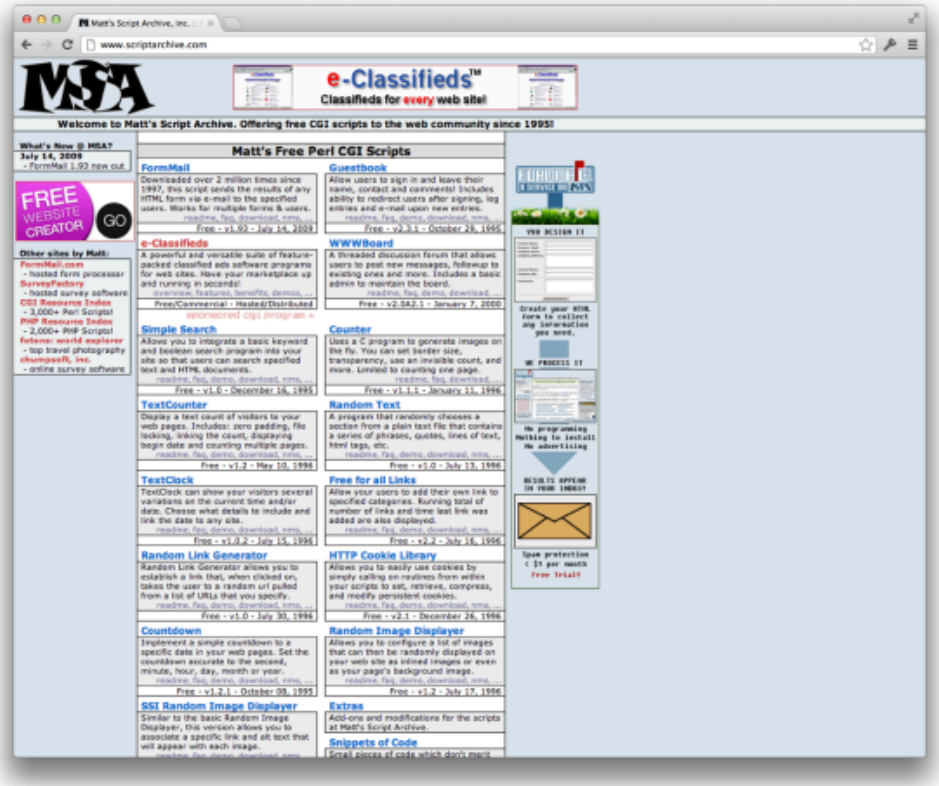
- Sites were static HTML



- Pros:
  - low computational overhead
  - highly cacheable
  - highly indexable
- Cons:
  - hard (easy?) to update
  - no personalization
  - usually poor UI

Let there be CGI

- Introduced dynamic generated pages



- Pros:
  - dynamic!
  - selectively cacheable
  - highly indexable
  - personalizable
- Cons:
  - “high” computational overhead
  - hard to create
  - usually poor UI

~~LiveScript~~ JavaScript

- Dynamic pages
- Lightweight complement to applets
- Mostly used for simple scripting
  - basic form validation
  - popup ads
  - comet cursor trails

- Pros:
  - enhanced usability, *maybe*
  - reduced trips to the server
- Cons:
  - abuses annoyed users
  - business logic often implemented twice: client and server

