CHAPTER 1

APPLICATION MODEL

Chapter 1: Application Model

CONTENT

1. Jakarta EE
   - ❑ Architecture Overview
   - ❑ Container Overview
   - ❑ APIs Overview
2. Web Application Architecture
3. Web Communication
4. HTTP Request/Response

Jakarta EE – History

- Jakarta EE: Java Platform Enterprise Edition (Java EE)
- J2EE: Java 2 Platform Enterprise Edition

Jakarta EE – History https://jakarta.ee/release/

Platform version	Released	Specification	Minimun Java SE Support	Important Changes
Jakarta EE 11	2024-06-26	11	Java SE 17	https://jakarta.ee/news/jakarta-ee-11-released/
Jakarta EE 10	2022-09-12	10	Java SE 17 Java SE 11	https://jakarta.ee/release/10/
Jakarta EE 9.1	2021-05-25	9.1	Java SE 11 Java SE 8	JDK 11 support https://jakarta.ee/release/9/
Jakarta EE 9	2020-12-08	9	Java SE 8	API namespace move from **javax** to **jakarta**
Jakarta EE 8	2019-09-10	8	Java SE 8	Full compatibility with Java EE 8
Java EE 8	2017-08-31	JSR 366	Java SE 8	HTTP/2 and CDI based Security
Java SE 7	2013-05-28	JSR-342	Java SE 7	WebSocket, JSON and HTML5 Support
Java SE 6	2009-12-10	JSR-316	Java SE 6	CDI managed Beans and REST
Java EE 5	2006-05-11	JSR-244	Java SE 5	Java annotations
J2EE 1.4	2003-11-11	JSR-151	J2EE 1.4	WS-I interoperable web services
J2EE 1.3	2001-09-24	JSR-58	J2EE 1.3	Java connector architectures
J2EE 1.2	1999-12-17	1.2	J2EE 1.2	Initial specification release

Jakarta EE Application Model

- The Jakarta EE application model begins with the Java programming language and the Java virtual machine. The proven portability, security, and developer productivity they provide form the basis of the application model.
- Jakarta EE is designed to support applications that implement enterprise services for customers, employees, suppliers, partners, and others who make demands on or contributions to the enterprise. Such applications are inherently complex, potentially accessing data from a variety of sources and distributing applications to a variety of clients.
- The Jakarta EE application model defines an architecture for implementing services as multitier applications that deliver the scalability, accessibility, and manageability needed by enterprise-level applications. This model partitions the work needed to implement a multitier service into the following parts:
  - The business and presentation logic to be implemented by the developer
  - The standard system services provided by the Jakarta EE platform

Jakarta EE Components

- Jakarta EE applications are made up of components. A Jakarta EE component is a self-contained functional software unit that is assembled into a Jakarta EE application with its related classes and files and that communicates with other components.
- The Jakarta EE specification defines the following Jakarta EE components:
  - Application clients and applets are components that run on the client.
  - Jakarta Servlet, Jakarta Faces, and Jakarta Server Pages technology components are web components that run on the server.
  - Enterprise bean components (enterprise beans) are business components that run on the server.

Jakarta EE Clients

A Jakarta EE client is usually either a web client or an application client.

- Web Clients - a web client consists of two parts:
  - Dynamic web pages containing various types of markup language (HTML, XML, and so on), which are generated by web components running in the web tier
  - A web browser, which renders the pages received from the server
- Application Clients
  - An application client runs on a client machine and provides a way for users to handle tasks that require a richer user interface than can be provided by a markup language.
  - Application clients directly access enterprise beans running in the business tier. However, if application requirements warrant it, an application client can open an HTTP connection to establish communication with a servlet running in the web tier. Application clients written in languages other than Java can interact with Jakarta EE servers, enabling the Jakarta EE platform to interoperate with legacy systems, clients, and non-Java languages.

Jakarta EE Architecture

Distributed Multitiered Applications

- The Jakarta EE application parts:
  - Client-tier components run on the client machine.
  - Web-tier components run on the Jakarta EE server.
  - Business-tier components run on the Jakarta EE server.
  - Enterprise information system (EIS)-tier software runs on the EIS server.
- Although a Jakarta EE application can consist of all tiers, Jakarta EE multitiered applications are generally considered to be three-tiered applications because they are distributed over three locations: client machines, the Jakarta EE server machine, and the database or legacy machines at the back-end storage.

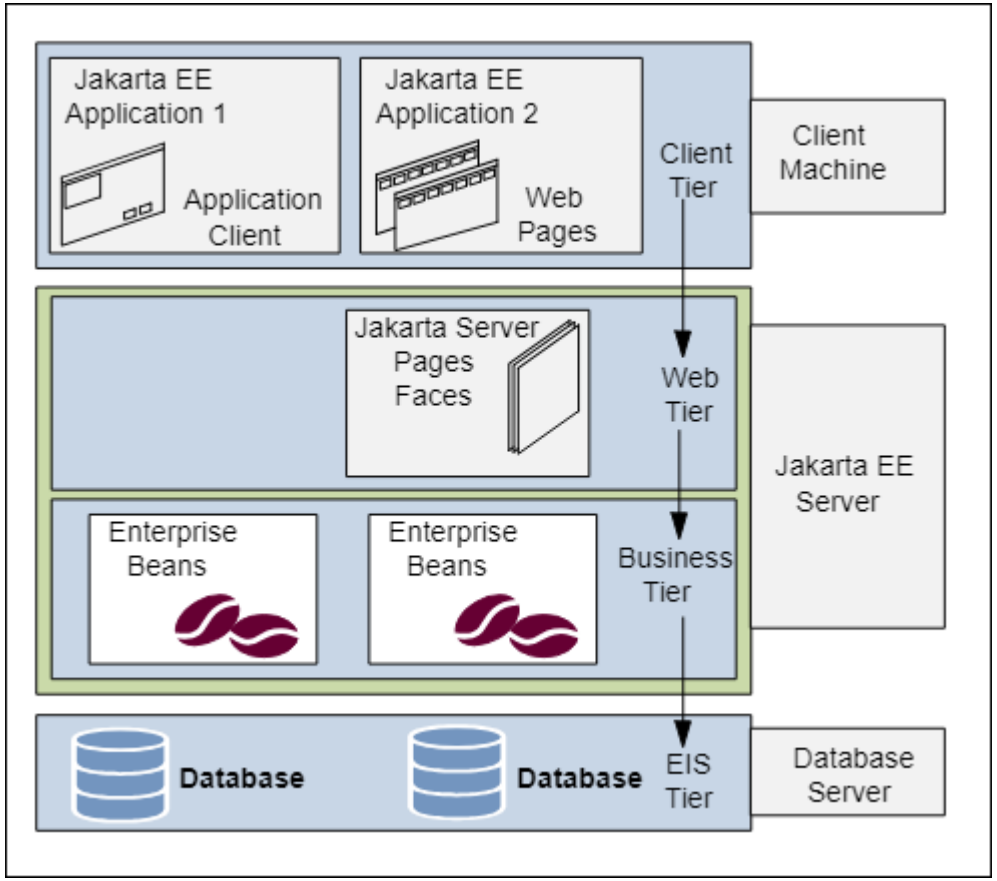


Figure 1-1 Multitiered Applications

Jakarta EE Architecture

Jakarta EE Server Communications

- The various elements that can make up the client tier.
- The client communicates with the business tier running on the Jakarta EE server either directly or, as in the case of a client running in a browser, by going through web pages or servlets running in the web tier.

