



Dokumentace k semestrální práci z KIV/PIA

KIVBOOK

Student: Martin Kružej
St. číslo: A17N0079P
E-mail: kruzej@students.zcu.cz
Datum: 6. ledna 2018

Obsah

| | | |
|----------|---------------------------------|----------|
| 1 | Zadání | 1 |
| 2 | Architektura aplikace | 2 |
| 2.1 | Obecná architektura | 2 |
| 2.2 | Architektura servletů | 3 |
| 2.3 | Architektura entit | 4 |
| 2.4 | Architektura Manažerů | 5 |
| 2.5 | Architektura Dao | 6 |
| 3 | Aplikace | 7 |
| 3.1 | Use case | 7 |

1 Zadání

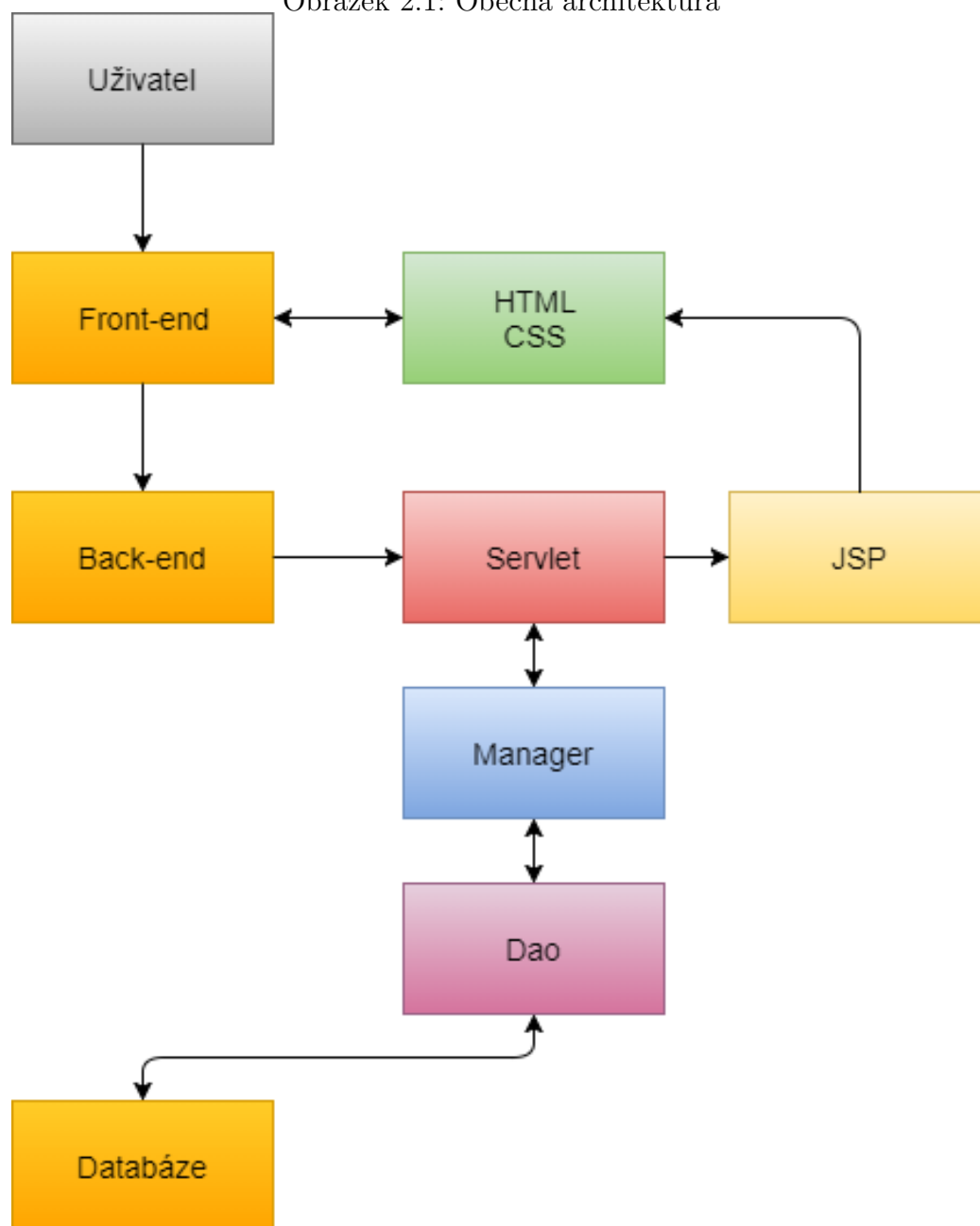
Standardní téma KIV/PIA, viz [KIV/PIA courseware](#).

