Pokročilé informační systémy

Jakarta EE – Business a prezentační vrstva

Ing. Radek Burget, Ph.D.
burgetr@fit.vutbr.cz

Implementace business operací

- 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 EJB

- Lokální
  - Anotace `@EJB` – kontejner dodá instanci EJB
- Vzdálené volání – dané rozhraní
  - Rozhraní definované pomocí @Remote

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 získáni a dodání instance

CDI – Injektovatelné objekty

- Téměř jakákoliv Javovská 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

CDI – Dodání instancí

- Anotace @Inject
- Vlastnost (field)

```
@WebServlet(urlPatterns = "/itemServlet")
public class ItemServlet extends HttpServlet {

    @Inject
    private NumberGenerator numberGenerator;

}
```

CDI – Dodání instancí

- Konstruktor

```
@WebServlet(urlPatterns = "/itemServlet")
public class ItemServlet extends HttpServlet {

    private NumberGenerator numberGenerator;

    @Inject
    public ItemServlet(NumberGenerator numberGenerator) {
        this.numberGenerator = numberGenerator;
    }
    ...
}
```

CDI – Dodání instancí

- Setter

```
@WebServlet(urlPatterns = "/itemServlet")
public class ItemServlet extends HttpServlet {

    private NumberGenerator numberGenerator;

    @Inject
    public void setNumberGenerator(NumberGenerator numberGenerator)
    {
        this.numberGenerator = numberGenerator;
    }

    ...
}
```

Webové API – REST

REST

- Representational state transfer
- Jednoduchá metoda vzdálené manipulace s daty
- Reprezentuje CRUD (Create-Retrieve-Update-Delete) operace
  - Ale ve skutečnosti přistupujeme k **business vrstvě, ne přímo k datům**!
- Úzká vazba na HTTP
- Nedefinuje formát přenosu dat, obvykle JSON nebo XML

Využití HTTP

- Každá položka dat (nebo kolekce) má vlastní URI. Např.:
  - http://noviny.cz/clanky (kolekce)
  - http://noviny.cz/clanky/domaci/12 (jeden článek)
- Jednotlivé metody HTTP implementují příslušné operace s daty

Metody HTTP

- GET
  - Kolekce: získání seznamu položek
  - Entita: čtení entity (read)
- POST
  - Kolekce: přidání prvku do kolekce (create)
- PUT
  - Kolekce: nahrazení celého obsahu kolekce
  - Entita: zápis konkrétní entity (update)
- DELETE
  - Kolekce: smazání celé kolekce
  - Entita: smazání entity

Formát přenosu dat

- Není specifikován, záleží na službě
  - Obvykle JSON nebo XML (schéma záleží na aplikaci)
- Často více formátu k dispozici
  - Např.
    http://noviny.cz/clanky.xml
    http://noviny.cz/clanky.json
  - Využití MIME pro rozlišení typu, HTTP content negotiation

REST a Jakarta EE

- JAX-RS API součástí standardu
- Vytvoření služeb pomocí anotací
- Aplikační server zajistí funkci endpointu
  - 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, …