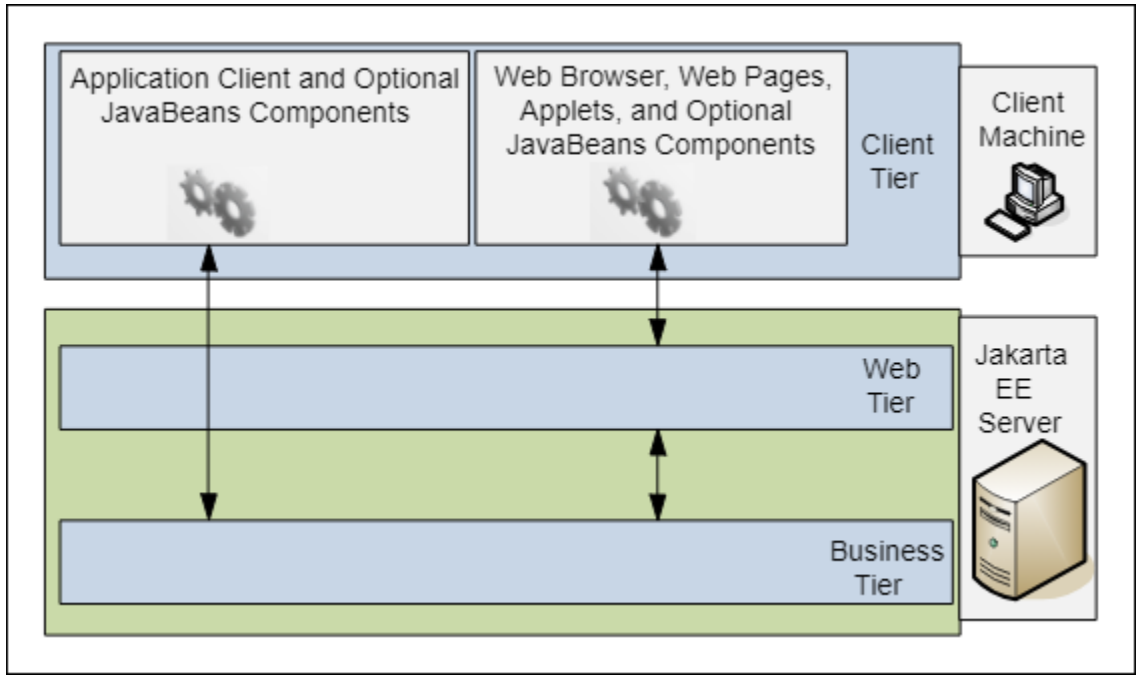


Figure 1-2 Server Communication

Jakarta EE Architecture

Web Components

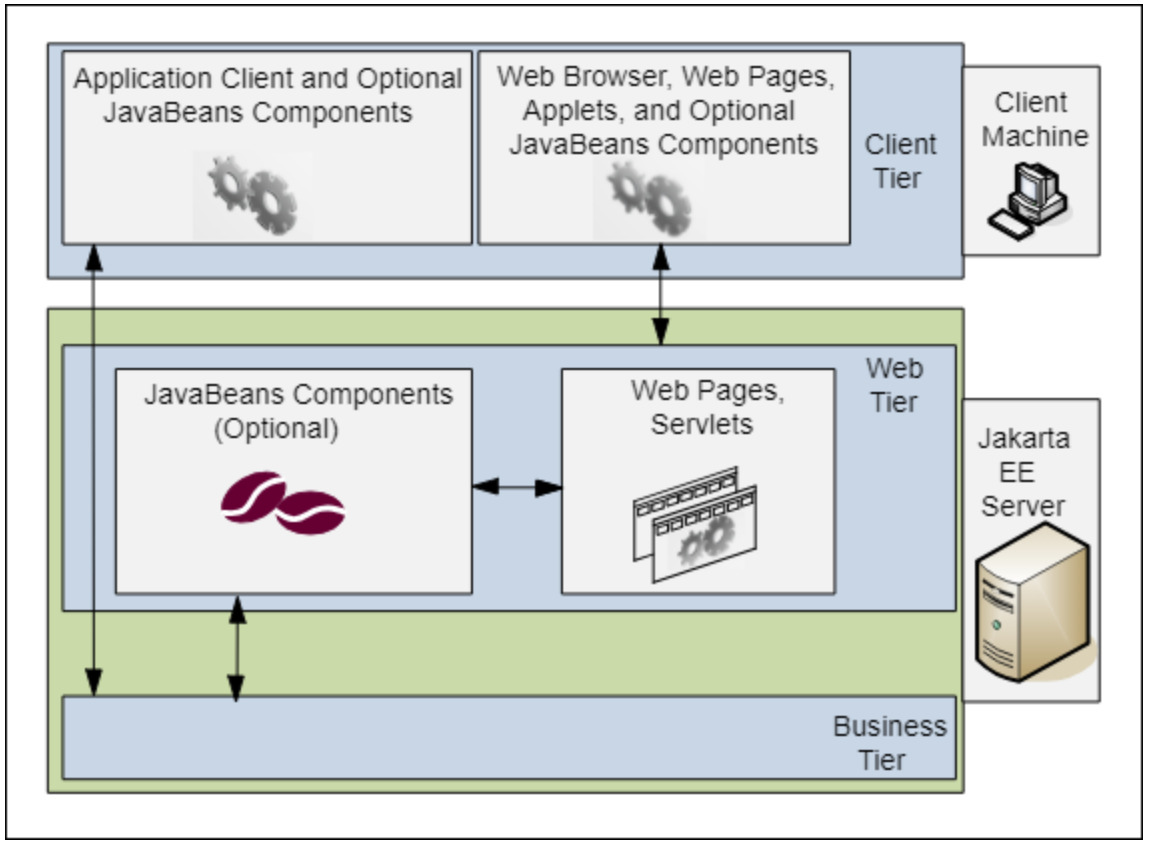


Figure 1-3 Web Tier and Jakarta EE Applications

Jakarta EE Architecture

Business Components

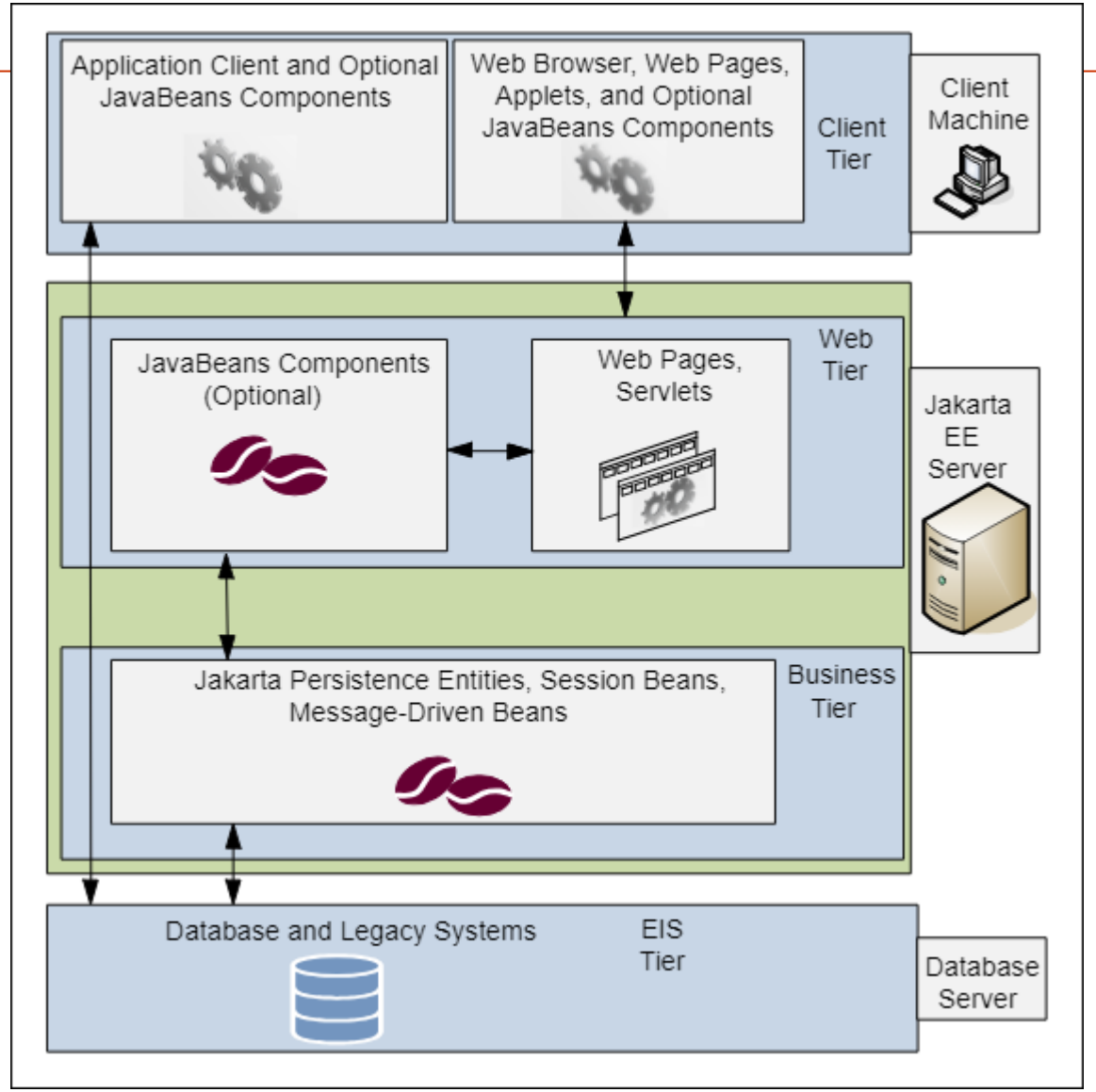


Figure 1-4 Business and EIS Tiers

Jakarta EE Containers

Container Services

- Containers are the interface between a component and the low-level, platform-specific functionality that supports the component. Before it can be executed, a web, enterprise bean, or application client component must be assembled into a Jakarta EE module and deployed into its container.
- The assembly process involves specifying container settings for each component in the Jakarta EE application and for the Jakarta EE application itself. Container settings customize the underlying support provided by the Jakarta EE server, including:
  - The Jakarta EE security model lets you configure a web component or enterprise bean so that system resources are accessed only by authorized users.
  - The Jakarta EE transaction model lets you specify relationships among methods that make up a single transaction so that all methods in one transaction are treated as a single unit.
  - Java Naming and Directory Interface (JNDI) lookup services provide a unified interface to multiple naming and directory services in the enterprise so that application components can access these services.
  - The Jakarta EE remote connectivity model manages low-level communications between clients and enterprise beans. After an enterprise bean is created, a client invokes methods on it as if it were in the same virtual machine.

