Grafická uživatelská rozhraní v Javě
Příklady k přednáškám z GJA

Tomáš Ambrož, Tomáš Dvořák, Zdenko Hornáček, Juraj Joščák

1 Servlet

V rámci tohto projektu bolo vytvorených niekoľko príkladov servletu. Pre prvotné zoznámenie sa s princípmi servletov slúži základný príklad `HelloWorldServlet`. V každom servlete musí byť prepísaná aspoň jedna z HTTP metód obsluhujúcich: DELETE, GET, HEAD, OPTIONS, POST, PUT alebo TRACE. V tomto príklade sú pre ilustráciu prepísané metódy doGet a doPost, ktoré obsluhujú HTTP GET, resp. HTTP POST metódy. Tento príklad je možné spustiť z index.html vyplnením jednoduchého formulára.

Druhým príkladom servletu je `ErrorHandler`, ktorý slúži, ako názov napovedá, k obsluhe errorov. Opäť sú prepísané metódy doGet a doPost a ich spoločná implementácia ilustruje ako sa dajú spracovať errorové stavy. Vo web.xml je podrobne zdokumentované, ako sa definuje servlet pre jednotlivé errorové stavy.

Tretí komplexnejší príklad ukazuje ako funguje nahrávanie súboru na server v servletoch. V implementácii `UploadServlet` triedy je podrobne vysvetlené ako sa spracovávajú súbory, aké informácie o týchto súboroch sú dostupné a v neposlednom rade jet u uvedené ako uložiť súbory na server. Táto možnosť nie je v servlete priamo implementovaná, ale v komentároch je presne uvedený postup ako toto chovanie docieliť.

V rámci príkladov pre servlet je implementovaný i príklad pre filter – `LogFilter`. Tento príklad dokumentuje základnú stavbu filtrov v rámci servletov. Definovanie tohto filtra, jeho inicializačných parametrov a mapovanie je popísané vo web.xml.

2 JSP

Pre ukázanie špecifikácie JSP je v projekte niekoľko príkladov. Bolo našou snahou v týchto príkladoch ukázať fungovanie JSP. Príklady dokumentujú základnú prácu s *request, response, session* a ostatnými *implicitnými* objektami, spolupráca s Java Beans, ďalej ako fungujú *direktívy, expression language (EL)* a v neposlednom rade ako sa pracuje s errormi. Všetky príklady sú podrobne vysvetlené v komentároch.

3 PrimeFaces

Knižnica *PrimeFaces* je veľmi rozsiahla a tomu preto je i počet príkladov k tejto knižnici tak veľký. Všetky príklady sú pre lepší prehlad prístupné z úvodnej stránky projektu *primefaces*. Ku každému príkladu prislúcha jedna `@ManagedBean` trieda, čo zaručuje lepšiu čitatelnosť kódu. Príklady obsahujú hlavne najčastejšie používané komponenty PrimeFaces. Konkrétne ide o inputy a selecty, s tým sú spojené eventy a listenery, autocomplete, validácia, správy a dataTable. Ďalšiou skupinou príkladov sú panely, panelGridy, dashboardy, layouty, separatory. Do tretej skupiny patrí toolbar, contextMenu, menu, menuBar, dialog, confirmDialog. Na koniec sú uvedené príklady ako counter, fileUpload či remoteCommand. Všetky tieto príklady sú pripravené tak, aby pokryli čo najviac možností využitia danej

komponenty - ukazujú danú komponentu s použitím rôznych hodnôt jej atribútov. Všetky príklady sú podrobne popísané v komentároch.

4 Spring

Za účelem co nejlepšího pokrytí přednášek byla k tomuto tématu sestavena řada příkladů demonstrující činnost jednotlivých komponent, nastavení a možností frameworku Spring. Z důvodu povahy jednotlivých částí byly příklady rozděleny do 2 projektů, oba pro vývojové prostředí Eclipse (http://www.eclipse.org).

V první části jsou rozebrány, a do jednotlivých balíků rozděleny, jednotlivé základní konstrukce, které framework nabízí. Pro zjednodušeni byly dodány i všechny potřebné knihovny v adresáři `libs`. Každá ukázka je doprovázena stručným komentářem k funkčnosti jednotlivých částí kódu, vstupních dat a očekávaných výstupních dat. Každý ukázkový příklad je definován se vstupní metodou Javy, takže spuštění vybraného příkladu spočívá pouze ve zvolení správného vstupního souboru.

Druhá část obsahuje příklad MVC projektu za využití Spring frameworku. Jeho spuštění vyžaduje nějaký aplikační server (projekt byl vyvíjen na aplikační server Tomcat verze 6.0). Obsahuje pouze statickou stránku (`index.jsp`), pro ověření funkčnosti daného serveru s odkazem na stránku generovanou pomocí třídy `ExampleController` z balíku `cz.vutbr.fit.gja.controllers` (stránka s názvem `spring.jsp`). Hlavní konfigurační soubor webového projektu (`web.xml`) v sobě obsahuje definici servletu `springmvc`, který je jediným servletem v projektu. Ten má za úkol najít a zpracovat komponenty, kterým pak v případě potřeby předává řízení. Projekt využívá maven pro vyřizování závislostí a případně i sestavování distribučních balíčků.