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 String getUser(@PathParam("username") String
                userName) {
        ...
     return someXmlString;
    }
}
```

Konfigurace

- JAX-RS servlet ve `web.xml`
  - Specifikace balíku s REST třídami nebo přímo výčet tříd
  - Specifikace JSON provider (Jackson)
- Alternativně
  - Třída odvozená od `javax.ws.rs.core.Application`
  - Konfigurace pomocí anotací

Klientská aplikace

- Zasílání REST požadavků
  - Jednotlivé JS frameworky mají vlastní infrastrukturu
- Prezentační logika
  - Navigace
  - Přechody mezi stránkami
  - Výpisy chyb, apod.

Autentizace v REST

- Protokol REST je definován jako ***bezstavový***
  - Požadavek musí obsahovat vše, žádné ukládání stavu na serveru
- To teoreticky vylučuje možnost použití sessions pro autentizaci
  - Technicky to ale možné je
  - Problém např. pro mobilní klienty
- Alternativy pro autentizaci:
  - HTTP Basic autentizace (nutné HTTPS)
  - Použití tokenu validovatelného na serveru – např. JWT
  - Složitější mechanismus, např. OAuth

HTTP Basic

- Standardní mechanismus HTTP, využívá speciální hlavičky
- Požadavek musí obsahovat hlavičku **Authorization**
  - Obsahuje jméno a heslo; nešifrované pouze kódované (base64)
    - **Je nutné použít HTTPS**
- Pro nesprávnou nebo chybějící autentizaci server vrací **401 Auhtorization Required**
  - V hlavičce **WWW-Autenticate** je identifikace oblasti přihlášení
  - Klient tedy zjistí, že je nutná autentizace pro tuto oblast

JSON Web Token (JWT)

- Řetězec složený ze 3 částí
  1. Header (hlavička) – účel, použité algoritmy (JSON)
  2. Payload (obsah) – JSON data obsahující id uživatele, jehopráva, expiraci apod.
  3. Signature (podpis) – pro ověření, že token nebyl podvržen nebo změněn cestou
- Tyto tři části se kódují (base64) a spojí do jednoho řetězce
  - **xxxxx.yyyyy.zzzzz**

Použití JWT

- Klient kontaktuje ***autentizační server*** a dodá autentizační údaje
  - Stejný server, jaký poskytuje API, nebo i úplně jiný (např. Twitter)
- Autentizační server vygeneruje podepsaný JWT a vrátí klientovi
- Klient předá JWT při každém volání API
  - Nejčastěji opět v hlavičce:
    Authorization: Bearer xxxxx.yyyyy.zzzzz
  - API ověří platnost, role uživatele může být přímo v JWT

JWT v Javě

- Součástí standardu Microprofile
  - Ne přímo součást Jakarta EE
  - Ale dostupná na běžných serverech (Payara)
- Lze snadno spojit s JAX-RS
- https://github.com/javaee-samples/microprofile1.4-samples/tree/master/jwt-auth

Serverová prezentační vrstva

Java Server Faces (JSF)

Prezentační vrstva

- Webové API + JS tlustý klient
  - Prezentační logika na klientovi
  - Business operace přístupné pomocí API (REST, SOAP, …)
  - Klientská aplikace může využívat klientský framework
    - Angular, React.js, Vue.js, …
- Serverový framework
  - Prezentační logika na serveru
  - Server generuje HTML (CSS, JavaScriptový) kód
  - Komponentově orientované frameworky

Prezentační vrstva na serveru

- Funkčnost zajišťuje webový kontejner
- Pro Java EE standardně 2 vrstvy
  1. Facelets
  2. Java Server Faces

- Existují další webové frameworky (Struts, Spring, Vaadin, …)

Java Servlet

- Třída implementující chování serveru
- Obecný **`javax.servlet.GenericServlet`**
  - Metody init(), destroy(), service()
- HTTP **`javax.servlet.http.HttpServlet`**
  - Metody doGet(), doPost(), …
- Vytváření instancí řídí server
- Jedna instance může obsluhovat více požadavků

Java Server Pages

- Dynamické webové stránky
  - HTML (XHTML, XML, …) kód
  - Vložený kód v Javě (*scriptlet*)
  - Definované značky
    - Výrazy – unified expression language (EL)
- Umožňuje definovat **knihovny značek** (tag libraries)
  - Chování každé značky implementováno jako třída
  - XML deskriptor knihovny

Příklad JSP

```
<%@ page language="java„
    contentType="text/html; charset=ISO-8859-2„
    pageEncoding="ISO-8859-2"%>
<!DOCTYPE html>
<html>
<head>
  <title>My JSP Page</title>
</head>
<body>
<h1>Hello World!</h1>
<p>
<%
  out.println("Current time: " + new java.util.Date());
 %>
</p>
</body>
</html>
```

Zpracování JSP stránky

- Stránka se překládá na servlet (třídu)
- Překlad zajišťuje kontejner
  - Skripty – vloženy do stránky
  - Tagy – volání metod příslušných tříd
  - Výrazy – volání evaluátoru

Facelets

- Nástupce JSP od Java EE 6
- Založeno na XML
  - Obvykle XHTML
- Podpora existujících i nových tag libraries
- Šablony komponent a stránek

Facelets

- Překlad XHTML stránek na interní objektovou reprezentaci – **Facelet**
  - Zpracování šablon
  - Související soubory (resources)
- Možnost definice knihoven značek
  - Vystačíme s existujícími
- Zpracování výrazů jazyka EL
  - Přístup k objektům, jejich vlastnostem a metodám

Odbočka: XML namespaces

- Motivace: možnost kombinovat v jednom dokumentu značky z více XML jazyků (např. (X)HTML + SVG)
- Namespace
  - Jmenný prostor, ve kterém jsou jména značek (a atributů) unikátní
  - Je identifikován jednoznačným identifikátorem – URI
  - V rámci jednoho dokumentu má přiřazen zkrácený identifikátor – *prefix*
- Použití v XML
  - Definujeme prefix pomocí zabudovaného atributu `xmlns`
  - Značky z daného jazyka zapisujeme `<prefix:jméno>…</prefix:jméno>`
  - Jeden jmenný prostor je výchozí – bez prefixu

Příklad XML namespaces