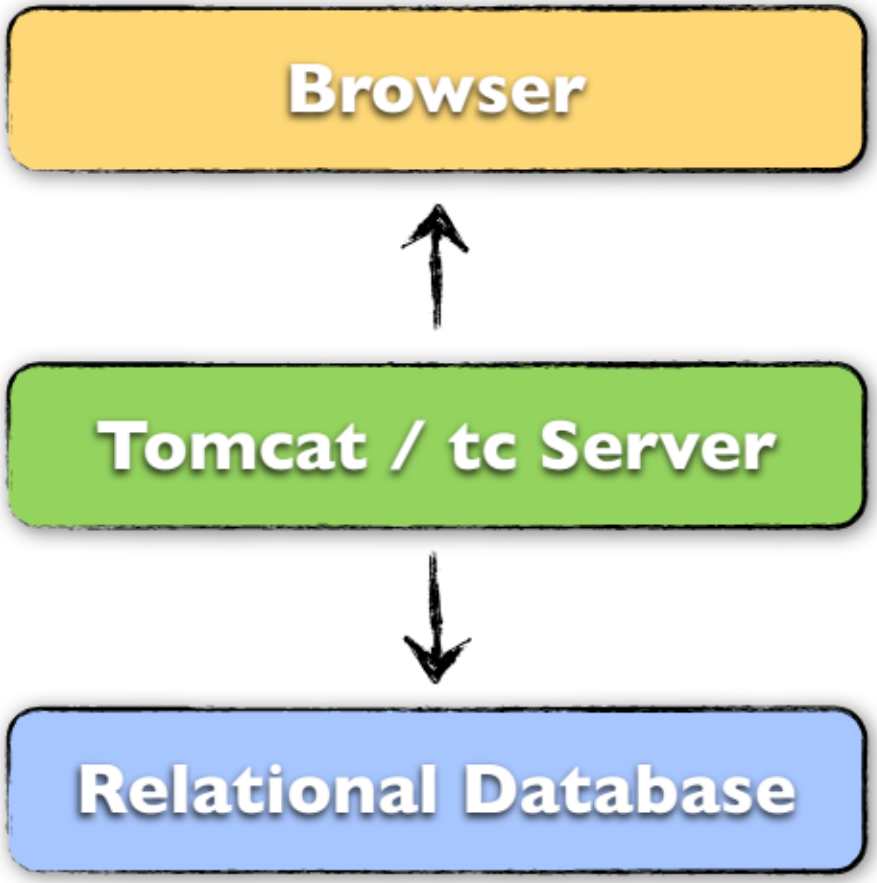
AJAX - Web 2.0

- Google Maps sparked Web 2.0
- GMail
  - required JavaScript

- Pros:
  - killer UI
  - more responsive apps