Jakarta EE Containers

Container Types

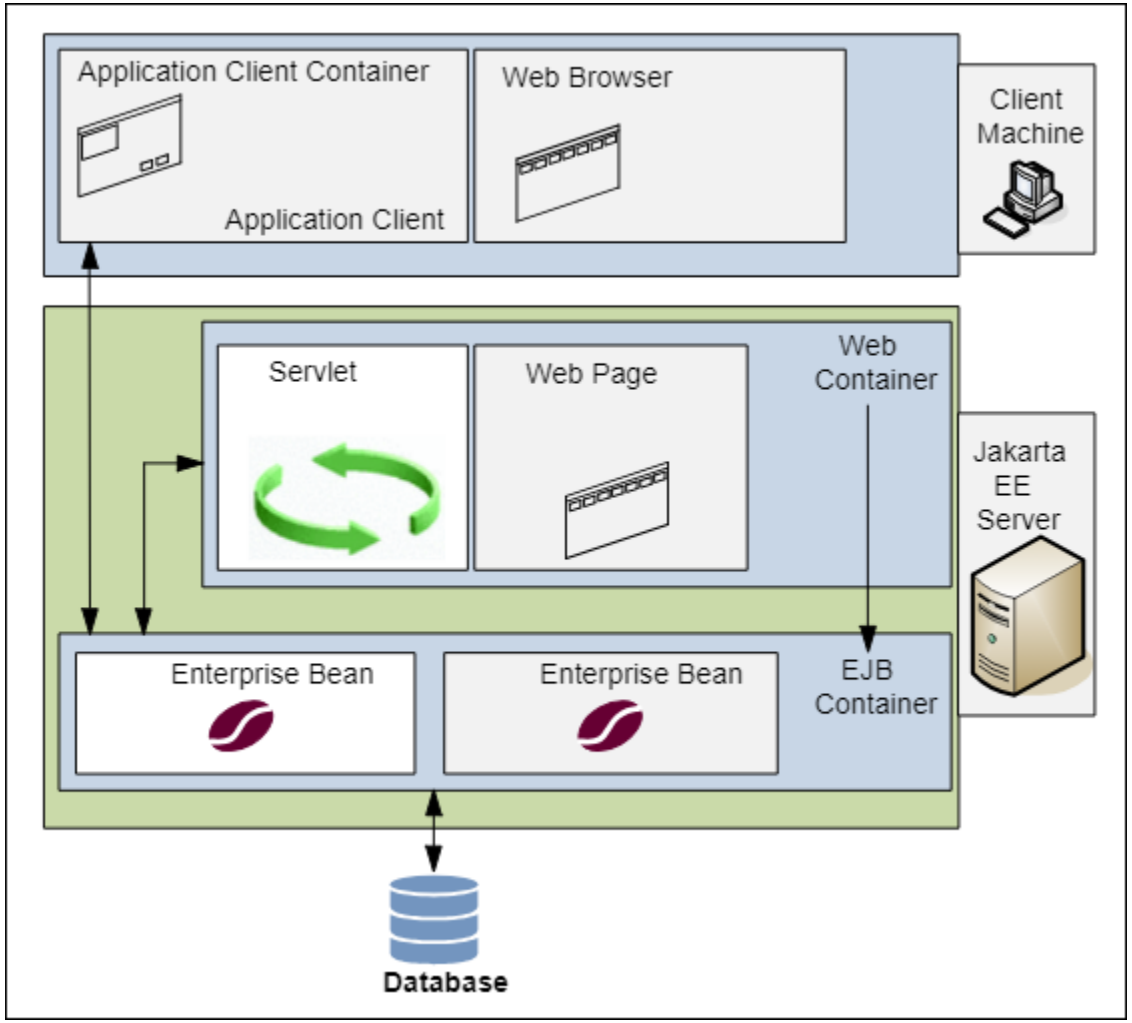


Figure 1-5 Jakarta EE Server and Containers

Jakarta EE Containers

Container Types

The server and containers are as follows:

- Jakarta EE server: The runtime portion of a Jakarta EE product. A Jakarta EE server provides enterprise and web containers.
- Jakarta Enterprise Bean container: Manages the execution of enterprise beans for Jakarta EE applications. Jakarta Enterprise Beans and their container run on the Jakarta EE server.
- Web container: Manages the execution of web pages, servlets, and some enterprise bean components for Jakarta EE applications. Web components and their container run on the Jakarta EE server.
- Application client container: Manages the execution of application client components. Application clients and their container run on the client.
- Applet container: Manages the execution of applets. Consists of a web browser and a Java Plug-in running on the client together.

