



Pokročilé informační systémy

Backend a platforma Jakarta EE

doc. Ing. Radek Burget, Ph.D.

burgetr@fit.vutbr.cz

Schéma třívrstvé architektury

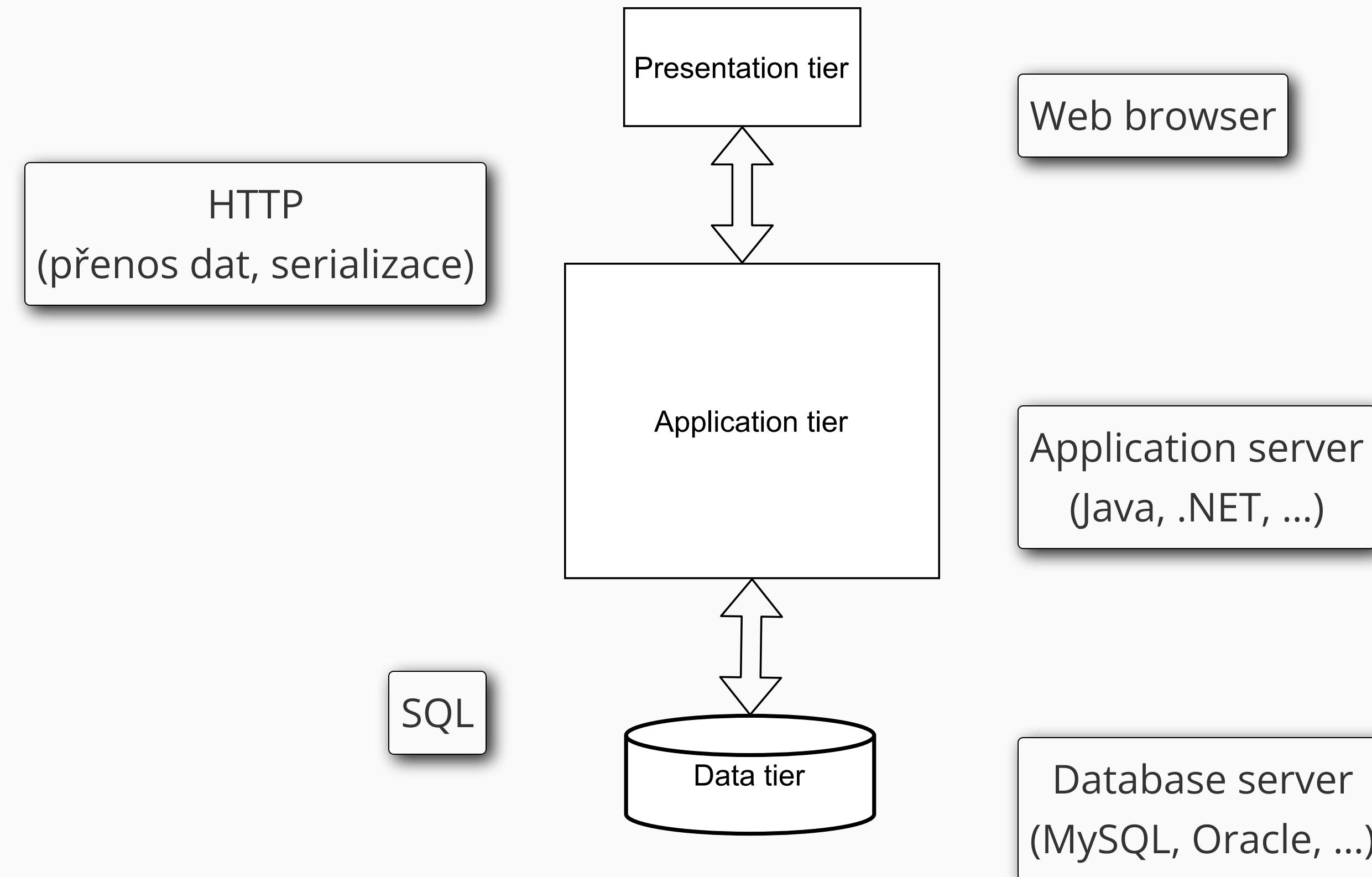
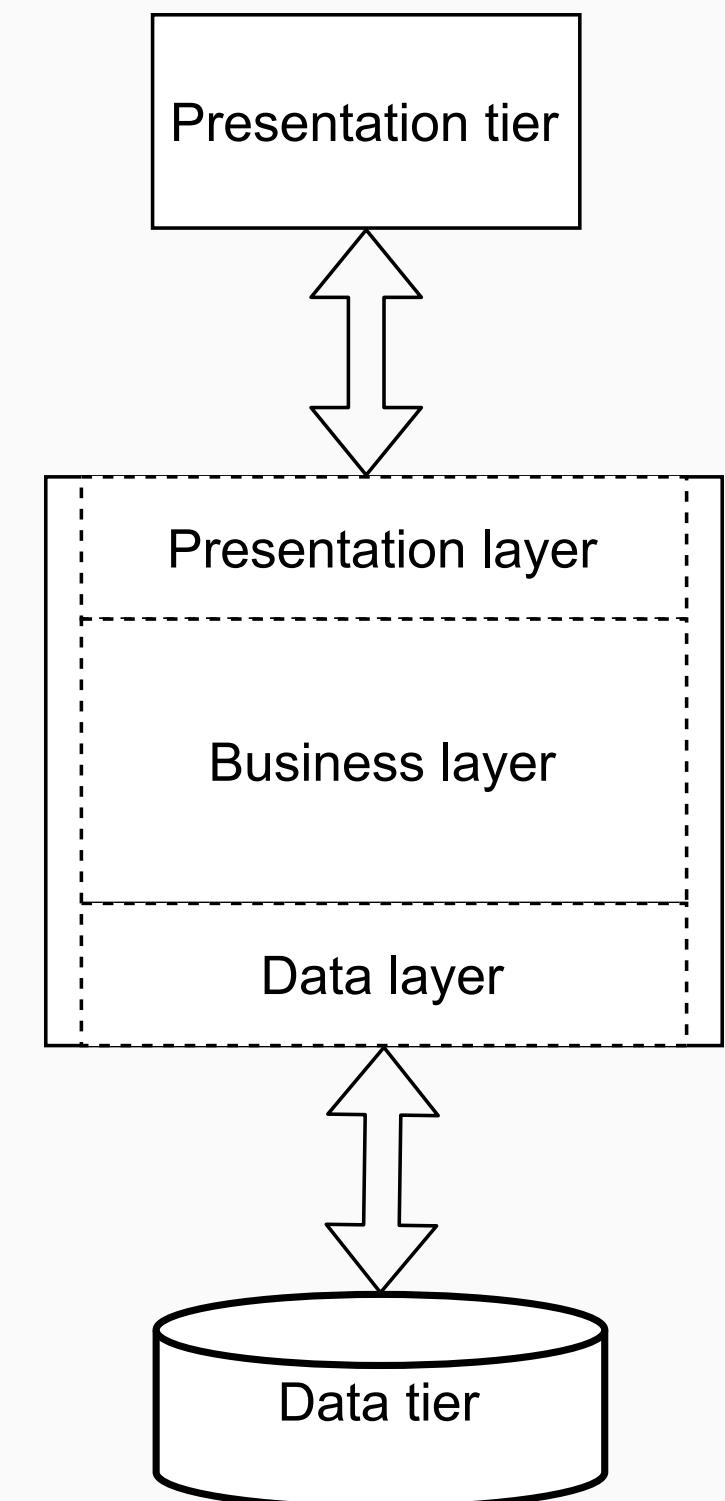


