Dceřiný element	Povinný	Popis
		Nejběžnější použití tohoto elementu je pro specifikaci souboru MP3 v podcastu.
guid	Ne	Unikátní identifikátor položky.
pubDate	Ne	Datum publikace položky.
source	Ne	Zdroj položky pocházející od třetí strany. Používá se při citacích položek.

Formát Atom

Existuje nový, třetí, formát pro publikování kanálů s názvem Atom. Tento formát se pokouší spojit rozšiřitelnost a jednoduchost nabízené jednotlivými verzemi RSS. Z pohledu struktury používá formát Atom pro kanály podobný model jako RSS – metadata následovaná záznamy. Názvy elementů jsou však trochu odlišné. V tomto projektu nebudeme formát Atom používat, ale je dobré o něm vědět. Ačkoli se jedná o nejméně vyzrálý formát a jeho podíl na trhu je relativně malý, jedná se o jediný formát podporovaný hlavními standardizačními komunitami (obzvláště komunitou Internet Engineering Task Force) a již ho používají takové veličiny jako je Google News.

Seznámení s YouTube

Dnes již stránku YouTube asi není třeba nikomu představovat. Tato stránka je stejným velikánem jako ostatní dříve diskutované stránky – Amazon, Google, Yahoo! a Microsoft Live Search. Odkazy na videa uložená na tomto serveru se často předávají pomocí e-mailů. Pro případ, že nejste příliš informováni, v rychlosti si stránku YouTube představíme, ukážeme si některé dostupné funkce a webové rozhraní API.

V kostce je YouTube stránka umožňující uživatelům sdílet svá domácí videa s veřejností pomocí Internetu. Uživatelé mohou uploadovat jakékoli video chtějí (s ohledem na dodržování autorských práv) a ostatní uživatelé si ho mohou přehrát a okomentovat. Druhá z uvedených funkcí umožnila vedle sdílení videí také vznik silné sociální infrastruktury.

Některé z dostupných funkcí stránky YouTube jsou tyto:

- **Značkování videí** – YouTube spoléhá na uživatelskou komunitu, aby správně označila videa v repozitáři. Za tímto účelem mají uživatelé právo připojovat popisná slova či krátké fráze ke konkrétním videím. Tento proces je známý jako **značkování** (tagging). Např. pokud sleduji klip kapely Kabát s názvem Šaman, umožní mi YouTube označkovat klip značkami jako je Kabat, Saman, Rock, Czech atd.
- **Vyhledávání videí** – YouTube disponuje robustním vyhledávačem, který pracuje na základě značek videí. Díky vyhledávání podle značek se označení videí stává demokratickou záležitostí. Pokud by YouTube vyhledával pouze na základě označení, které poskytl uživatel, jenž video poskytl, měla by tato osoba veliký

vliv na to, jestli se video ve výsledcích hledání objeví nebo ne. Tím, že se prověří také značky videa, najde vyhledávač i videa, o kterých si uživatelská komunita myslí, že by měla být součástí výsledků hledání. V našem příkladu s kapelou Kabát nemusel uživatel, jenž video nahrál, uvést, že se jedná o českou kapelu. Protože je však k dispozici značka Czech, může kdokoli hledající české klipy nalézt také tento klip kapely Kabát.

- **Přihlášení k odběru videí** – YouTube sleduje vkládání videí jednotlivými uživateli. Díky tomu je možné, aby se ostatní uživatelé přihlásili k odběru nahrávek určitého uživatele. Svými videi si díky tomu už několik lidí vysloužilo na YouTubu svých 15 minut slávy.
- **Komunita** – YouTube umožňuje vybudování silné sociální infrastruktury tím, že umožňuje uživatelům komentovat videa.
- **Vkládání do jiných stránek** – YouTube umožňuje uživatelům, aby vkládali videa na své vlastní stránky. YouTube pak v podstatě vystupuje jako poskytovatel úložného prostoru pro videa. Externí stránky mohou mimo to také zobrazovat pouze rychlý náhled videa s odkazem na video a všechny s ním spojené informace na YouTube.