- Cons:
  - difficult to cache
  - impossible to index
  - required JavaScript

Typical Runtime Structures

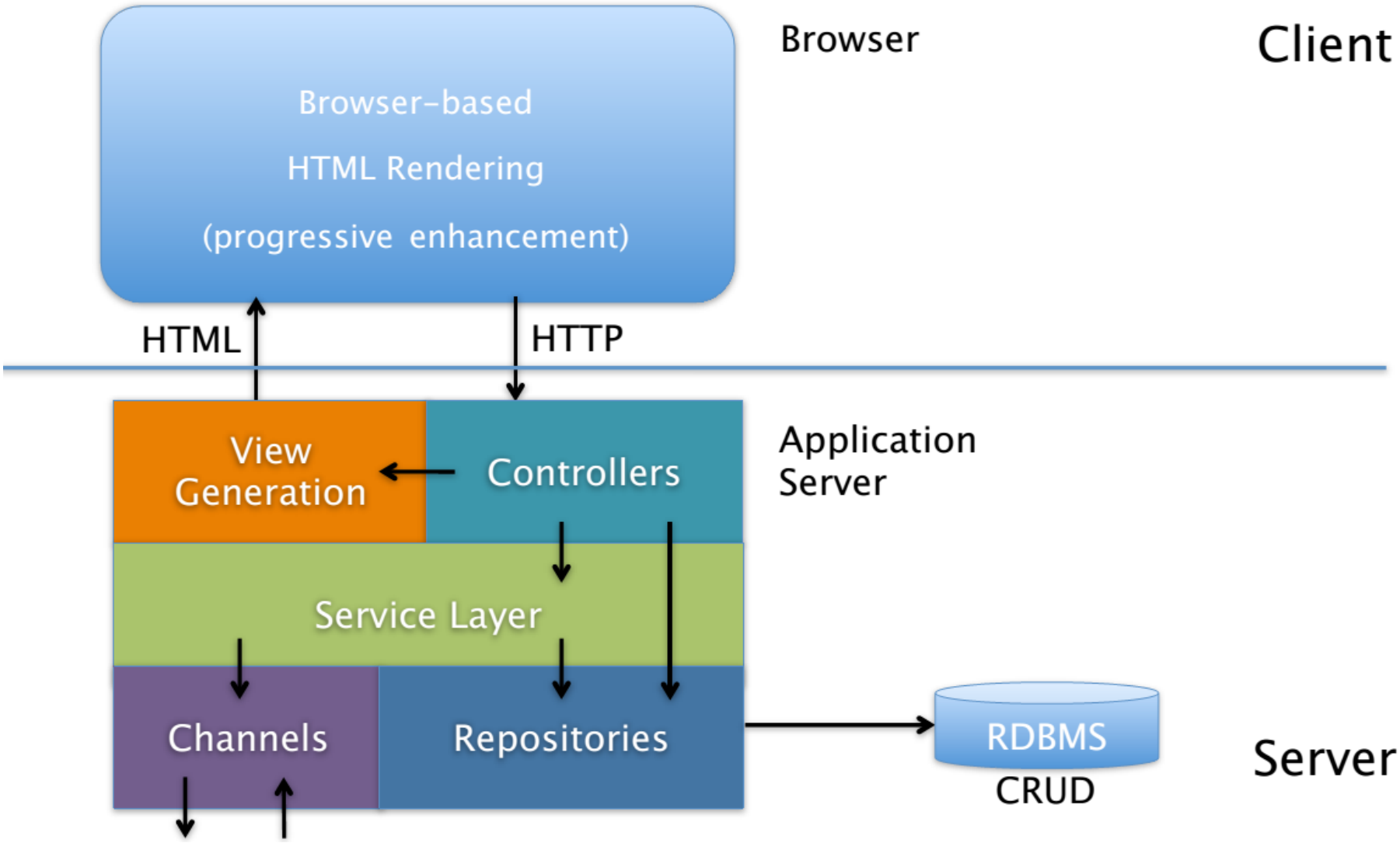


render HTML