Jakarta EE Containers

The relationships among the Jakarta EE Containers

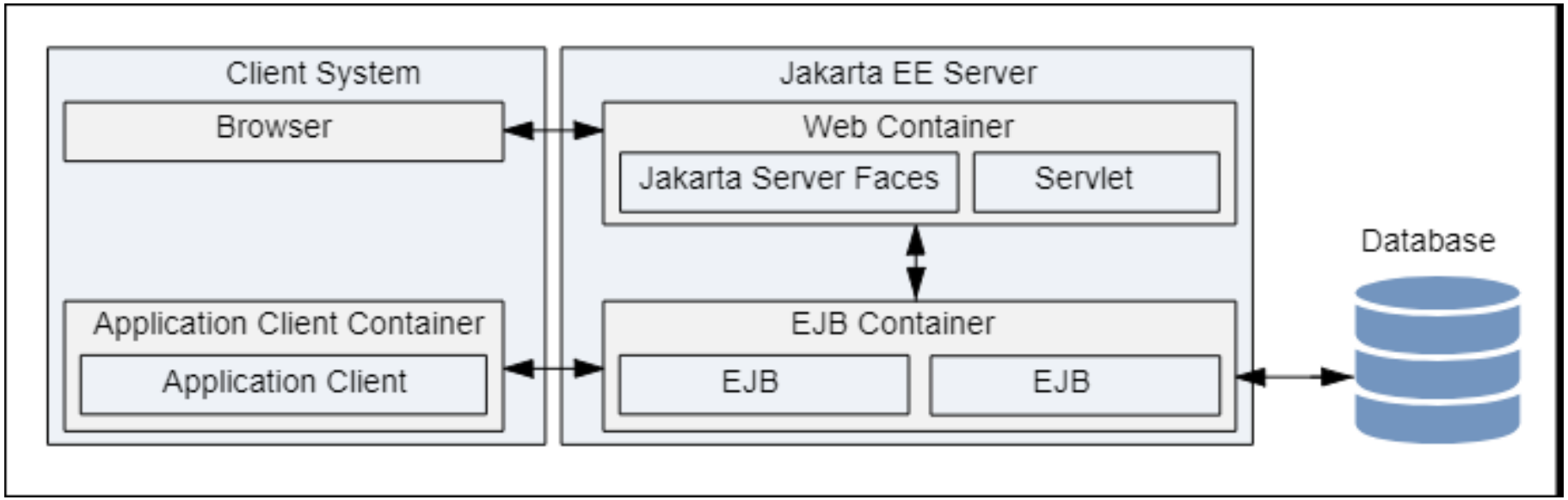


Figure 1-6 The Jakarta EE containers

Jakarta EE APIs

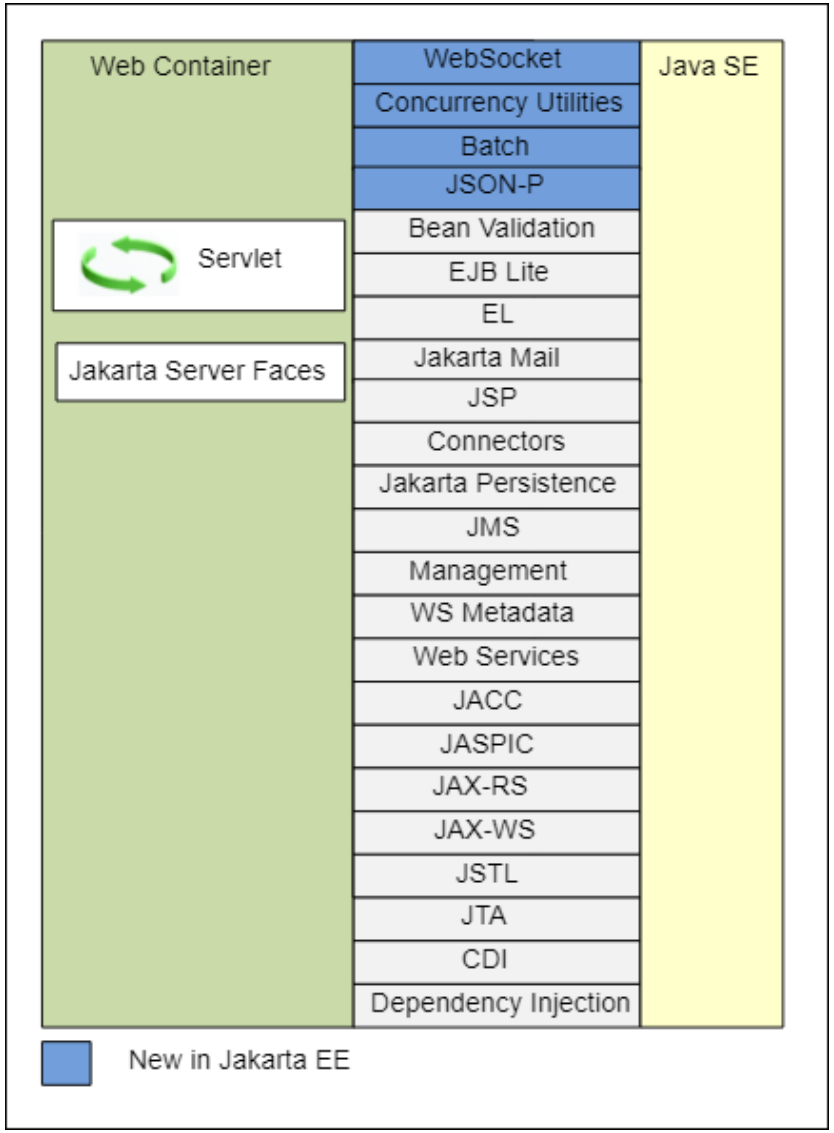


Figure 1-7 Jakarta EE APIs in the Web Container

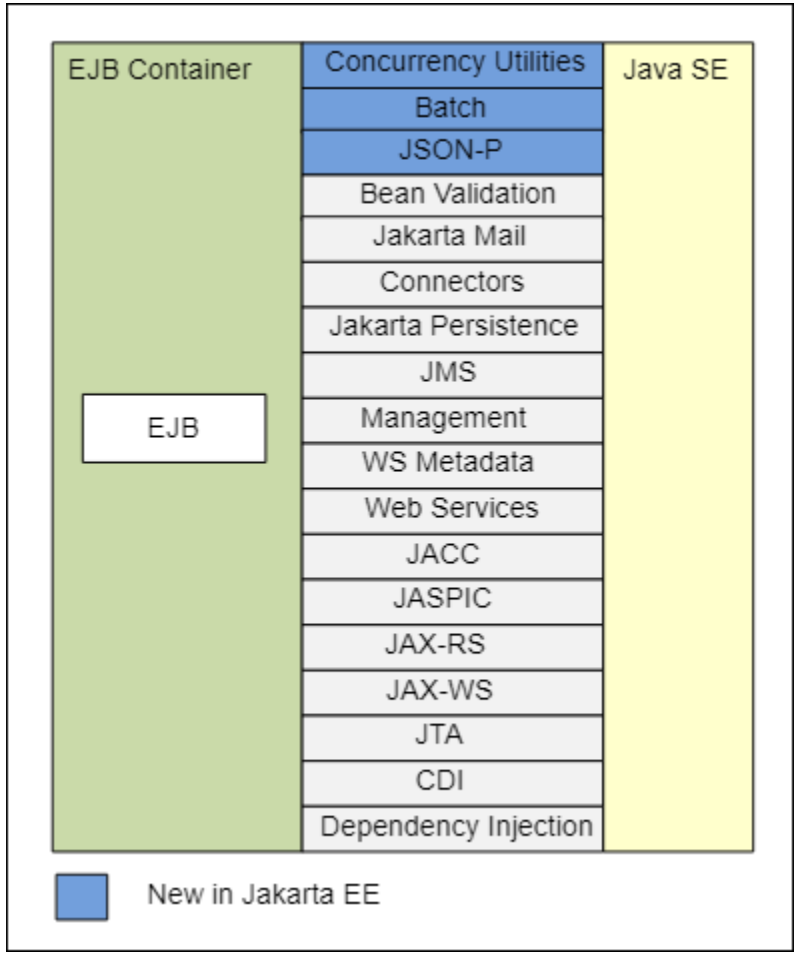


Figure 1-8 Jakarta EE APIs in the enterprise bean Container

Jakarta EE APIs

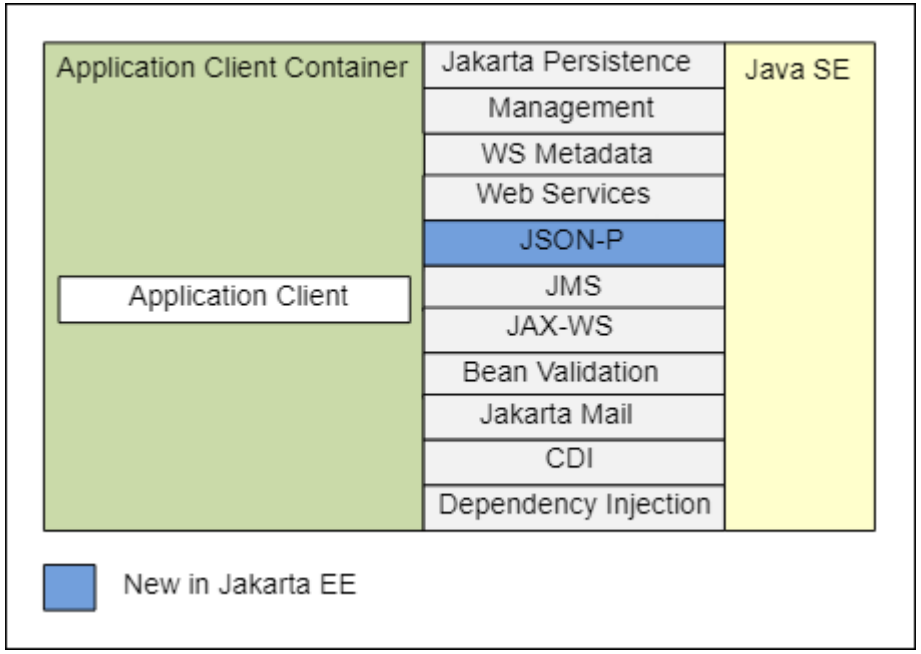


Figure 1-9 Jakarta EE APIs in the Application Client Container

Jakarta Enterprise Beans Technologies

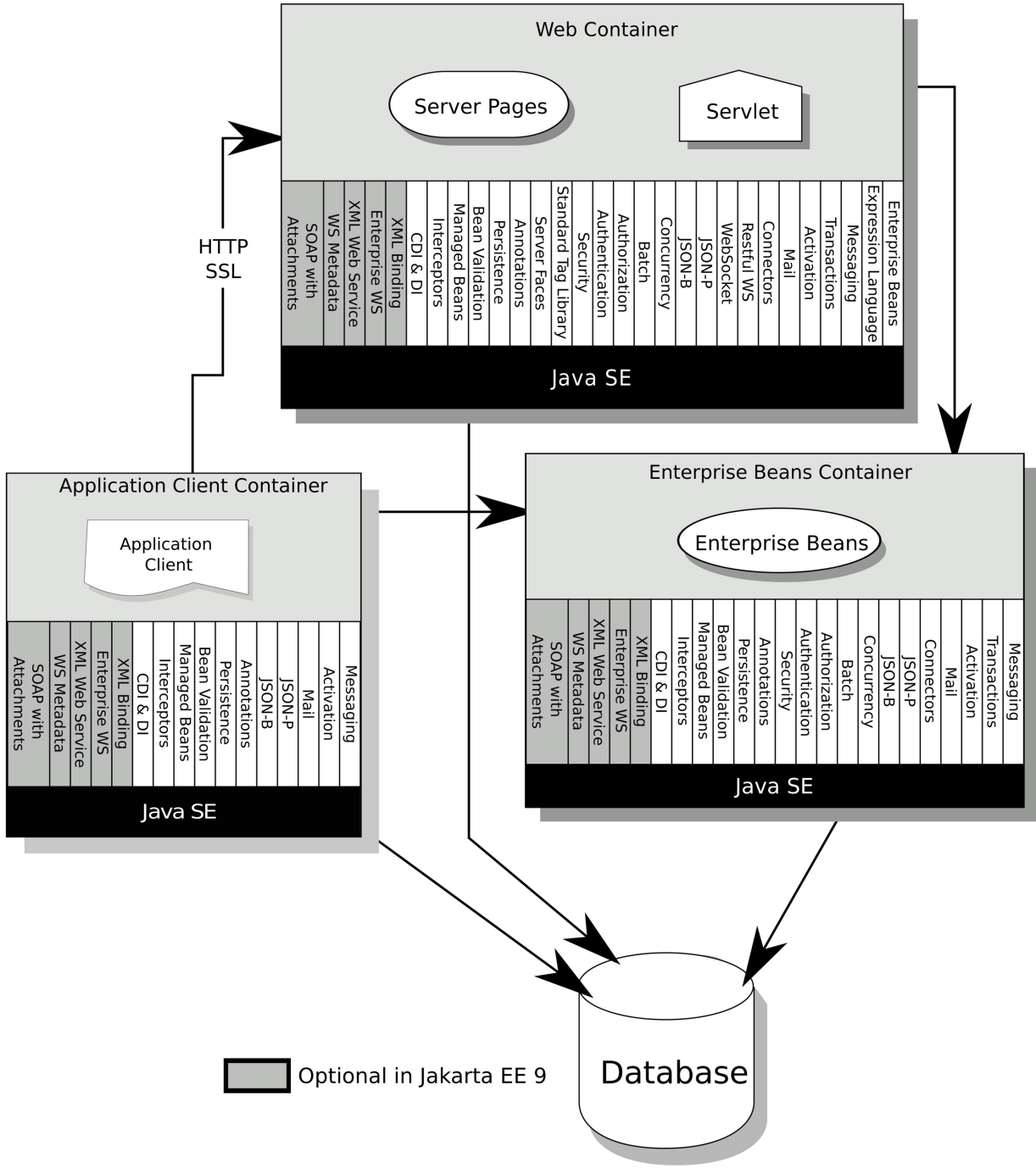
Jakarta Enterprise Beans Technologies	
Jakarta Servlet Technology	Jakarta Authorization
Jakarta Faces Technology	Jakarta Authentication
Jakarta Server Pages Technology	Jakarta Security
Jakarta Standard Tag Library	Jakarta WebSocket
Jakarta Persistence	Jakarta JSON Processing
Jakarta Transactions	Jakarta JSON Binding
Jakarta RESTful Web Services	Jakarta Concurrency
Jakarta Managed Beans	Jakarta Batch
Jakarta Contexts and Dependency Injection	Jakarta Activation
Jakarta Dependency Injection	Jakarta XML Binding
Jakarta Bean Validation	Jakarta XML Web Services
Jakarta Messaging	Jakarta SOAP with Attachments
Jakarta Connectors	Jakarta Annotations
Jakarta Mail	