2 Architektura aplikace

2.1 Obecná architektura

Aplikace je postavená na technologii Java EE. Jednoduchý diagram ukazující základní obecný chod aplikace je uveden na diagramu 2.1. Uživatel pracuje s tzv. front-endem, který je na klientovi ve formě HTML a CSS. Při zaslání požadavku je tento požadavek předán tzv. back-endu. Zde je zpracován jedním ze servletů, ti v případě potřeby mohou volat databázi přes třídy Manažerů a Dao. Poté vyberou některý z JSP souborů dle požadavku a ten vrací zpět na front-end, kde je zobrazen již ve formátu HTML a CSS. K architektuře bude řečeno více detailů v následujících kapitolách.

Obrázek 2.1: Obecná architektura



2.2 Architektura servletů

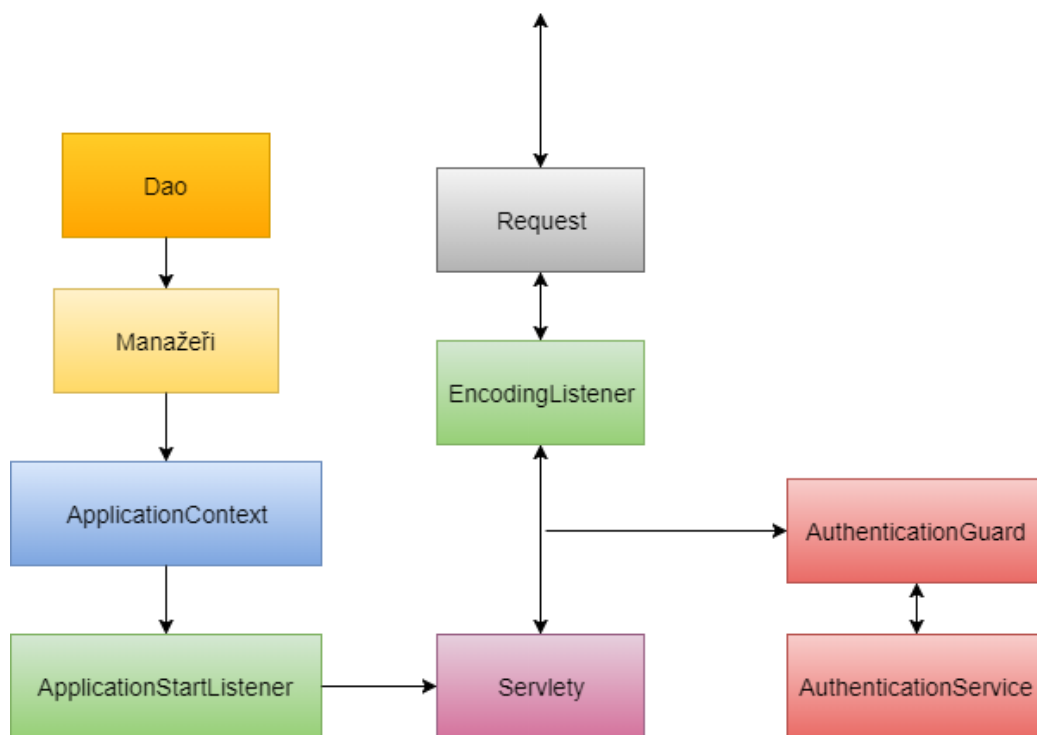
Servlety jsou základní kámen serveru - tyto servlety obsluhují veškeré požadavky, které na server chodí. Pro přístup k databázi využívají různé

manažery a tyto manažery jim musí někdo poskytnout. Máme zde třídu `ApplicationContext`, která vytvoří všechny potřebné třídy Manažerů a Dao. Při inicializaci je spuštěn `StartupListener`, který třídy z `ApplicationContext` injekčně dopraví do servletů, které to potřebují. Tento styl se často označuje jako *Dependency injection*.

Při vyslání požadavku na server se tento požadavek nedostane k servletům hned. Nejdříve projde `EncodingListener`, jenž zajišťuje kódování charakterů ve formátu UTF-8. Dále jsou určité servlety za jakousi bránou, která je brání před neautorizovaným přístupem. Tato brána je `AuthenticationGuard` využívající `AuthenticationService`. Tyto dvě třídy pouze kontrolují, že pro přístup k některým servletům je třeba být přihlášen.

Diagram znázorňující tuto architekturu je uveden na obrázku 2.2.