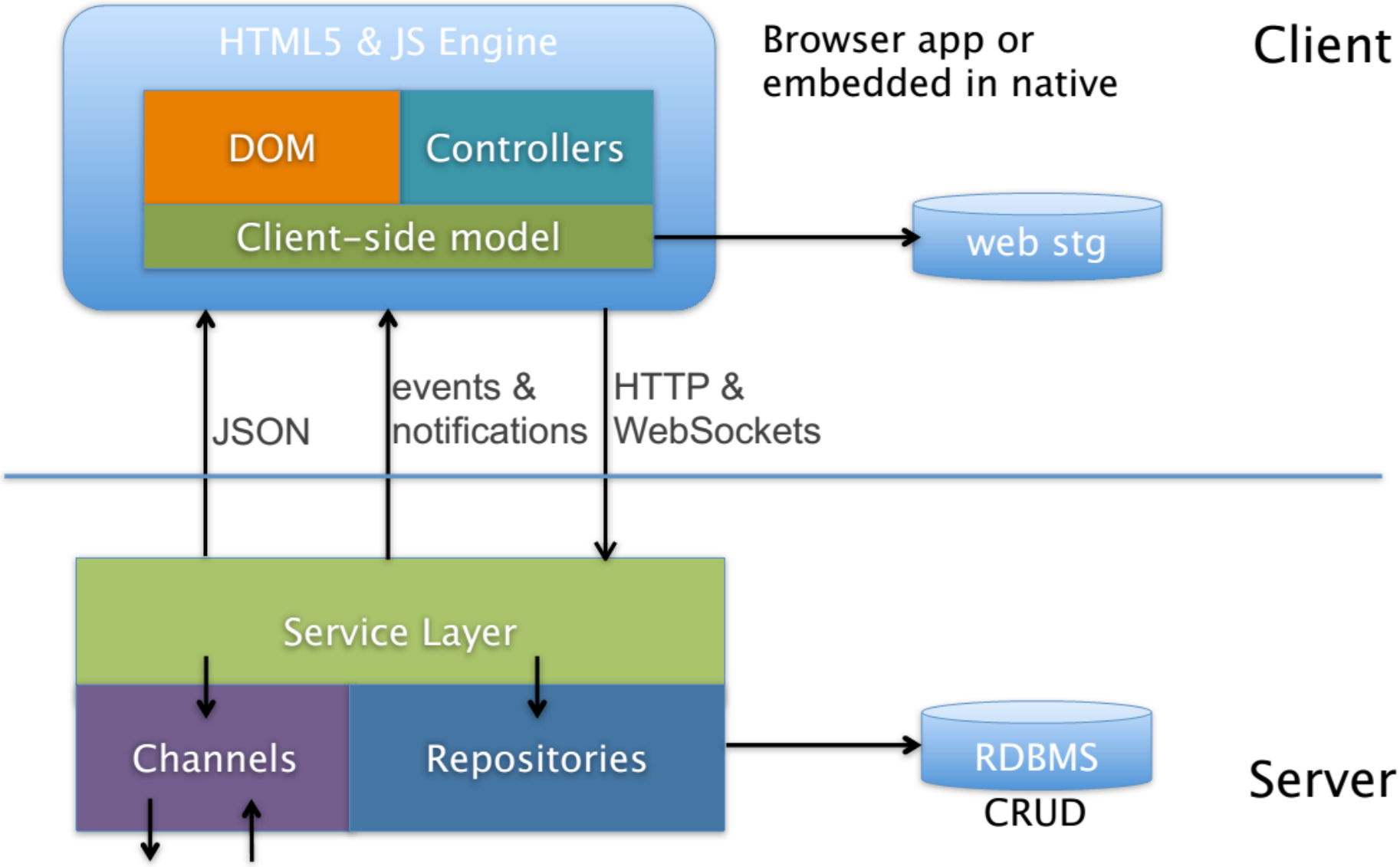
business logic &
page rendering

contains data

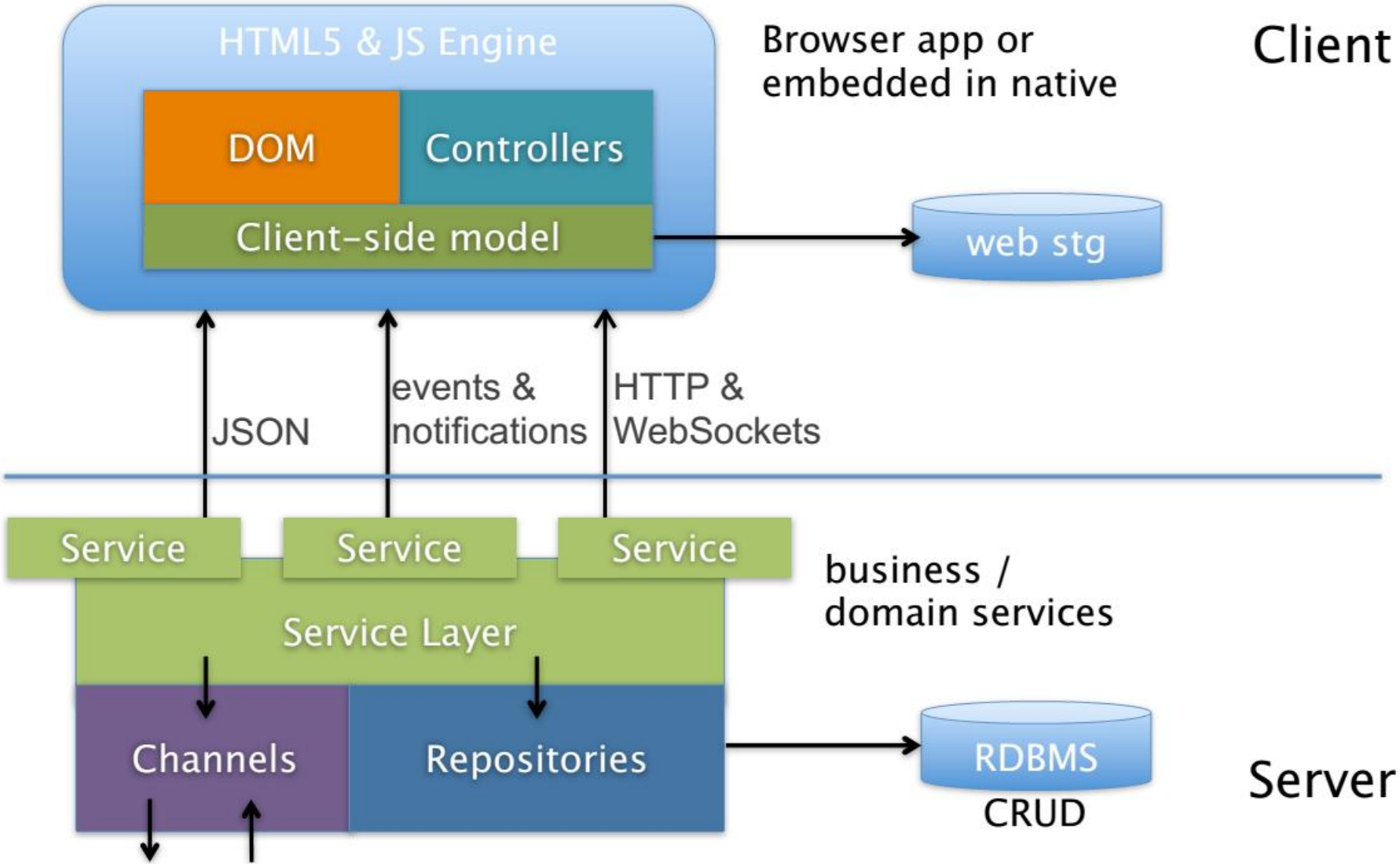
From server-side app to smart clients and services