Read more: https://eclipse-ee4j.github.io/jakartaee-tutorial/

Jakarta EE Architecture Diagram

https://jakarta.ee/specifications/platform/10/jakarta-platform-spec-10.0

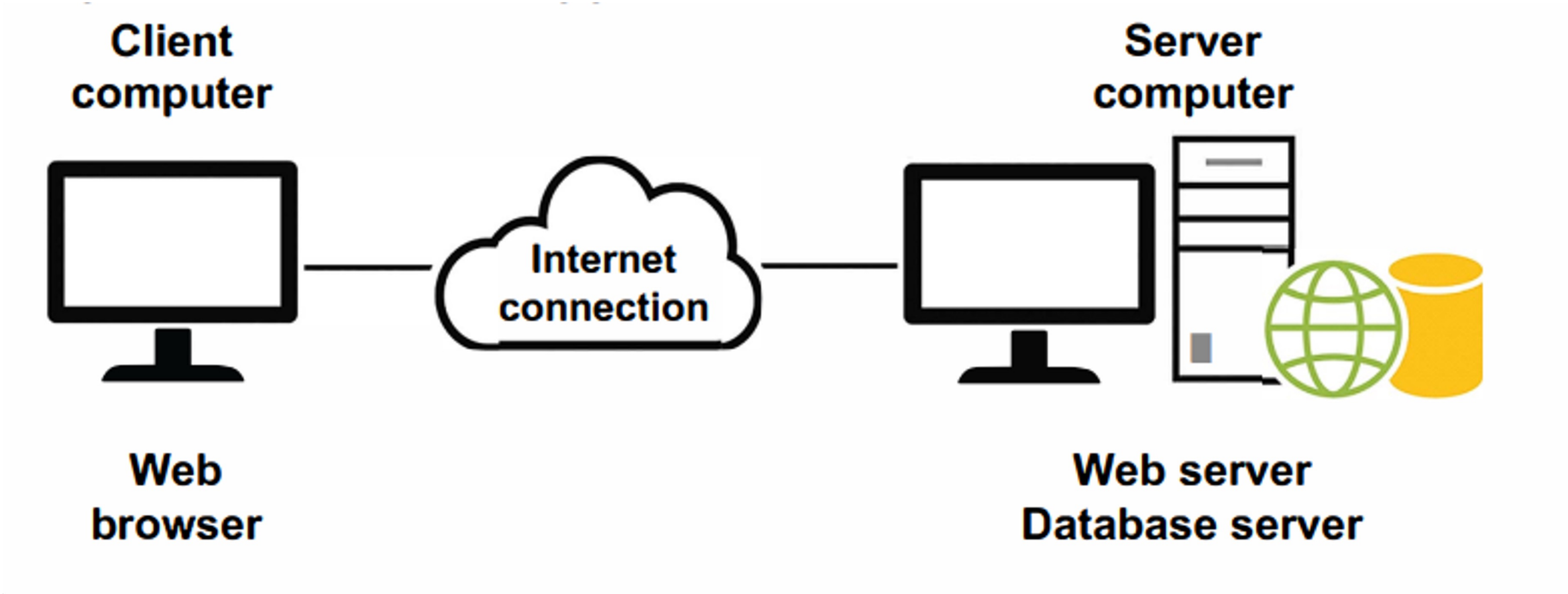
https://jakarta.ee/specifications/platform/11/jakarta-platform-spec-11.0



Web Application Architecture

- A web app architecture presents a layout with all the software components (such as databases, applications and middleware) and how they interact with each other.
- It defines how the data is delivered through HTTP and ensures that the client-side server and the backend server can understand.
- Moreover it also secures that valid data is present in all user requests. It creates and manages records while providing permission-based access and authentication.

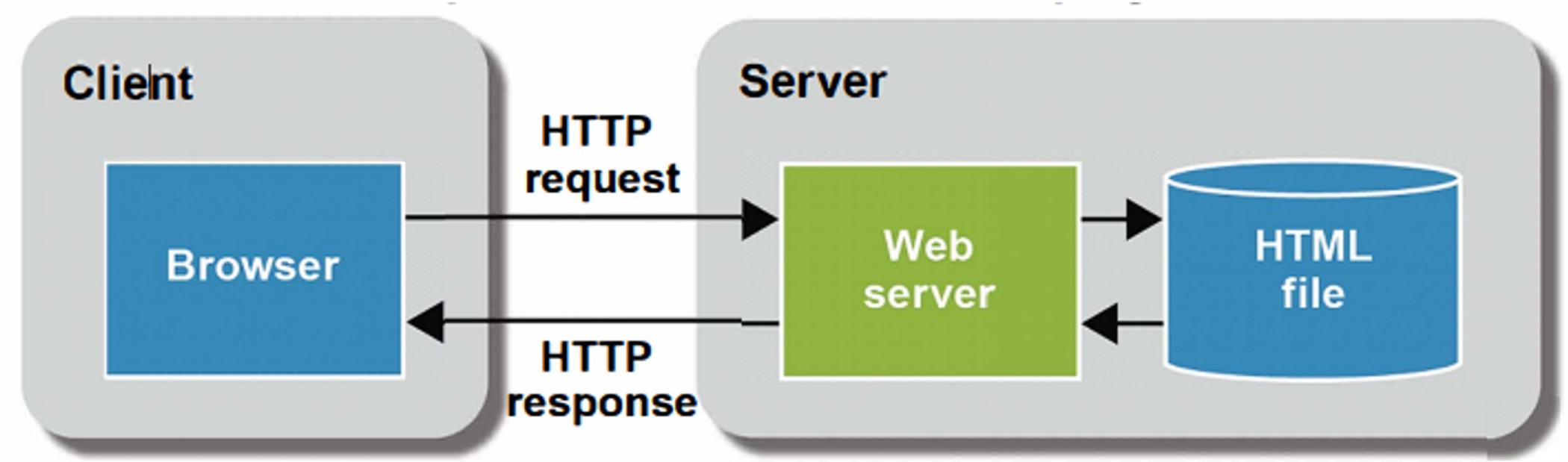
Web Application Architecture Components



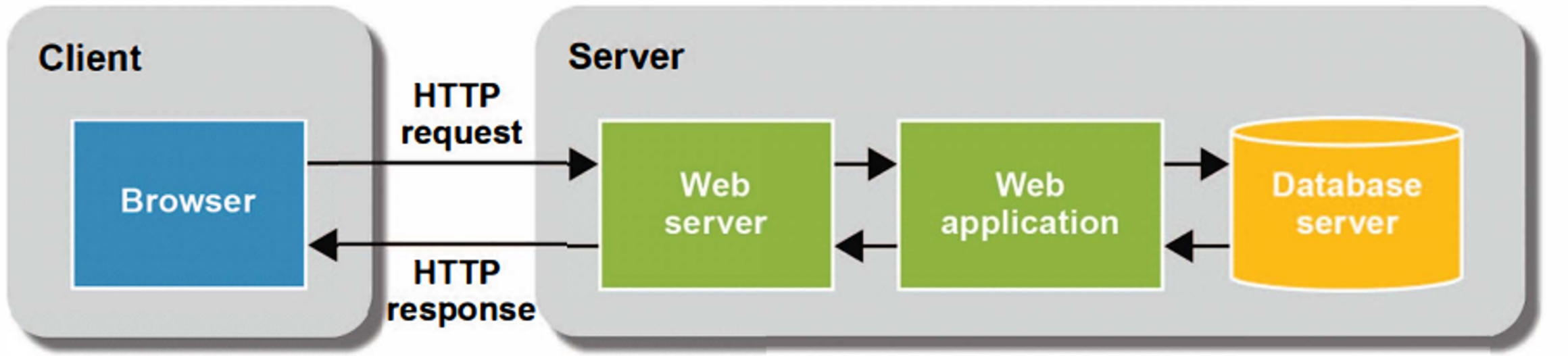
Web Application Architecture Components (cont.)