Sociální síť okolo YouTube nahrála do karet vzniku webové služby s dobrým jménem, čehož stránka také náležitě využila.

Rozhraní API pro vývojáře

Stejně jako některé jiné webové služby, se kterými jsme se setkali, vyžaduje také rozhraní API serveru YouTube, aby se při volání webové služby serveru předal i identifikátor vývojáře. Tento identifikátor můžete získat na adrese *http://www.youtube.com/signup?next=my_profile_dev*. Poté, co máte vlastní identifikátor, můžete se ponořit do dokumentace. Dokumentace se nachází na adrese *http://www.youtube.com/dev_docs*.

Dostupné operace se dělí do dvou kategorií.

Operace se může vztahovat k uživatelským informacím nebo k zobrazení videa. YouTube používá pro pojmenování nabízených metod tečkovou notaci podobnou jazyku Java. Např. metoda pro přístup k profilu uživatele má název `youtube.users.get_profile`, metoda pro přístup k seznamu nabízeních videí má název `youtube.videos.list_featured`.

Každou metodu je možné zavolat buďto pomocí REST nebo XML-RPC. Požadavek REST bere jako parametr název metody společně s jakýmikoli dalšími potřebnými parametry. Používá se zápis ve tvaru *http://www.youtube.com/api2_rest?method=NÁZEV_METODY&parametr1=HODNOTA1&parametr2=HODNOTA2*. Pojďme se podívat např. na metodu `get_profile`, která získá profil uživatele. Dokumentace této metody (nacházející se na adrese *http://youtube.com/dev_api_ref?m=youtube.users.getprofile*) říká, že tato metoda vyžaduje tři parametry:

- Parametr `method` představuje název samotné metody. Pro metodu `get_profile` je hodnotou tohoto parametru formální název metody, tj. `youtube.users.getprofile`. Tento parametr se vyžaduje jen pro požadavky REST, které v tomto příkladu používáme.
- Parametr `dev_id` představuje identifikátor vývojáře.
- Parametr `user` představuje název uživatele, jehož profil se má získat.

Požadavek REST představující volání metody get_profile by vypadal následovně: *http://www.youtube.com/api2_rest?method=youtube.users.getprofile&dev_id=ID_VÝVOJÁŘE&user=NÁZEV_PROFILU*.

Stejné názvy metod se používají i při komunikaci pomocí XML-RPC. Při použití XML-RPC představuje požadavek standardní volání XML-RPC s pouze jedním elementem `param` obsahujícím strukturu (element `struct`) a každým parametrem volané metody uvedeným v dceřiném elementu `member` elementu `struct`. Název volané metody se nepředává v elementu `struct`. Místo toho se postupuje podle standardu XML-RPC a název metody se vkládá do elementu `methodName` rodičovského elementu `methodCall`. Požadavek XML-RPC představující volání metody `get_profile` by vypadal následovně:

```
<?xml version="1.0" ?>
<methodCall>
  <methodName>youtube.users.get_profile</methodName>
  <params>
    <param>
      <value>
        <struct>
          <member>
            <name>dev_id</name>
            <value>
              <string>ID VÝVOJÁŘE</string>
            </value>
          </member>
          <member>
            <name>user</name>
            <value>
              <string>NÁZEV PROFILU</string>
            </value>
          </member>
        </struct>
      </value>
    </param>
  </params>
</methodCall>
```

Odpověď serveru vždy představuje dokument XML. Odpověď na volání každé metody detailně popisuje dokumentace. Pokud se použije REST, je odpovědí řetězec obsahující dokument XML. Pokud se použije XML-RPC, pak se tentýž řetězec s dokumentem XML naformátuje jako platná odpověď XML-RPC.