From server-side app to smart clients and services



From server-side app to smart clients and services

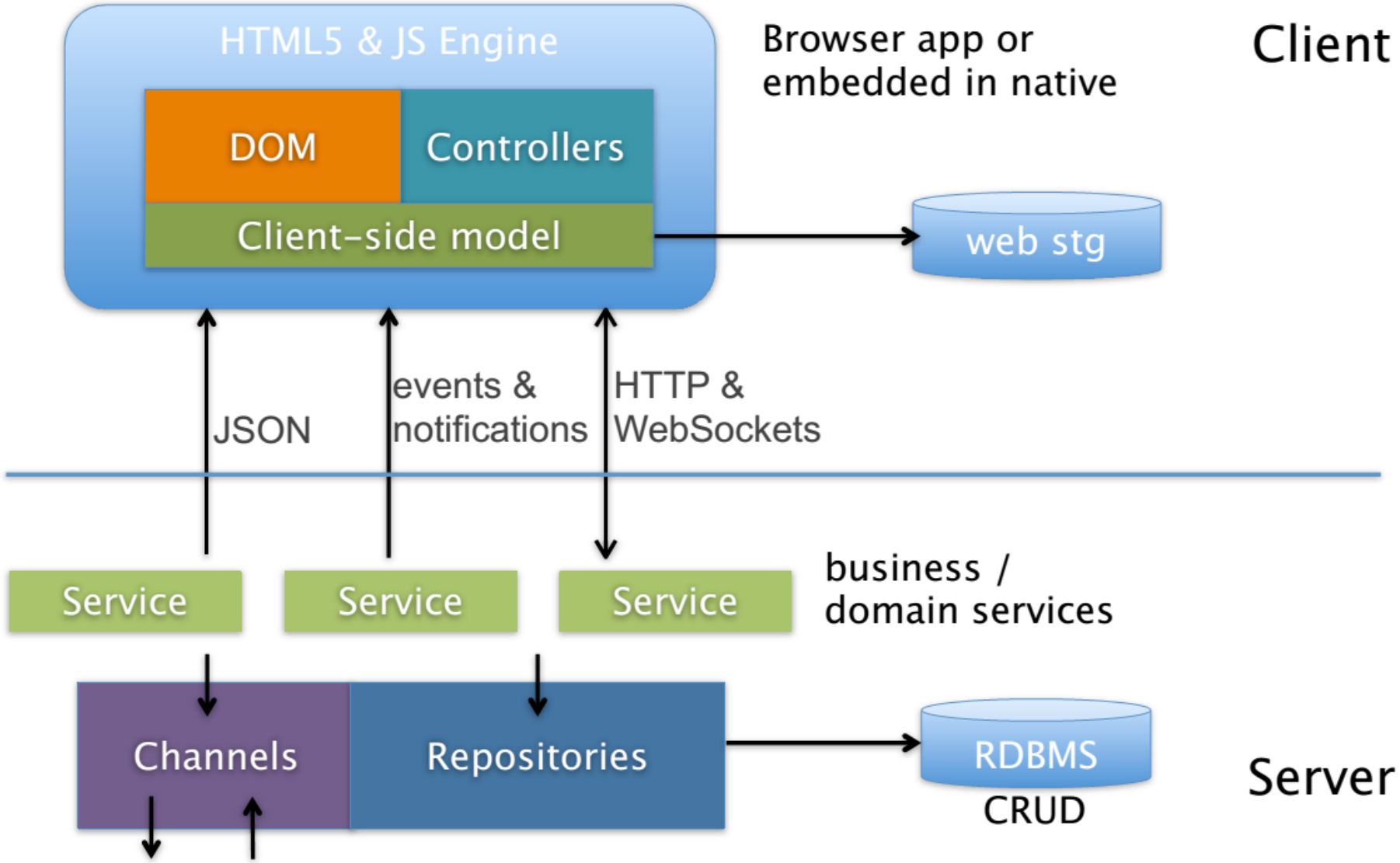


Client Side Applications

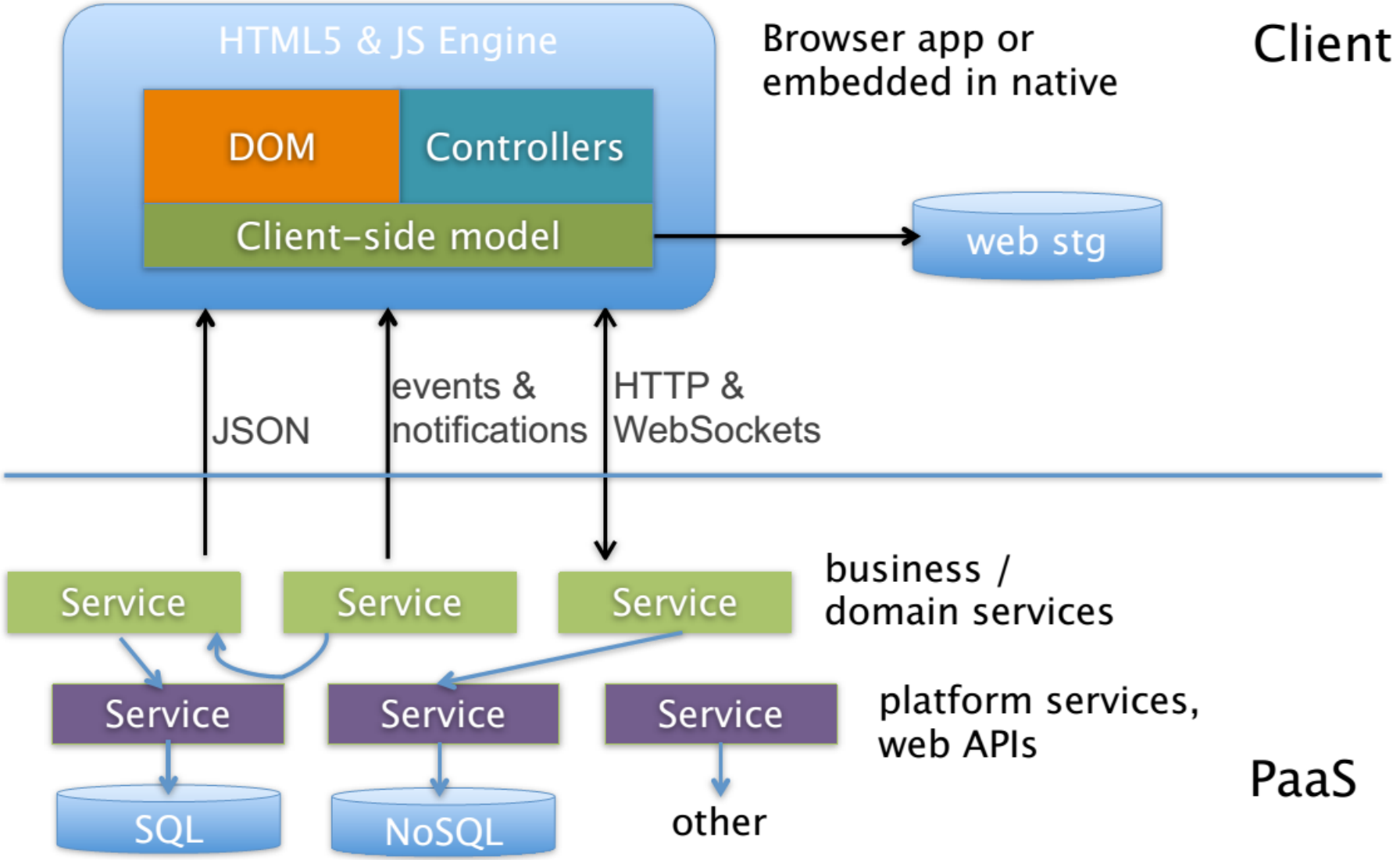
- Business logic lives on the client
- Resources and permanent state stored on the server