1) **Web Browser**: The browser or the client-side component or the front-end component is the key component that interacts with the user, receives the input and manages the presentation logic while controlling user interactions with the application. User inputs are validated as well, if required.
2) **Web Server:** The web server also known as the backend component or the server-side component handles the business logic and processes the user requests by routing the requests to the right component and managing the entire application operations. It can run and oversee requests from a wide variety of clients.
3) **Database Server**: The database server provides the required data for the application. It handles data-related tasks. In a multi-tiered architecture, database servers can manage business logic with the help of stored procedures.

Static web pages



Dynamic web pages



- A dynamic web page is an HTML document that's generated by a web application.
- The web page changes according to parameters that are sent to the web application by the web browser.

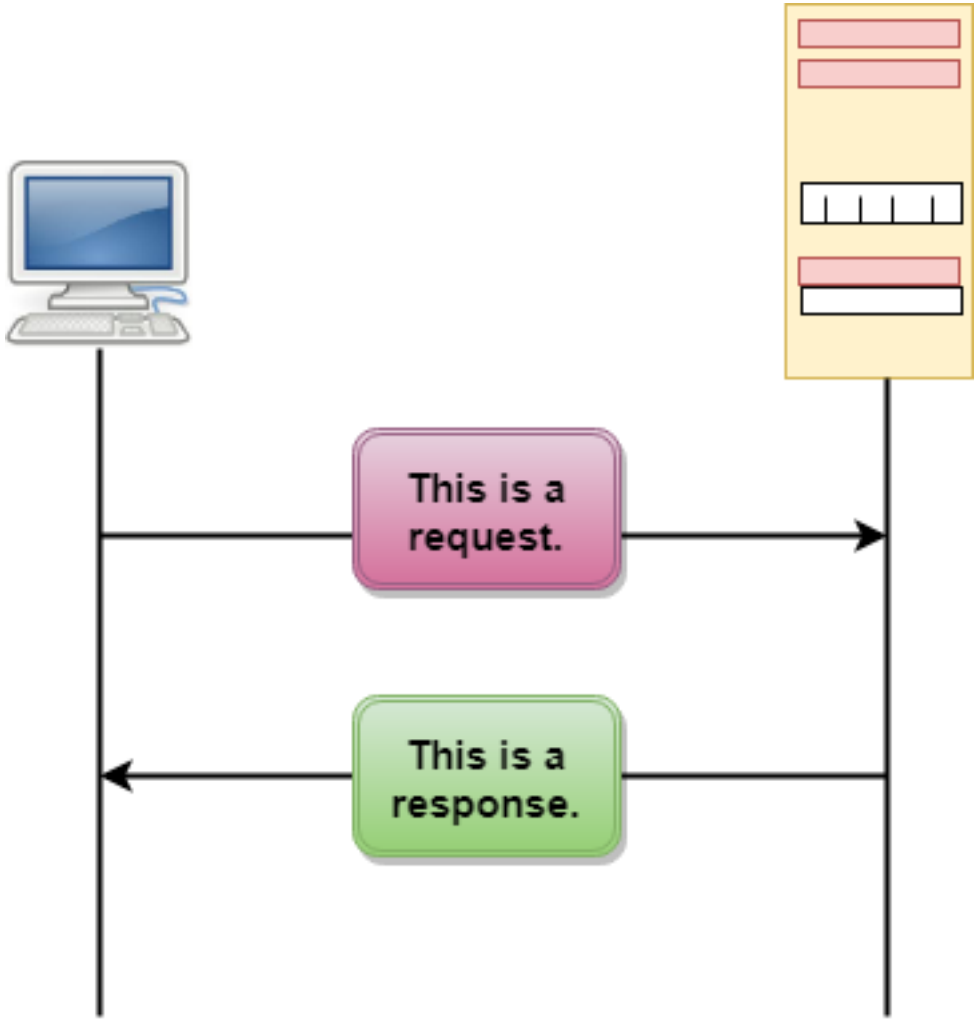
HTTP – Hypertext Transfer Protocol

- It is a protocol used to access the data on the World Wide Web (www).
- The HTTP protocol can be used to transfer the data in the form of plain text, hypertext, audio, video, and so on.
- This protocol is known as Hypertext Transfer Protocol because of its efficiency that allows us to use in a hypertext environment where there are rapid jumps from one document to another document.

Feature of HTTP

1) **Connectionless protocol:** HTTP is a connectionless protocol. HTTP client initiates a request and waits for a response from the server. When the server receives the request, the server processes the request and sends back the response to the HTTP client after which the client disconnects the connection. The connection between client and server exist only during the current request and response time only.
2) **Media independent:** HTTP protocol is a media independent as data can be sent as long as both the client and server know how to handle the data content. It is required for both the client and server to specify the content type in MIME-type header.
3) **Stateless:** HTTP is a stateless protocol as both the client and server know each other only during the current request. Due to this nature of the protocol, both the client and server do not retain the information between various requests of the web pages.

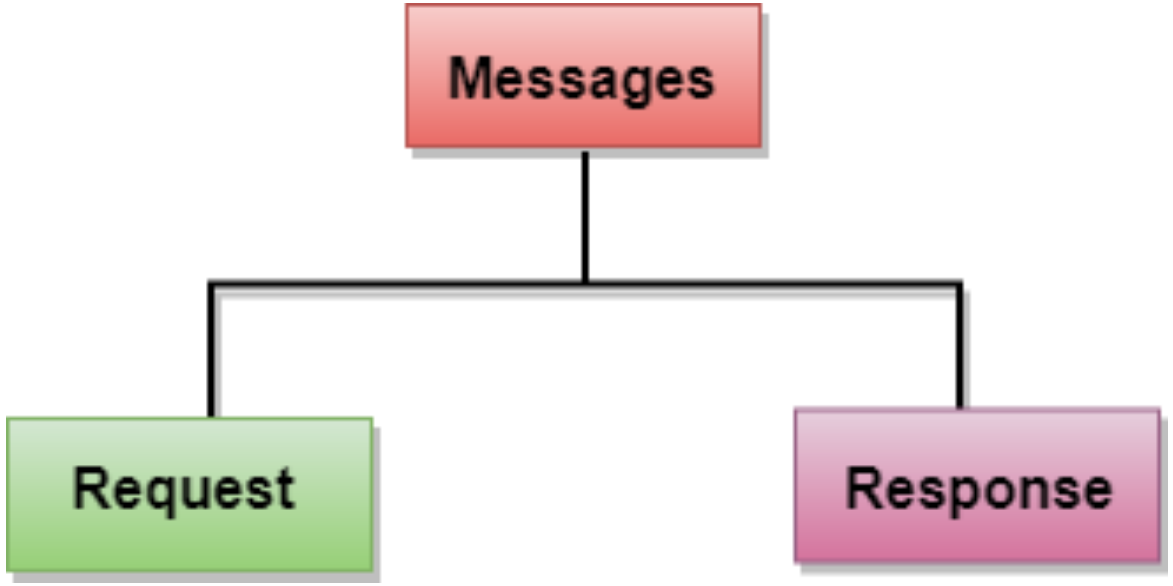
HTTP Transactions



- The client initiates a transaction by sending a request message to the server.
- The server replies to the request message by sending a response message.

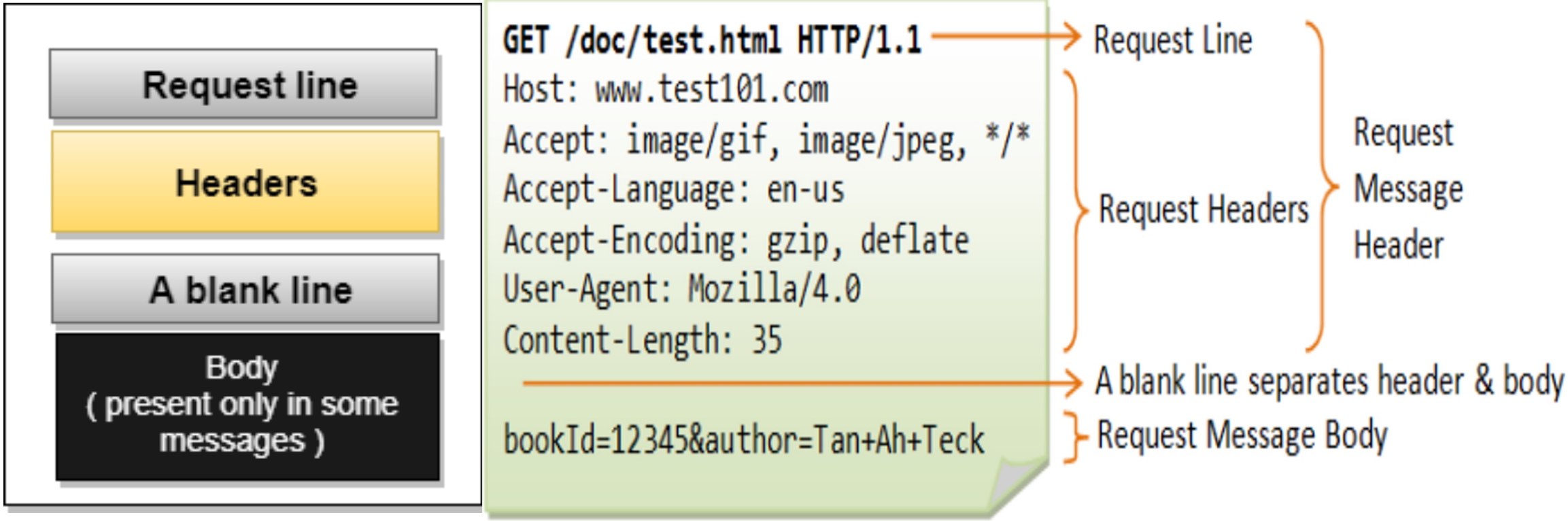
HTTP Messages

- Two types: request and response. Both the message types follow the same message format