5 Google Web Toolkit

Jako příklad pro toto téma byla vytvořena IP kalkulačka. IP kalkulačka má na základě IP adresy v daném rozsahu a zadané masky sítě vypočítat adresu sítě a adresu broadcastu. V příkladu je aplikace rozdělena na serverovou část, která se stará o samotný výpočet a klientskou část, která má na starost zadávání potřebných údajů pro výpočet včetně kontroly vstupních údajů a zobrazení výsledků získaných od serverové části. Serverovou část obstarává třída `CalculatorServiceImpl`, která implementuje rozhraní `CalculatorService`. Rozhraní má pouze metodu `getResult()`. V této metodě třída `CalculatorServiceImpl` dostane pomocí parametrů IP adresu a masku sítě, které jsou předávány jako textový řetězec. V metodě se nejprve adresa a maska sítě rozdělí na jednotlivé oktety. Ty se následně převádějí na datový typ integer. Nyní může proběhnout samotný výpočet, kdy se adresa sítě vypočítá jako logický součin adresy a masky sítě. Adresa broadcastu se spočítá jako logický součet adresy a znegované masky sítě. Výpočet probíhá po jednotlivých oktetech. Výsledek výpočtu se následně uloží do třídy `Result`, která slouží pro přenos dat od serveru na klienta.

V klientské části se požadavek na výpočet na server odesílá až potom co proběhne kontrola vstupních údajů. Pokud vstupní údaje obsahují chybu, tak se zobrazí dialogové okno s chybovou hláškou. Pokud žádná chyba nenastane, pošle se požadavek na server. Až server zašle odpověď, tak se výsledky vypíší do určených polí.

Dále pro tento příklad byly vytvořeny unit testy pro kontrolu výpočtů adres a za účelem testování byl také vytvořen projekt se *SeleniumIPCalculator*.

6 OS Android

Toto téma pokrývá příklad, ve kterém je ukázáno, jak získat polohu zařízení, informace o zařízení, jako jsou například stav baterie, výrobce zařízení. Dále se v příkladu ukazuje jak použití UI prvků tlačítka, progressbaru a textových popisků.

Příklad byl vyvíjen pod vývojovým prostřením Android Studio. Toto vývojové prostředí je založené na vývojovém prostředí IntelliJ a podporuje operační systémy Windows, Linux, Mac OS. Prostření je dostupné na https://developer.android.com/studio. Součástí instalace Android Studio je i instalace SDK Tools pro android ,bez které bychom aplikaci nezkompilovali a nebyli bychom schopni ji nahrát do mobilního zařízení. Testovat aplikaci je možno buď na reálném zařízení, na kterém je povolené „Ladění USB“ a zařízení musí být samozřejmě připojené k počítači USB kabelem. Druhou možností je ladit aplikaci pomocí virtualizace zařízení. Při vytváření virtuálního zařízení je možné zvolit velikost a rozlišení displeje. Dalším parametrem, který je při vytváření třeba zadat, je na jaké verzi operačního systému Android virtuální zařízení poběží. Pro ladění bylo použito jak reálné zařízení s verzí Android 7.0, tak i virtuální zařízení s totožnou verzí operačního systému Android.

Aplikace v příkladu se skládá ze dvou oken Status a LocalizacionActivity. Okno Status je hlavním oknem, což znamená, že se spustí hned po spuštění aplikace. Na okno LocalizationActivity pak lze přejít kliknutím na tlačítko „ZOBRAZIT POLOHU“ v hlavním okně. Nyní si popíšeme, co jednotlivá okna obsahují. V hlavním okně (Status) se zobrazuje název zařízení, který se skládá z názvu výrobce zařízení a názvu modelu zařízení. Tato informace se vykresluje pomocí dvou UI elementů, a to popisku pro nadpis a popisku pro výpis informace o názvu zařízení. Další informací, která se zobrazuje v hlavním okně je aktuální kapacita baterie. Zobrazení této informace používá progressbar a popisek pro zobrazení kapacity v procentech. Posledním prvkem v hlavním okně je tlačítko „Zobrazit polohu“. Po kliknutí na toto tlačítko se přejde na druhé okno (LocationActivity). Toto okno zobrazuje GPS souřadnice s aktuální polohou ve stupních a adresu aktuální pozice. Nyní bude popsáno, jak je tento příklad implementován. Důležité soubory pro implementaci:

- `/app/java/cz.vutbr.fit.gja.AndroidStatus/Status.java`
  - logika hlavního okna
- `/app/java/cz.vutbr.fit.gja.AndroidStatus/LocationActicity.java`
  - logika okna s aktuální polohou
- `/app/manifests/AndroidManifest.java`
- `/app/res/layout/activity_status.xml`
  - rozmístění UI prvků hlavního okna
- `/app/res/layout/activity_location.xml`
  - rozmístění UI prvků okna pro aktuální polohu

V logice hlavního okna bylo třeba řešit implementaci události kliknutí na tlačítko. To se realizuje pomocí metody `onClickBtn()`. V metodě `onCreate()`, která se provádí po spuštění okna, se nastaví název zařízení a zaregistruje receiver pro zjišťování aktuálních informací o kapacitě baterie. K tomu abychom mohly registrovat receiver musíme mít vytvořenou instanci objektu `BroadcastReceiver`. U instance receiveru byla překryta metoda `onReceive()`, ve které nastavujeme hodnoty aktuálního stavu kapacity baterie jednotlivým UI prvkům. Logika okna pro výpis aktuální polohy je řešena třídou `LocationActivity`. Aby tato třída mohla přistupovat a sledovat aktuální polohu musí implementovat rozhraní `LocationListener`. Zde stojí za zmínku metoda `onLocationChanged()`, kde se nastavují hodnoty zeměpisné šířky a délky. A z těchto údajů se zjišťuje adresa aktuální polohy pomocí třídy `Geocoder`.