- Application and session state stored on client

- Pros:
  - reduce server workloads
  - application is highly cacheable
  - extremely rich UI
- Cons:
  - content not indexable
  - requires JavaScript
  - often requires a ‘modern’ browser

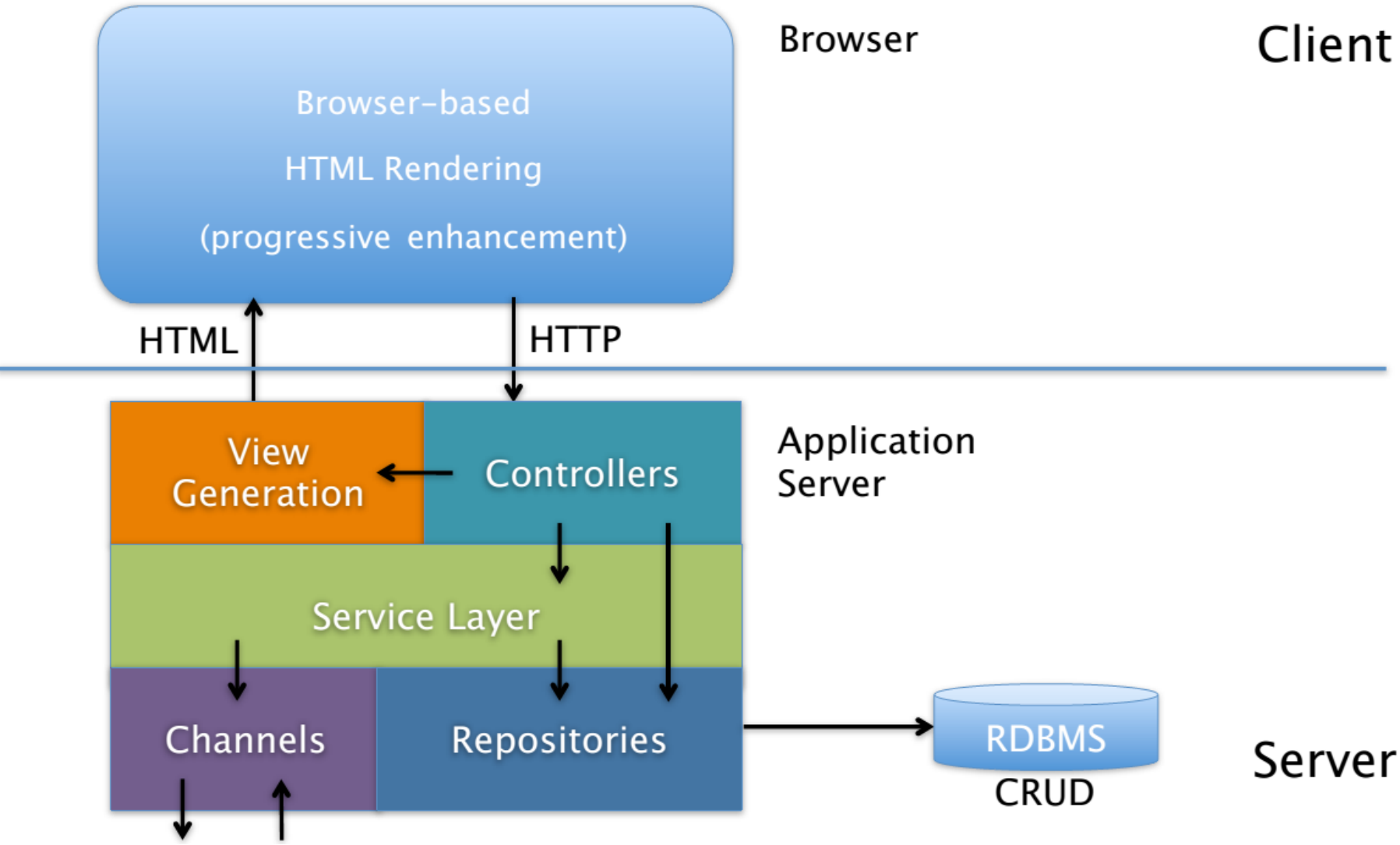
From server-side app to smart clients and services