Schéma třívrstvé architektury (II)



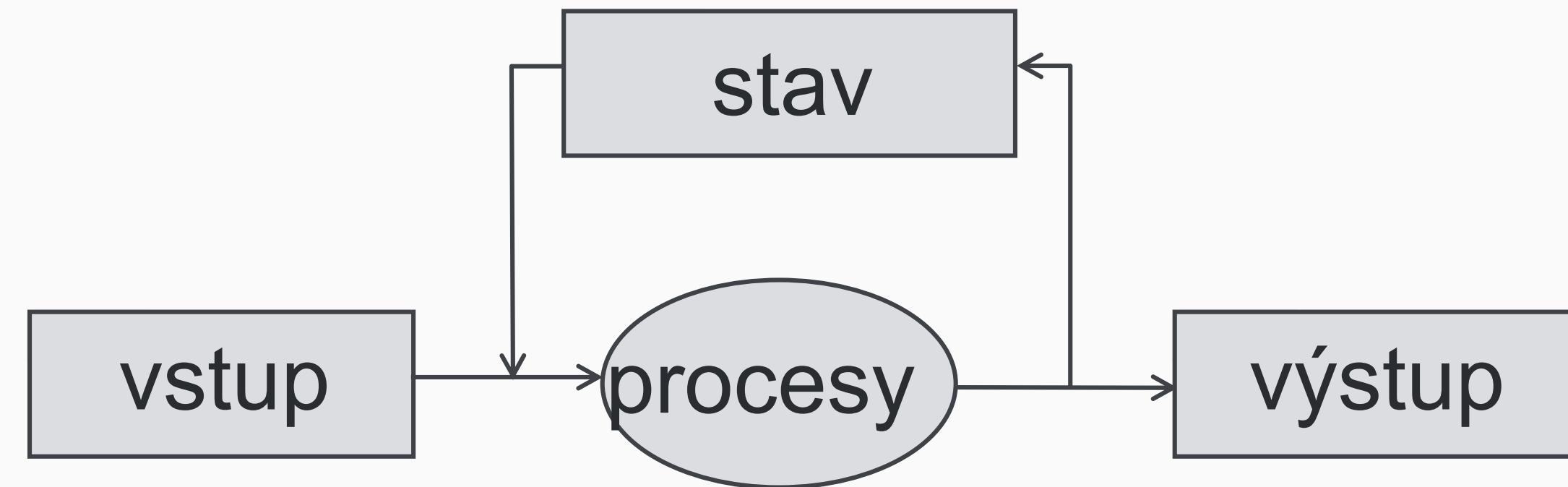
Tenčí nebo tlustší klient v prohlížeči

Java, .NET, PHP ...
Různá rámcová řešení (framework)

Datový model (objektový, relační, ...)