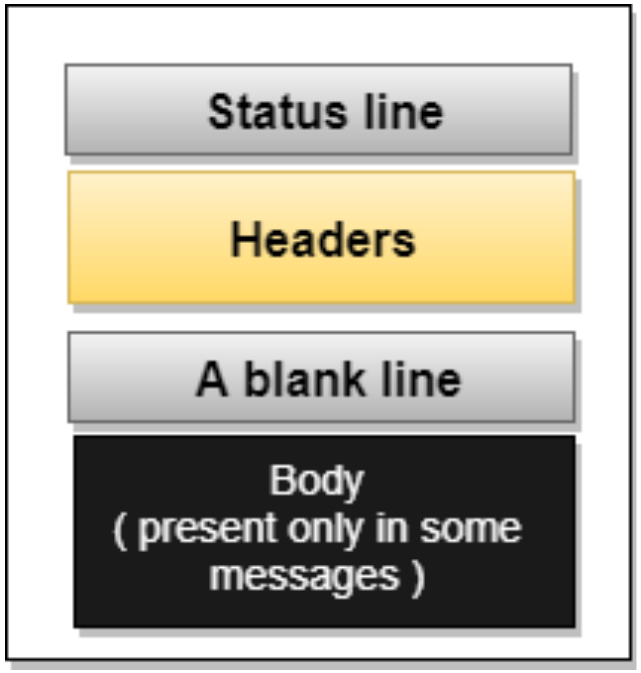
Request Message

- The request message is sent by the client that consists of a request line, headers, and sometimes a body.



Response Message

- The response message is sent by the server to the client that consists of a status line, headers, and sometimes a body.




```
HTTP/1.1 200 OK
Date: Sun, 08 Feb xxxx 01:11:12 GMT
Server: Apache/1.3.29 (Win32)
Last-Modified: Sat, 07 Feb xxxx
ETag: "0-23-4024c3a5"
Accept-Ranges: bytes
Content-Length: 35
Connection: close
Content-Type: text/html

<h1>My Home page</h1>
```

Status Line

Response Headers

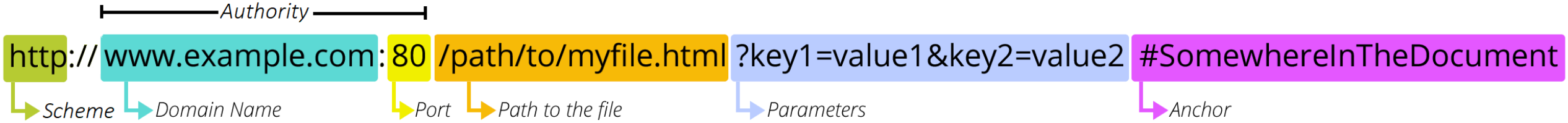
Response Message Header

A blank line separates header & body

Response Message Body

URL – Uniform Resource Locator

- A client that wants to access the document in an internet needs an address and to facilitate the access of documents, the HTTP uses the concept of Uniform Resource Locator (URL).
- The Uniform Resource Locator (URL) is a standard way of specifying any kind of information on the internet.
- The URL defines four parts: method, host computer, port, and path.



URL – Uniform Resource Locator (cont.)

- protocol://[host.]domain[:port][/context][/resource][?query string]

Or

- protocol://IP address[:port][/context][/resource][?query string]

The approaches for Java web applications

1) Servlet/JSP:

- Servlets store the Java code that does the server-side processing
- Jakarta Server Pages (JSPs) store the HTML that defines the user interface
- Is a lower-level API that does less work for the programmer.
- Provides a high degree of control over the HTML/CSS/JavaScript that's returned to the browser.

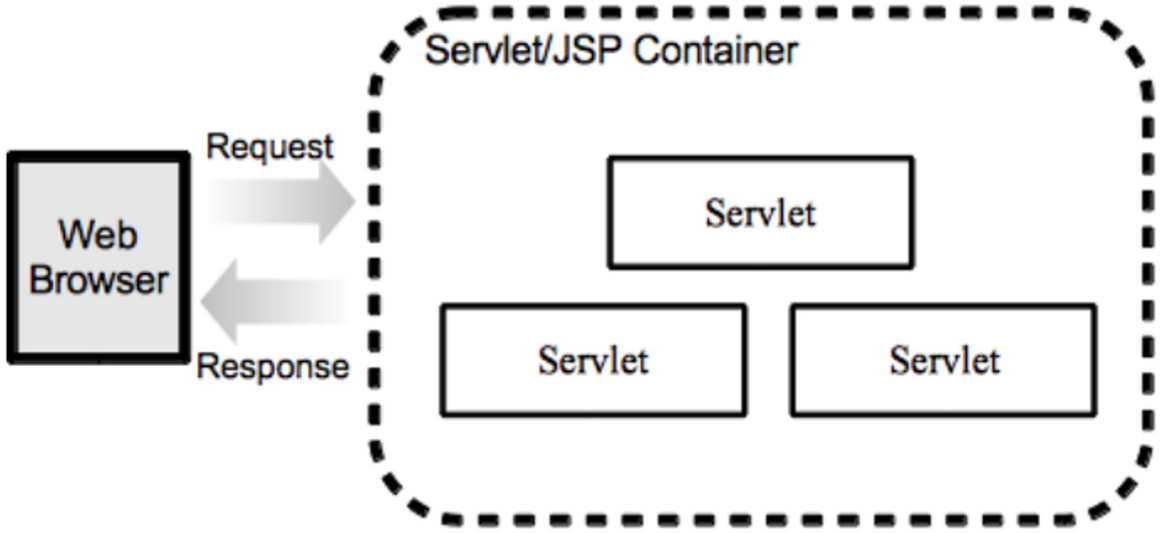
2) Jakarta Server Faces (JSF):

- A newer technology that's designed to replace both servlets and JSPs. It provides a higher-level API that does more work for the programmer
- Makes it more difficult to control the HTML/CSS/JavaScript that's returned to the browser.

3) Spring Framework:

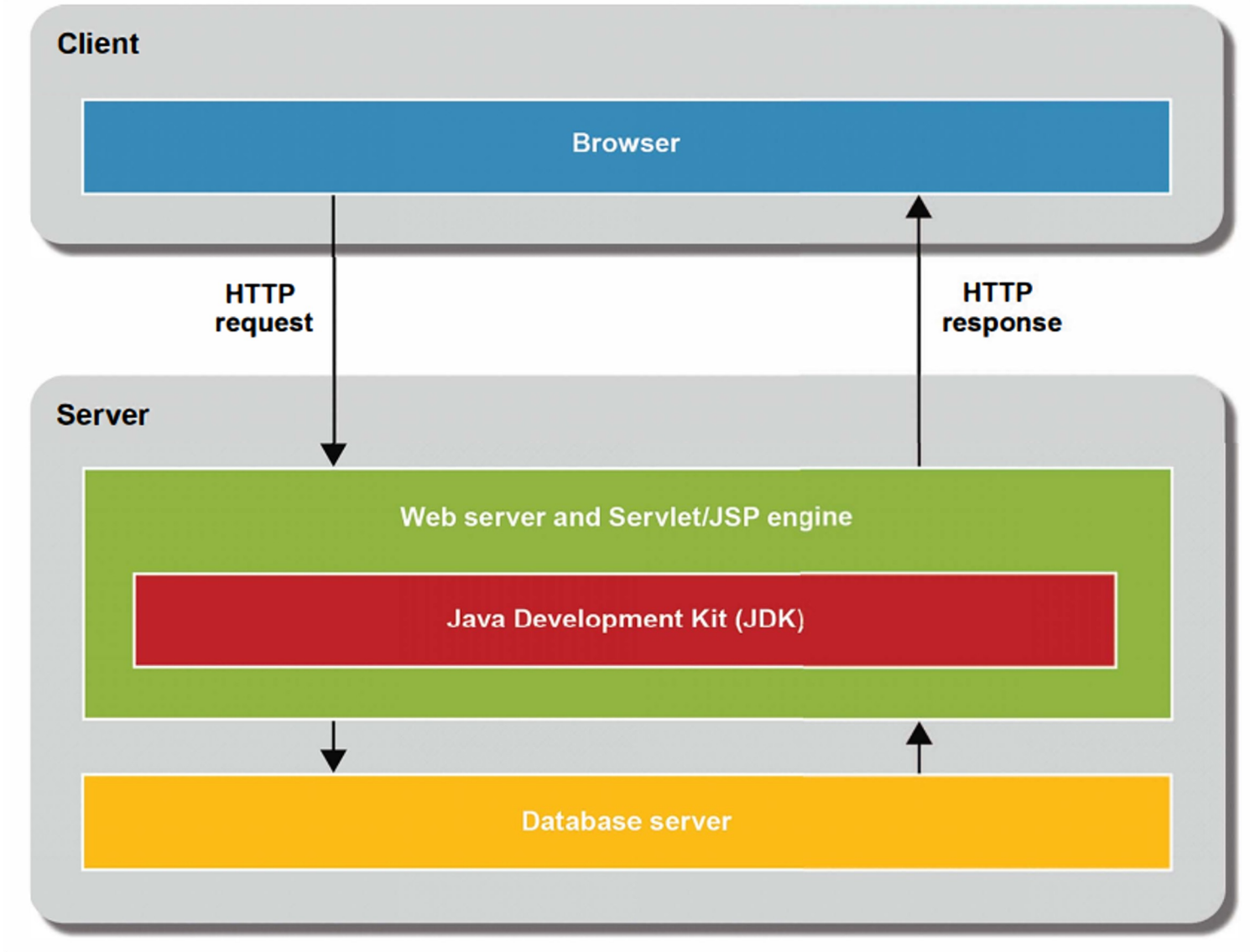
- Is a higher-level API that does more work for the programmer.
- Provides a high degree of control over the HTML/CSS/JavaScript that's returned to the browser