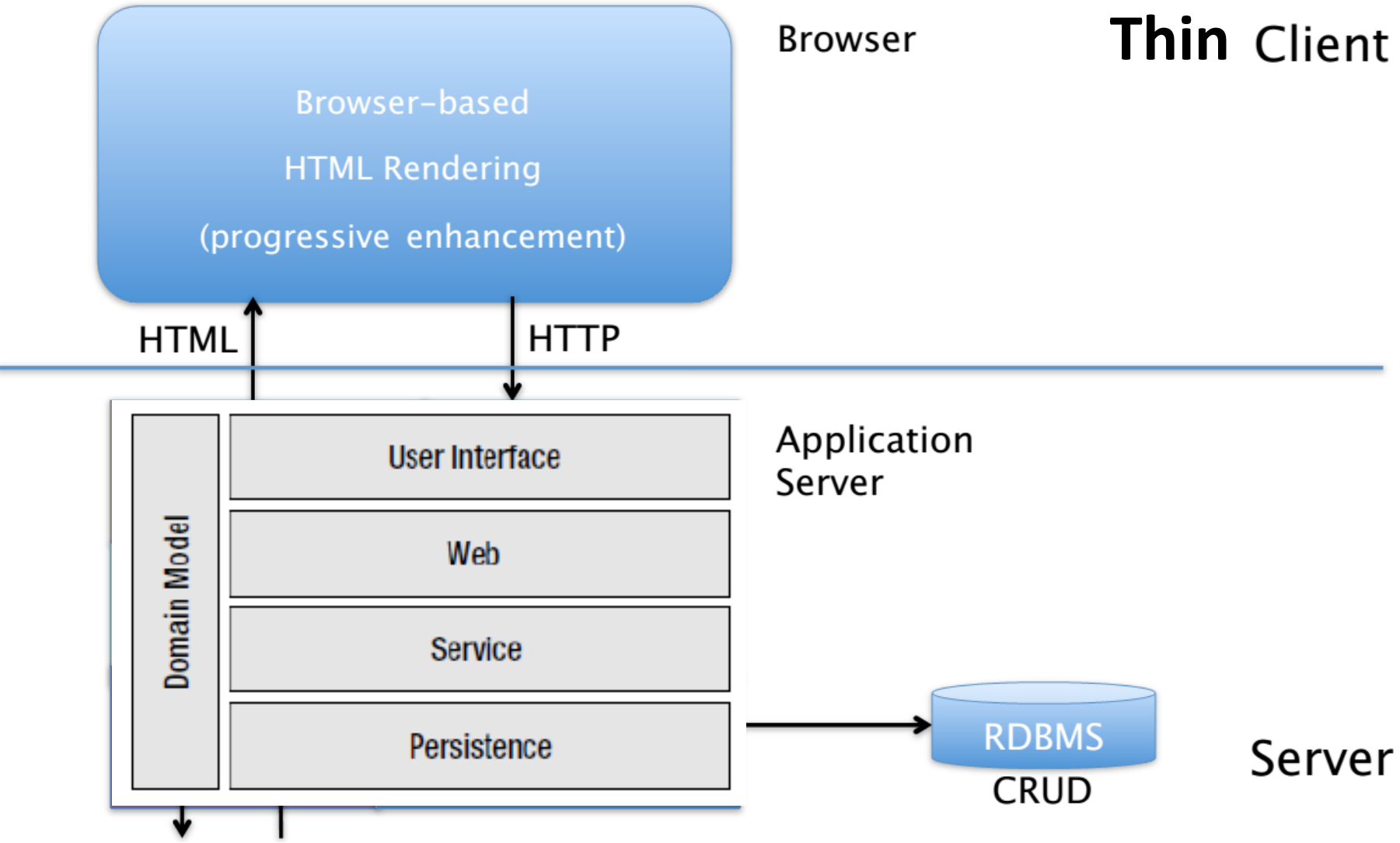
From server-side app to smart clients and services