Informační systém a vrstvy

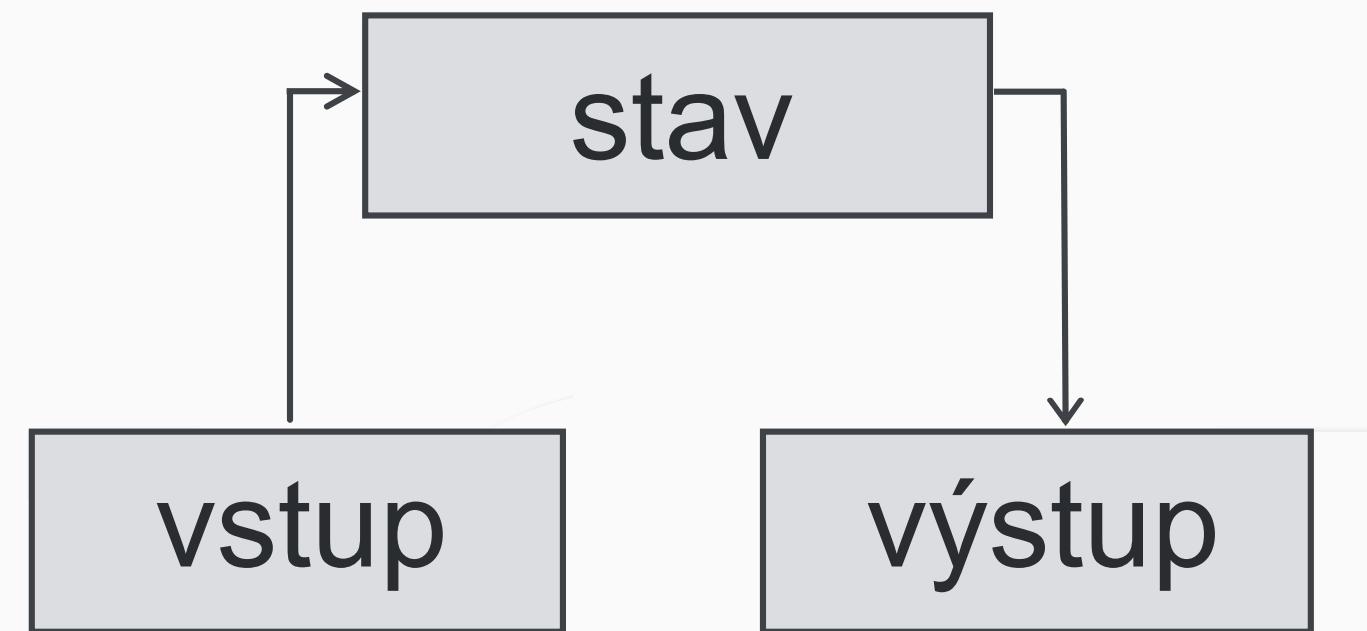
- **Data layer**
- Třídy reprezentující data, databázové operace



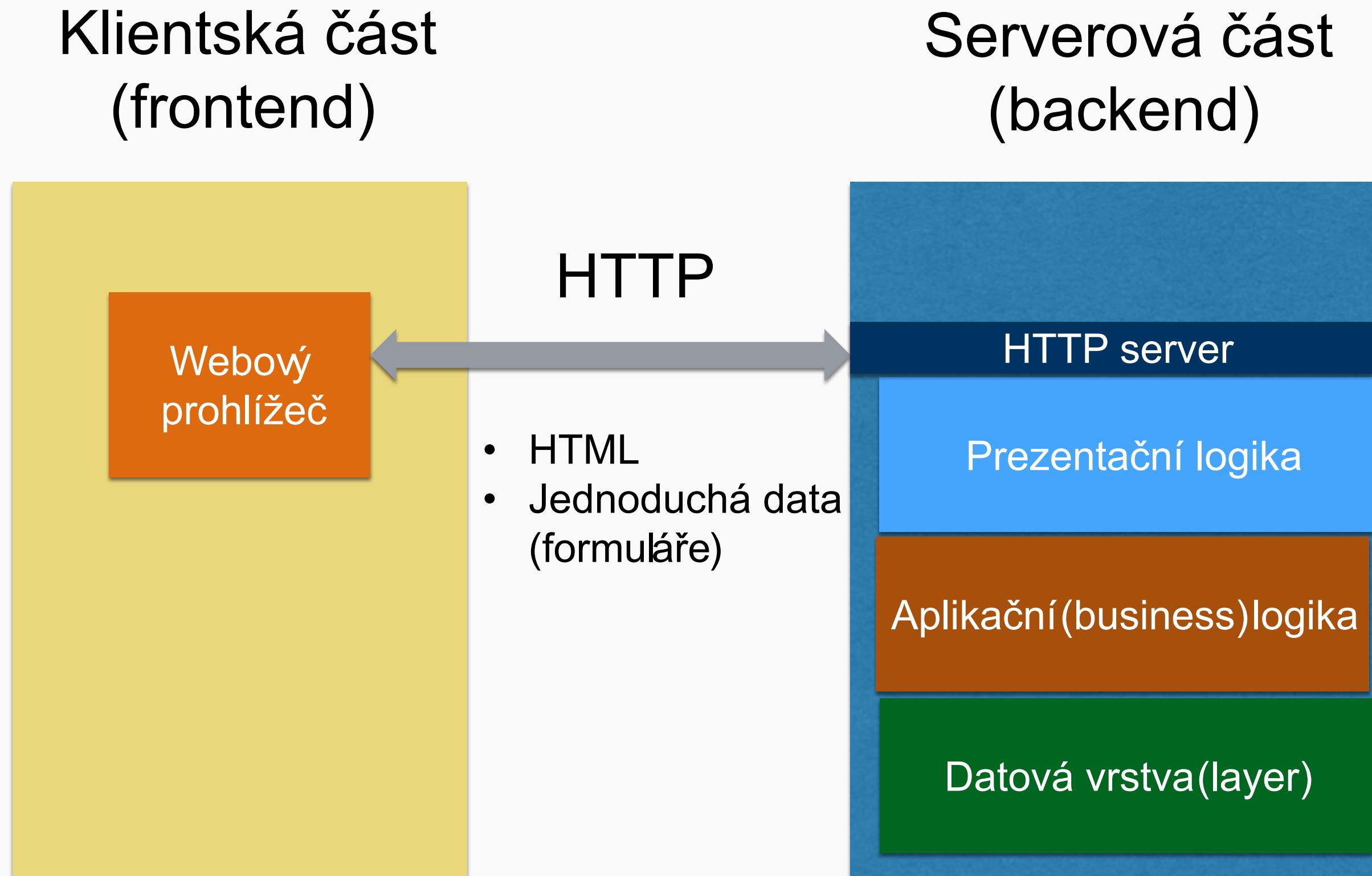
- Prezentační vrstva
- Vstupní prvky, **logika GUI**

- **Business vrstva**
- Business funkce (transakční zpracování dat, **logika GUI**)