Obrázek 2.2: Architektura servletů



2.3 Architektura entit

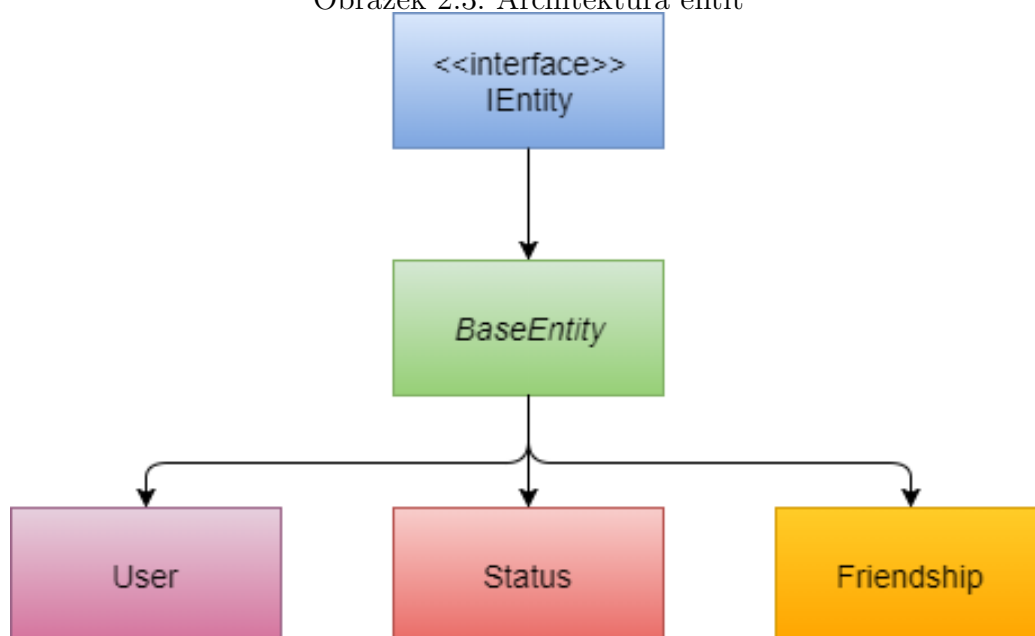
Před podíváním o Manažerech je třeba se podívat na nejnižší úroveň této architektury a to jsou entity. Tyto entity jsou v podstatě tabulky, které se v databázi používají. Nejdříve je zde rozhraní `IEntity`, které dědí abstraktní

třída BaseEntity. Tyto dvě třídy poskytují jednotné *id* pro všechny entity. Ty existují následující:

- **User** - uživatel v databázi
- **Status** - status/tweet v databázi
- **Friendship** - přátelství / požadavek na přátelství v databázi

Následuje dědění abstraktní třídy entitami. Tento vztah je vyobrazen na obrázku 2.3.

Obrázek 2.3: Architektura entit

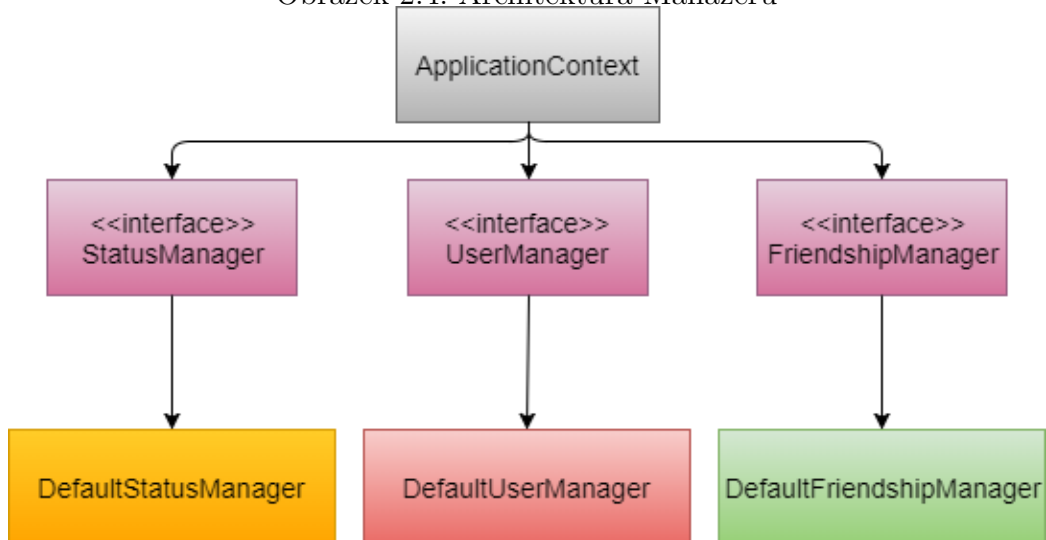


2.4 Architektura Manažerů

Manažeři mají za úkol poskytnout prostředníka mezi servlety a databází, potažmo Dao. Poksyťují různé metody pro práci s databází či entitami. Opět se zde setkáváme s rozhraními, ze kterých jsou potom odvozeny samotné třídy. Pro každou jednotlivou entitu existuje jednotliví Manažer. Servlety obecně pracují právě s rozhráním, které je jim injektováno a již je jim jedno, jakou implementaci Manažerů používají.