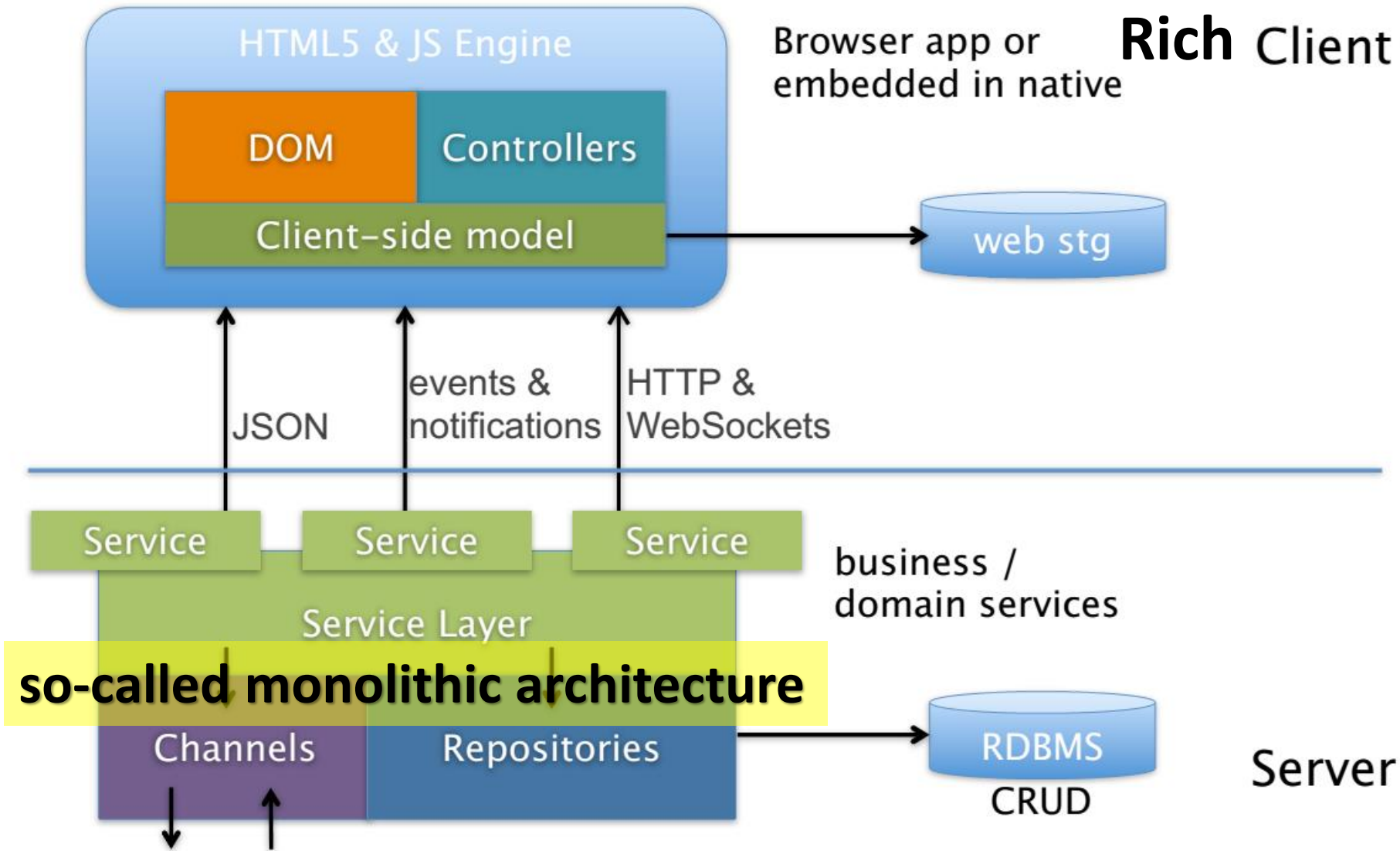
From server-side app to smart clients and services



From server-side app to smart clients and services



From server-side app to smart clients and services



Hexagonal architecture