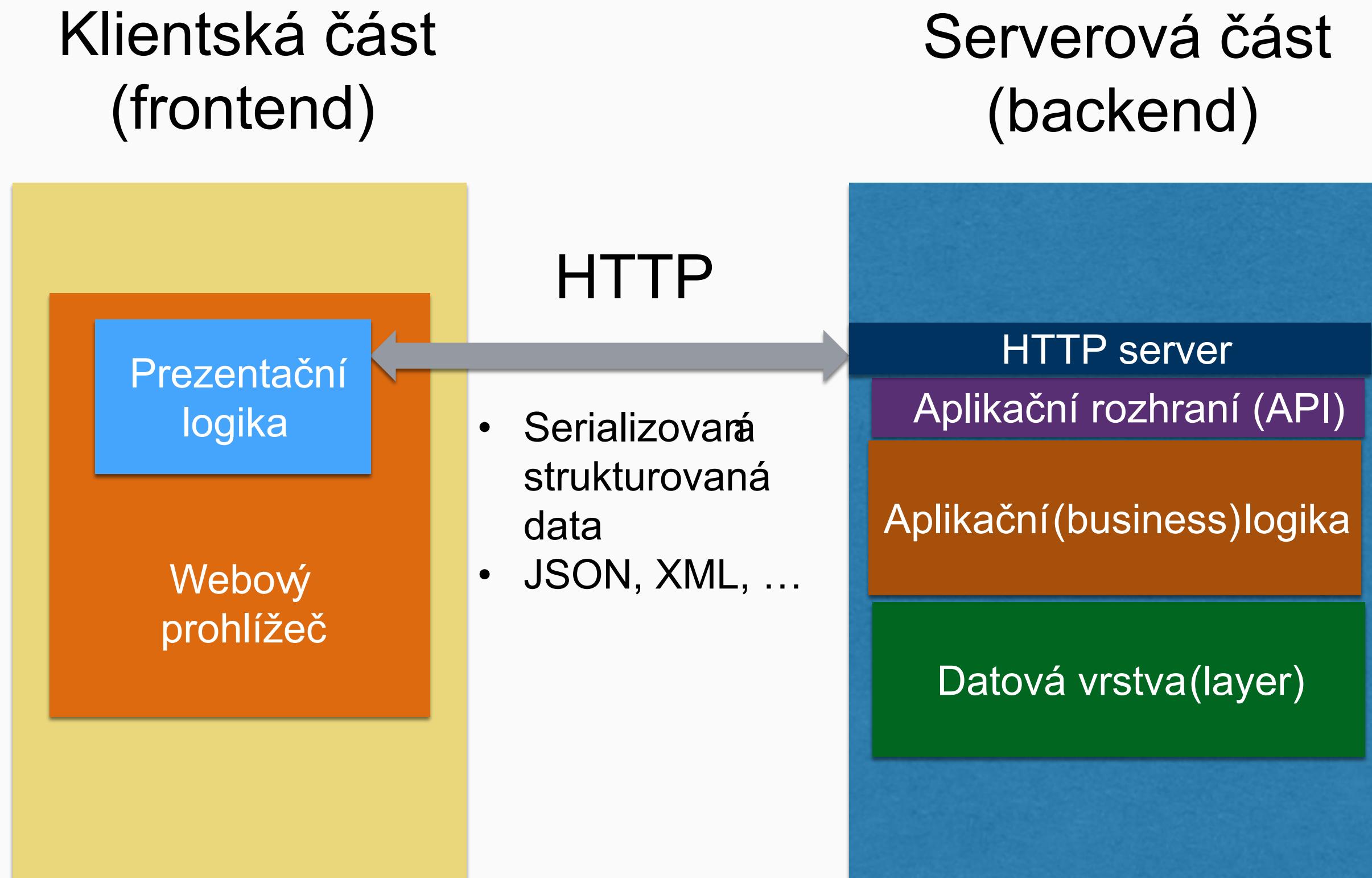
Jiné aplikace



Webový IS



Webový IS s aplikáčním rozhraním



Jakarta EE

Serverová část informačního systému (backend)

Java

- Programovací jazyk
 - Silně typovaný, objektově orientovaný
- Platforma pro vývoj a provoz aplikací
 - Virtuální stroj
 - Podpůrné nástroje
 - (překladač, debugger, dokumentace, ...)
- Balík Java SE (JRE nebo **JDK**)
 - Alternativní implementace (i open source)

Jakarta EE

- V současnosti [Jakarta EE 11](#), více dostupných implementací
 - Aka Enterprise Java (dříve Java EE)
- Platforma pro vývoj podnikových aplikací a IS v Javě
- Množina standardních technologií a API
 - Jakarta Enterprise Beans (EJB)
 - Jakarta Transactions (JTA)
 - Jakarta Persistence API (JPA)
 - Jakarta Message Service (JMS)
 - Jakarta Server Faces (JSF)
 - ... a další

Vrstvy aplikace Jakarta EE

- Databázová vrstva
 - Abstrakce nad db serverem
- Business vrstva
 - Implementace business operací (transakce)
 - Potenciálně distribuovaná
- Webová vrstva
 - Webové API nebo komponentový serverový framework

Nejdůležitější součásti specifikace

- **Datová vrstva**
 - Jakarta Persistence API (JPA)
- **Business vrstva**
 - Jakarta Transactions API (JTA)
 - Jakarta Enterprise Beans (EJB), Contexts and Dependency Injection (CDI)
 - Java Messaging Service (JMS)
- **Webová vrstva**
 - Jakarta Servlet
 - Jakarta RESTful Web Services (JAX-RS)
 - Jakarta Server Faces (JSF)

Struktura aplikace Jakarta EE

- Moduly
 - EJB moduly (*.jar)
 - Chování + veřejná rozhraní (Java)
 - Webové moduly (*.war)
 - Chování + Webové rozhraní (REST nebo Web)
 - (*web.xml* – deployment descriptor)
 - Lze je odděleně nasadit na aplikační server
- Enterprise aplikace
 - Více modulů, archiv EAR
 - *application.xml*

Běhové prostředí – Kontejnery

- Prostředí pro běh aplikace na serveru
- EJB kontejner
 - Běh EJB modulů, volání funkcí
- Webový kontejner
 - Běh webové vrstvy, HTTP server
- Java EE kontejner
 - Webový + EJB kontejner