Servlet/JSP application architecture

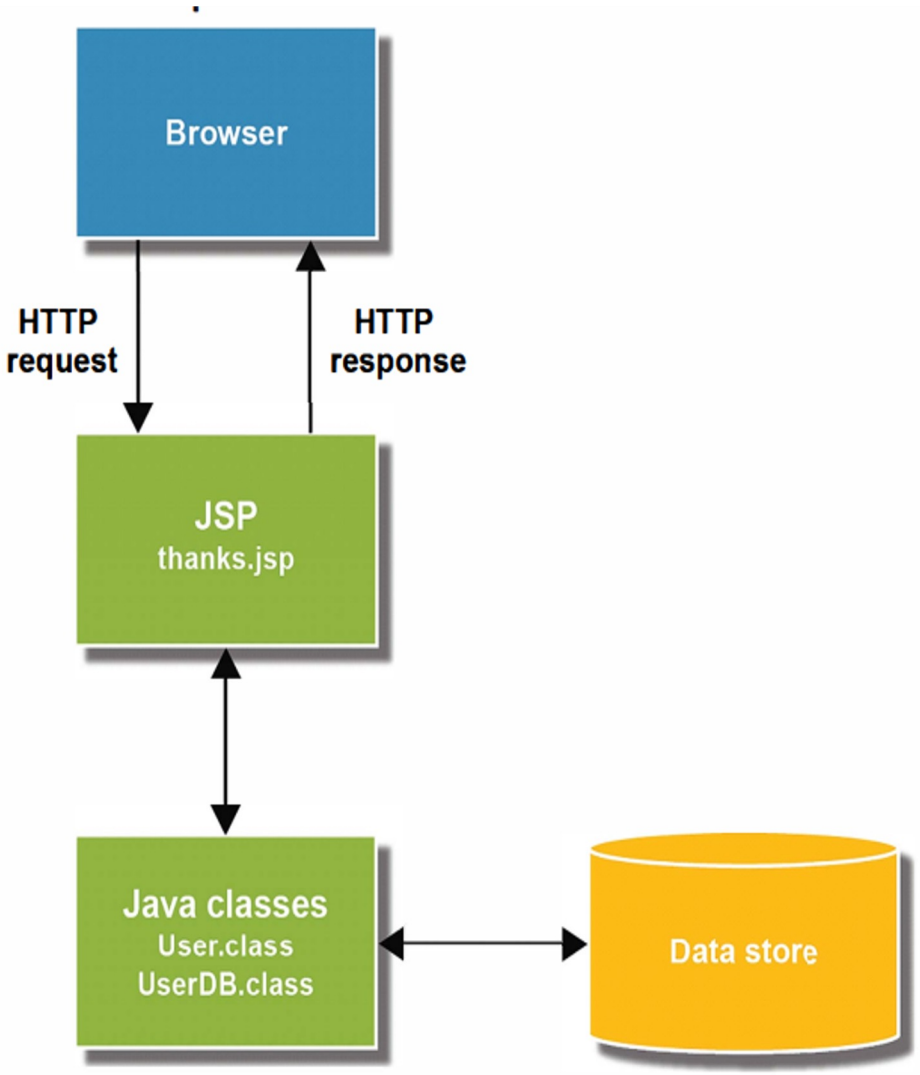


- A servlet/JSP container is a special web server that can process servlets as well as serve static contents
- Apache Tomcat and Jetty are the most popular servlet/JSP containers and are free and open-source

Servlet/JSP application architecture (cont.)



Two patterns for servlet/JSP applications

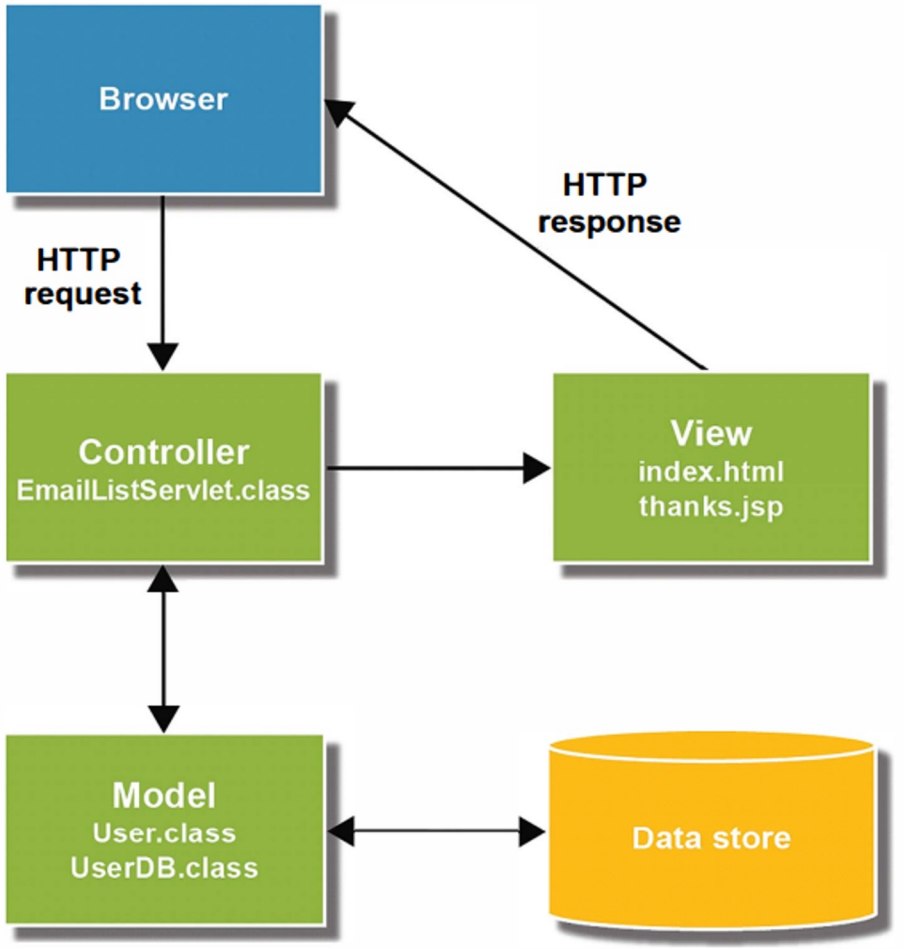


1) The Model 1 pattern:

- JSPs handles all of the processing and presentation for the application.

- This pattern is sometimes adequate for web applications with limited processing requirements, but it is generally considered a **bad practice**.

Two patterns for servlet/JSP applications (cont.)



2) The Model 2 (MVC) pattern:

- The Model 2 pattern separates the code into a model, a view, and a controller. As a result, it's also known as the Model-View-Controller (MVC) pattern.
- In the MVC pattern, the model consists of business objects like the User object, the view consists of HTML pages and JSPs, and the controller consists of servlets.
- In MVC pattern, if you need to make changes to one layer, any changes to the other layers are minimized.

IDE and Server for developing Java web applications

1) SDK:
   - Oracle JDK 8/11/**17**
   - OpenJDK
   - …
2) IDE:
   - **IntelliJ IDE**
   - Eclipse IDE for Java EE Developers
   - Apache NetBeans
3) Build tool:
   - **Maven**
   - Gradle
4) Servers:
   - **Apache Tomcat Server**
   - Payara Server
   - Glassfish