```
<html xmlns="http://www.w3.org/1999/xhtml"
   xmlns:ui="http://java.sun.com/jsf/facelets">


<ui:composition template="template.xhtml">
   <ui:define name="content">
      <h2>Welcome</h2>
      <p><a href="person_list.xhtml">Simple
                forms demo</a></p>
   </ui:define>
</ui:composition>
```

Java – Namespaces

- Facelets tags

```
xmlns:ui="http://xmlns.jcp.org/jsf/facelets"
```

- JSF Core – obecné definice

```
xmlns:f="http://xmlns.jcp.org/jsf/html"
```

- JSF HTML – obsah stránky

```
xmlns:h="http://xmlns.jcp.org/jsf/core"
```

Šablony

- Šablona definuje vzorovou stránku
  - Proměnné části jsou označeny pomocí <ui:insert name=„“>
- Jednotlivé stránky využívají šablonu
  - Místo <body> se použije <ui:composition> a obsah proměnných částí se definuje pomocí <ui:define>.

Java Server Faces

- MVC framework nad Facelets
- Značky pro generování základních prvků uživatelského rozhraní
  - Rozšiřitelné knihovny komponent
- Řídicí servlet implementující chování

Nadstavbové knihovny

- PrimeFaces
- Apache MyFaces
- OmniFaces
- RichFaces (JBoss, není dále vyvíjen)
- ...

Aplikace nad JSF

- Založeno na vzoru Model-View-Controller
- Model – Business vrstva (služby)
- View – JSF kód
  - XHTML kód využívající komponenty
  - Managed beans
- Controller – FacesServlet, konfigurovatelný

Experssion language

- Podobný JavaScriptu
- Zápis v `#{...}` nebo `${...}`
  (zpožděné vs. okamžité vyhodnocení)
- Abstrakce sdílených objektů v libovolném kontextu

Sdílené objekty a EL

- `#{objekt}` – vyhledá objekt daného jména v kontextu stránky, požadavku, session a aplikace
- Metody: `#{customer.sayHello}`
- Vlastnosti: mapují se na set / get
  - #{customer.name}

Logika uživatelského rozhraní

- EJB služby – model
- Managed beans
  - Spravované objekty přístupné přes EL
  - Implementují chování
- Validators
  - Kontrola vstupních dat
- Converters
  - Převod dat na řetězec a zpět

Managed beans

- CDI beans
- Vlastnosti přístupné pomocí get() a set()
- Vlastnosti vázané na prvky GUI
- Scope:
  - none – vždy nový objekt je vytvořen na požádání
  - request – pro každý požadavek bude vytvořena nová instance
  - session – zachován po celou dobu sezení
  - application – po dobu běhu kontejneru

Validátory

- Položky povinné a nepovinné
  - Atribut required
- Zabudované
  - f:validateDoubleRange
  - f:validateLongRange
  - f:validateLength
- Uživatelské validátory
  - f:validator

Konvertory

- Zabudované konvertory
  - <f:convertDateTime>
  - <f:convertNumber>
- Obecný konvertor podle identifikátoru
  - <f:converter>
- Zabudované konvertory javax.faces.*
  - Boolean, Byte, Character, DateTime, Double, Long, …
- Uživatelské konvertory

Definice chování

- Každá stránka produkuje výsledek ve formě řetězce (outcome)
  - Statický (zadaný konstantou)
  - Dynamický (generovaný metodou)
    - Často po provedení nějaké akce
- Přechody mezi stránkami externě v XML souboru `faces-config.xml`

Autentizace

- Informace o přihlášení reprezentované session beanem
- Filtrování požadavků filtrem definovaným ve WEB-INF/web.xml
  - Alternativně pouze označeným anotací @WebFilter

Alternativa: Vaadin

- Komponentový server-side framework
- Veškerá specifikace GUI v Javě
  - Využívá existujících komponent
- Servlet zajišťující komunikaci s prohlížečem
  - Neřeší programátor
- https://github.com/vaadin/framework
- https://vaadin.com/framework

Otázky?