Aplikace a knihovny

- Jakarta EE server full
 - Implementace všech standardů v runtime serveru
 - Aplikace obsahuje jen „nadstandardní“ knihovny
- Odlehčený server (*micro edition* apod.)
 - Jen část knihoven na serveru
 - Zbytek musí být přibalen k aplikaci (např. JPA)
- Pouze webový server (web container)
 - Jen knihovny pro webovou vrstvu (Servlet, JSP)

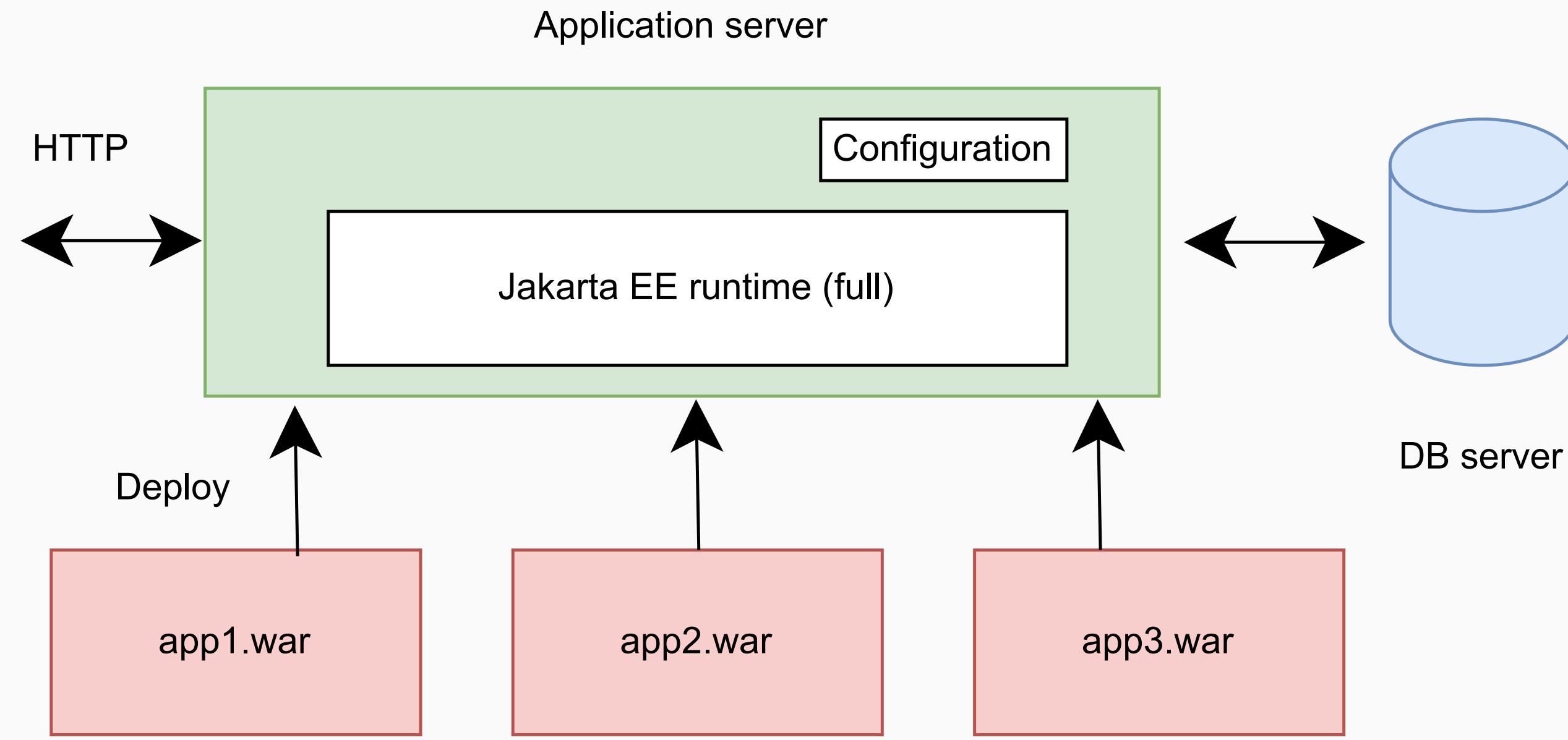
Profily

- Jakarta EE Full
 - Všechny vlastnosti (features)
- Jakarta EE Web Profile
 - Zaměření na webové aplikace (zahrnuje např. i JSF)
- Jakarta EE Core Profile
 - Zaměření na mikroslužby (minimální sada vlastností)
- Microprofile
 - Jakarta EE Core + doplňující standardy
 - Zaměření na mikroslužby (API)