Vyobrazeno na obrázku 2.4.

Obrázek 2.4: Architektura Manažerů



2.5 Architektura Dao

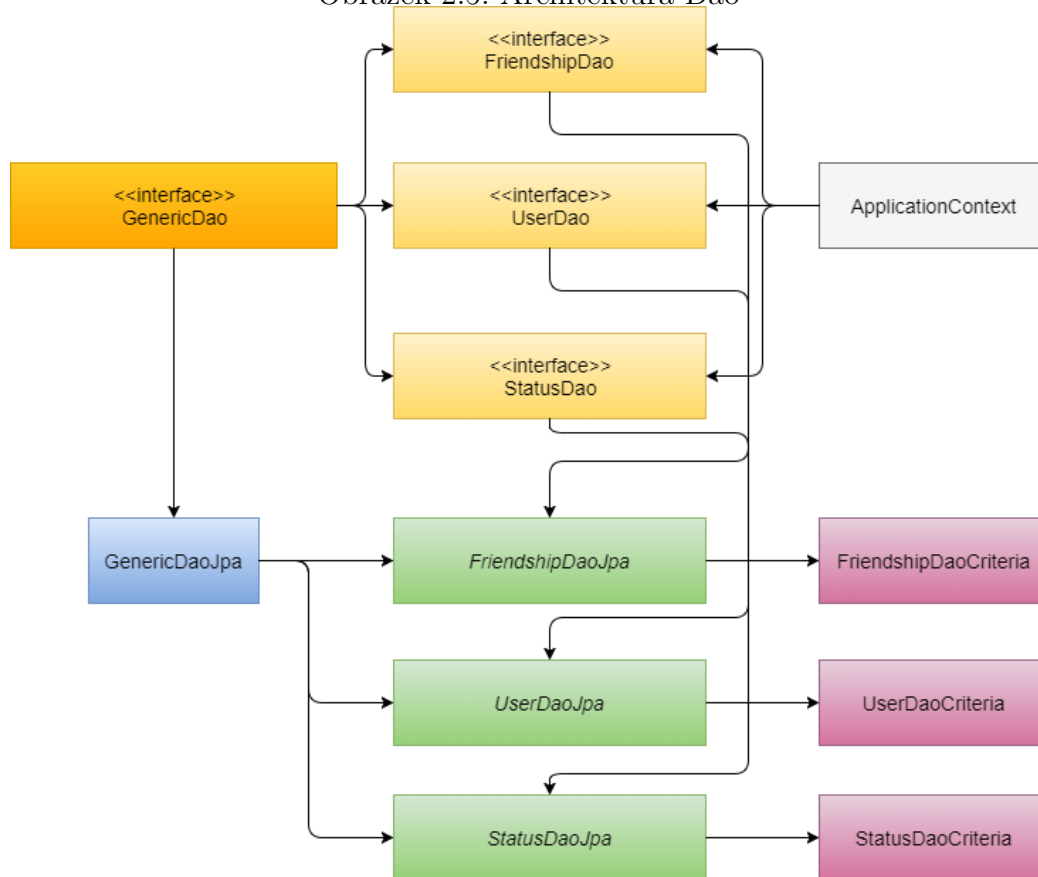
Tato úroveň pracuje s databází. Přes rozhraní slibuje různé metody pro tuto práci a poté je i implementuje. Vzhledem k tomu, jak je tato architektura navržena, je možno udělat několik různých implementací těchto Dao tříd. V aplikaci je použito Criteria, ale je možno udělat třeba JPQL pouhým přidáním tříd a změnou injektování.

Základní rozhraní je GenericDao, kde se definují ty nejzákladnější metody, které každé Dao musí implementovat - jako najít instanci, uložit instanci, smazat instanci. Toto rozhraní dědí rozhraní deklarující již specifické metody pro různé entity - FriendshipDao, UserDao a StatusDao. Tyto třídy jsou to rozhraní, které je injektováno třídou ApplicationContext.

Následuje specifikace JPA. Nejdříve obecná třída GenericDaoJpa s velice obecnými metodami. Následují abstraktní třídy pro jednotlivé entity. Tyto třídy dědí jak GenericDaoJpa tak jednotlivé Dao rozhraní. Poslední částí architektury jsou samotné implementace jednotlivých metod. Jak bylo řečeno v předešlých odstavcích, tato aplikace využívá Criteria, ale zde by mohly být další třídy, např. FriendshipDaoJPQL využívající JPQL.

Vyobrazeno na obrázku 3.1.

Obrázek 2.5: Architektura Dao



3 Aplikace

3.1 Use case