Dostupné aplikační servery

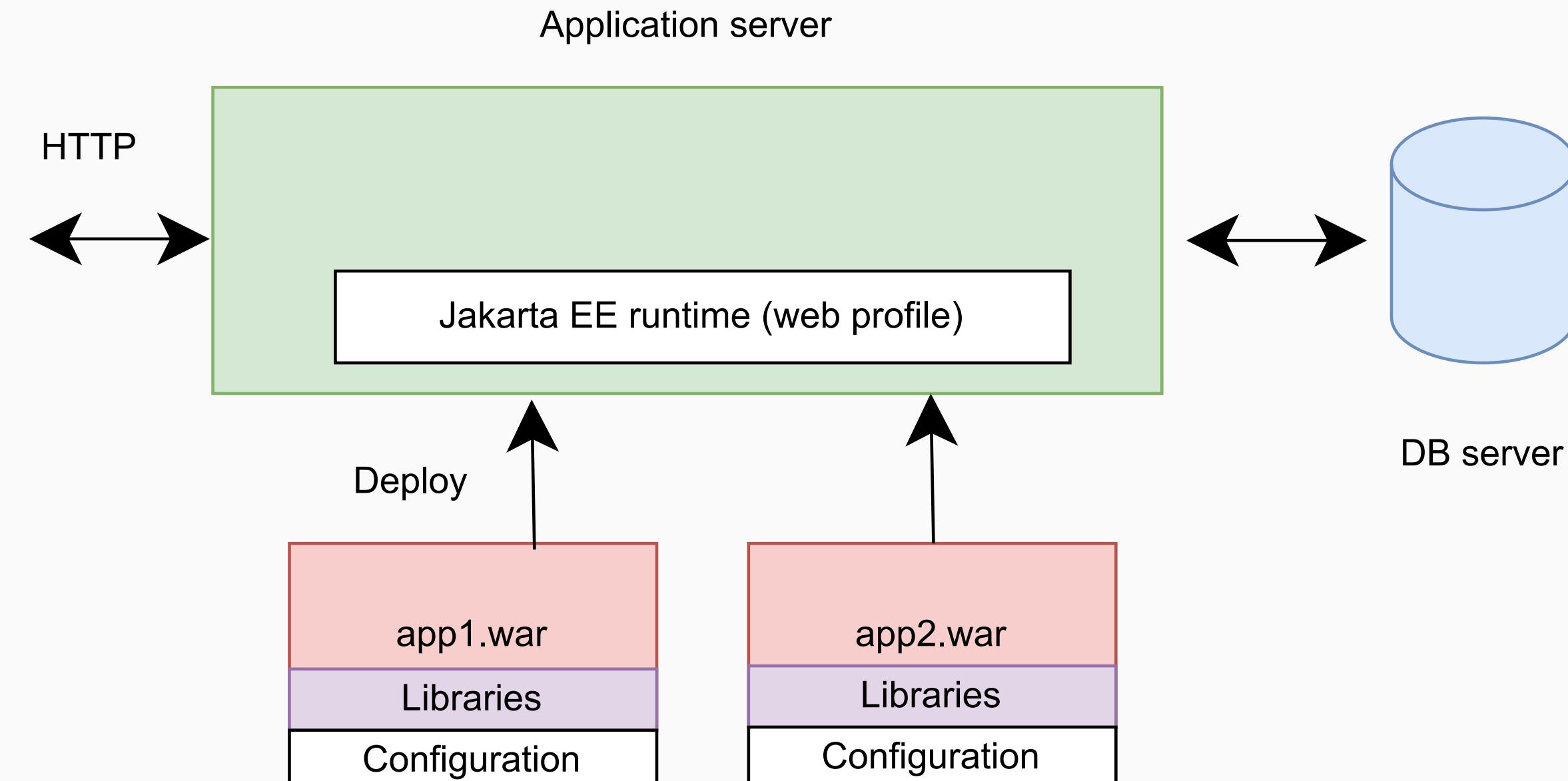
- Jakarta EE kontejnery (aplikaciční servery)
 - [Eclipse GlassFish](#)
 - [Payara](#)
 - WebSphere, [Open Liberty](#) (IBM)
 - [WildFly](#) (Red Hat, dříve JBoss AS)
 - [TomEE](#) (Apache)
- Pouze webové servery
 - Tomcat (Apache)
 - Jetty (Mort Bay Consulting)

Nasazení - plný server



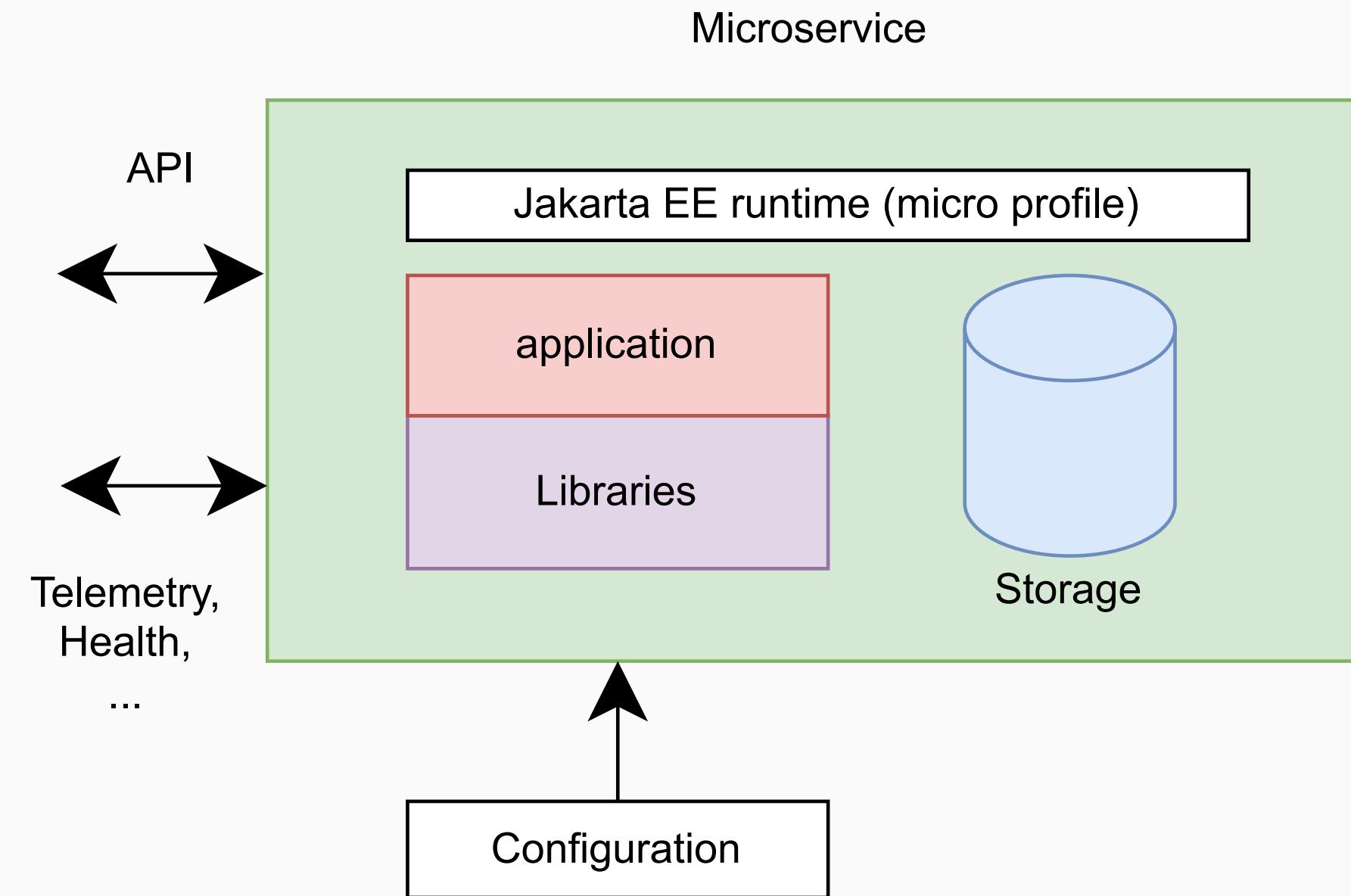
- Aplikace se nasazují za běhu serveru

Nasazení - odlehčený server



- Server spouštěný spolu s aplikacemi

Nasazení - mikroslužba



- Samostatná jednotka (vše v jednom) (embedded)

Konfigurace – Open liberty

- Plný server (více aplikací)
 - Konfigurovatelné *features* (`server.xml`)
- Lokální server specifický pro aplikaci (maven plugin)
 - [Building a web application with Maven](#)

Konfigurace – Payara

- Plný server (Payara full, Payara web profile)
- Payara micro
 - Jeden spustitelný JAR balík
 - Při spuštění se nasadí aplikace (WAR archiv)
- Payara micro maven plugin
 - Vytvoří spustitelný JAR s aplikací
 - Např. [payara-micro-javaee-crud-rest-starter-project](#)

Praktický vývoj v Jakarta EE

Instalace a první projekt

Prerekvizity

- Java 17+ (21) – **musí být JDK**
 - V Linuxu v distribuci (OpenJDK)
 - Nebo např. [Adoptium](#)
- Maven
 - V Linuxu v distribuci
 - Nebo např. [návod zde](#)
- Git (volitelně)
 - 100% doporučeno pro práci v týmu

Komponenty

- Jakarta EE server
 - **Open Liberty** openliberty.io
 - Payara community edition payara.fish
 - TomEE, WildFly, ...
- Vývojové prostředí
 - **Eclipse** for Enterprise Java Developers
 - IntelliJ IDEA, VS Code ...
- Databázový server
 - Jakýkoliv relační ([H2](https://h2database.com/), [Derby](https://derby.apache.org/), MySQL, ...)
- Databázový konektor (JDBC ovladač)

Vývojové prostředí – Eclipse

- Instalovat Eclipse
 - [Eclipse IDE for Enterprise Java and Web Developers](#)
- Instalovat nástroje pro Payara a/nebo Liberty
 - Eclipse Marketplace
- Definovat server v IDE
 - Podle způsobu spouštění serveru
- Alternativy
 - VS Code, IntelliJ IDEA, ...
 - [Liberty tools](#)

Tvorba aplikace v Jakarta EE

Maven projekt

- Použití existujícího projektu
 - Např. [Jakarta EE starter](#)
 - viz. `pom.xml`
- Generátory projektů
 - [Liberty start](#)
 - [Microprofile starter](#)
- Maven archetypes
 - [JakartaEE-essentials-archetype](#)
 - [WildFly-getting-started-archetype](#)

Vytvoření projektu v Eclipse

- Import existujícího Maven projektu [Starter project](#)
 - Např. přepnout do perspektivy Git
- Update Maven projektu (Alt-F5)
- Konfigurace později
 - Project -> Properties
 - Project Facets

Maven projekt – konfigurace

- Nastavení parametrů
 - *Project coordinates* (groupId a artifactId)
 - Packaging `war`
 - Dependencies (provided?)
- Zdrojové soubory
 - `src/main/java`
- Překlad
 - `mvn clean package`
 - Výsledný balík v `target/*.war`

Struktura WAR

- Kořenový adresář je / webu
- META-INF
 - Informace o archivu
- WEB-INF/lib
 - Potřebné knihovny (*.jar)
- WEB-INF/class
 - Přeložené třídy (*.class)

Java Servlet

- Třída implementující vyřizování HTTP požadavků
- Implementuje rozhraní `HttpServlet`
- Anotovaná pomocí `@WebServlet`
 - Mapování na URL

Webové API

REST rozhraní pomocí JAX-RS

REST

- Předpokládá CRUD (Create-Retrieve-Update-Delete) operace s *entitami*
 - Ale ve skutečnosti přistupujeme k **business vrstvě**, ne přímo k datům!
 - Tzn. **voláme aplikační logiku**
- Úzká vazba na HTTP
 - Využití HTTP metod a jejich významu
 - Využití stavových kódů v HTTP
- Nedefinuje formát přenosu dat, obvykle JSON, méně XML (často obojí)

Endpointy

- Endpoint = URL, na které lze zaslat požadavek
- Reprezentuje **zdroj** (*resource*), který má nějaký **stav** (*state*)
- Endpointy pro operace se zdroji
 - Kolekce entit, např. <http://obchod.cz/api/objednavky>
 - Jedna entita, např. <http://obchod.cz/api/objednavky/8235>
- Endpointy pro volání funkcí
 - Např. <http://obchod.cz/api/odesli-objednavku>

Metody HTTP – Operace se zdroji

- **GET**
 - Čtení stavu zdroje (read)
- **POST**
 - Přidání podřízeného zdroje (přidání do kolekce, create)
- **PUT**
 - Nahrazení zdroje novým stavem (update)
- **PATCH**
 - Nahrazení části zdroje (update)
- **DELETE**
 - Smazání zdroje (delete)

Metody HTTP – Volání funkcí

- **GET i POST**
 - Vykoná operaci vrátí výsledek (serializovaná data)
- Pokud je výsledkem operace nový zdroj, jeho URL se vrátí v hlavičce **Location**.

Stavový kód

- Stavový kód odpovědi HTTP může odpovídat výsledku operace
- Typicky například:
 - 200 Ok
 - 201 Created
 - 400 Bad request
 - 403 Forbidden
 - 404 Not found
 - 500 Internal server error

Formát přenosu dat

- Není specifikován, záleží na službě
 - Obvykle JSON nebo XML (schéma záleží na aplikaci)
- Často více formátu k dispozici
 - Využití MIME pro rozlišení typu, HTTP content negotiation
(hlavička **Accept** :)

REST a Jakarta EE

- JAX-RS API součástí standardu
- Vytvoření služeb pomocí anotací
- Aplikační server zajistí funkci endpointu (JAX-RS servlet)
 - Mapování URL a HTTP metod na Javovské objekty a metody
 - Serializace a deserializace JSON/XML na objekty
- Různé implementace
 - JAX-RS – Jersey (Glassfish), Apache Axis
 - Serializace – Jackson, gson, MOXy, ...
 - Neřešíme – je vždy součástí aplikáčního serveru

REST v Javě

```
import javax.ws.rs.GET;
import javax.ws.rs.Produces;
import javax.ws.rs.Path;

@Path("/users/{username}")
public class UserResource {

    @GET
    @Produces("text/xml")
    public User getUser(@PathParam("username") String userName) {
        // volání business operací (aplikační logika)
        return someUser;
```

- Definované typy pro přenos dat – **Data Transfer Objects (DTO)**
- Jednoduchý příklad: [ping endpoint](#)

Konfigurace

- Třída odvozená od `javax.ws.rs.core.Application`
- Konfigurace pomocí anotací
- Např. [ApplicationConfig](#)

Persistence dat

Objektové rozhraní JPA

Java Bean

```
public class Person
{
    private long id;
    private String name;
    private String surname;
    private Date born;

    public String getName() {
        return name;
    }

    public void setName(String name) {
```

Java Bean - role

- POJO – Plain old Java object
- Data Transfer Object (DTO)
 - Pro přenos (HTTP) – serializace a deserializace
- Entita (Entity)
 - Pro ukládání do databáze – persistenci

Persistence

- Pomocí anotací vytvoříme z třídy **entitu** persistence

```
@Entity  
@Table(name = "person")  
public class Person  
  
{  
    @Id  
    private long id;  
    private String name;  
    private String surname;  
    private Date born;  
  
    ...
```

- Více v další přednášce

Konfigurace persistence

- Soubor `persistence.xml`
- Jméno jednotky persistence
- Odkaz na konfigurovaný *data source*
 - JNDI name (např. `jdbc/demo`)
 - Případně další parametry pro mapování
 - Např. řízení automatického generování schématu

Databázová konektivita

- Konfigurace zdroje dat (JDBC data source)
 - Ovladač, adresa serveru, databáze, přihlašovací údaje
 - Přiřazené jméno (např. `jdbc/demo`)
- Aplikace následně využívá definovaný zdroj daného jména
 - Přenositelnost aplikaci do různých prostředí

Konfigurace – Open Liberty

- Pro „velký“ server
 - `wlp/usr/servers/<nazev>/server.xml`
 - Definice ovladačů i datového zdroje
- Pro aplikaci
 - `src/main/liberty/config/server.xml`
 - Příklad knihovny, konfigurace zdroje

Viz [Relational database connections with JDBC](#)

Konfigurace – Payara

- Pro server
 - Webové administrační rozhraní <http://localhost:4848>
 - příp. nástroje příkazové řádky
 - [Using MySQL with Payara](#)
- Pro aplikaci
 - Soubor `WEB-INF/glassfish-resources.xml`
 - Příklad: [knihovny, konfigurace zdroje](#)

Business vrstva

Implementace aplikáční logiky

Enterprise Java Beans (EJB)

- Zapouzdřují business logiku aplikace
- Poskytují business operace – definované rozhraní (metody)
- EJB kontejner zajišťuje další služby
 - Dependency injection
 - Transakční zpracování
 - Metoda obvykle tvoří transakci, není-li nastaveno jinak

Vytvoření EJB

- Instance vytváří a spravuje EJB kontejner
- Vytvoření pomocí anotace třídy
 - **@Stateless** – bezstavový bean
 - Efektivnější správa – pool objektů přidělovaných klientům
 - **@Stateful** – udržuje se stav
 - Jedna instance na klienta
 - **@Singleton**
 - Jedna instance na celou aplikaci
- Použití (získání instance)
 - Anotace **@EJB**

Contexts and Dependency Injection (CDI)

- Obecný mechanismus pro DI mimo EJB
- Omezuje závislosti mezi třídami přímo v kódu
 - Flexibilita (výměna implementace), lepší testování, ...
- Injektovatelné objekty
 - Třídy, které nejsou EJB
 - Různé vlastnosti pomocí anotací
- Použití objektu
 - Anotace `@Inject`
 - CDI kontejner zajistí získání a dodání instance

CDI – Injektovatelné objekty

- Téměř jakákoli Javaovská třída
- Scope
 - `@Dependent` – vzniká pro konkrétní případ, zaniká s vlastníkem (default)
 - `@RequestScoped` – trvá po dobu HTTP požadavku
 - `@SessionScoped` – trvá po dobu HTTP session
 - `@ApplicationScoped` – jedna instance pro aplikaci
 - *Pozor na shodu jmen se staršími anotacemi JSF*
- Pokud má být přístupný z GUI (pomocí EL)
 - Anotace `@Named`

A to je vše!
Dotazy?