Na metodě `get_profile` si můžeme ukázat, jak to celé pracuje. Informace o struktuře konkrétní odpovědi se nachází v dokumentaci. Podle dokumentace bude odpověď v případě použití REST vypadat následovně:

```
<user_profile>
  <first_name>Petr</first_name>
  <last_name>Novak</last_name>
  <about_me>Pracuju jako programator.</about_me>
  ...
  <friend_count>3</friend_count>
  <favorite_video_count>7</favorite_video_count>
  <currently_on>false</currently_on>
</user_profile>
```

Pokud bychom volání provedli za pomoci XML-RPC, byl by vrácený dokument XML stejný, ale tentokrát by byl zabalen do struktury odpovědi XML-RPC:

```
<?xml version='1.0' ?>
<methodResonse>
  <params>
    <param>
      <value><string>
        <user_profile>
          <first_name>Petr</first_name>
          <last_name>Novak</last_name>
          <about_me>
            Pracuju jako programator.
          </about_me>
          ...
          <friend_count>3</friend_count>
          <favorite_video_count>7</favorite_video_count>
          <currently_on>false</currently_on>
        </user_profile>
      </string></value>
    </param>
  </params>
</methodResponse>
```

Jak vidíte, je rozhraní API pro YouTube jednoduché, konzistentní a dobře dokumentované.

Seznámení s Last.fm

Stránka Last.fm (http://www.last.fm) sama sebe označuje jako největší veřejnou hudební platformu. Jedná se o zajímavý způsob, jak najít novou hudbu vloženou uživateli. Uživatelé se mohou pro přístup ke stránce zaregistrovat zdarma. Poté je třeba stáhnout si malý klientský program, který poběží na vašem počítači. Tento program sleduje jakou hudbu si pouštíte v přehrávačích médií jako je iTunes nebo WinAmp. Hudbu můžete pomocí programu také označkovat. Program poté odešle informace o přehrávané hudbě, včetně značek, serveru Last.fm. V závislosti na získaných informacích o interpretech, skladbách, žánrech a na tom, jak ostatní uživatelé označkovali skladby, vytvoří server Last.fm váš hudební profil. Tento profil pak používá k tomu, aby vytvořil vaši vlastní streamovanou stanici rádia, kterou můžete poslouchat pomocí klientského programu a také k tomu, aby doporučil další interprety a skladby, jež se podobají hudebnímu stylu, který máte rádi.

Poznámka: Ostražitým uživatelům mohou pochopitelně vstávat vlasy na hlavě při představě instalace programu, který odesílá informace o jejich hudebních preferencích serveru nacházejícímu se neznámo kde. Server Last.fm však odvádí při ochraně vašeho soukromí dobrou práci. V prvé řadě je klientský program open-source. Neobsahuje žádný malware, který by mohl ohrozit vaše soukromí a zabezpečení vašeho počítače a jeho programový kód je pro všechny zcela otevřený. Za druhé jejich licenční ujednání jasně uvádí, že server nesdílí vaše osobní informace s žádnou další stranou. Konečně se pak při registraci nevyžaduje zadání informací jako je adrese nebo celé jméno, ačkoli můžete zadat svůj kód ZIP nebo poštovní kód, abyste získali seznam hudebních událostí ve vaší oblasti. Hudební profil, který se vygeneruje, je spjat s vytvořeným uživatelským účtem. Jakékoli osobní informace týkající se vaší osoby, coby jednotlivce, se nevyžadují.

Jak budeme v tomto projektu a příkladech postupovat, demonstrujeme si vše na základě mého osobního uživatelského účtu. To zaručí, že příklady skutečně fungují. Přejete-li si tento projekt upravit pro svou potřebu, měli byste zvážit vytvoření účtu a uploadování dat o přehrávaných skladbách skrze klientský program pro server Last.fm.

Webová služba Audioscrobbler

Webová služba Audioscrobbler je webová služba umožňující přístup k datům zobrazeným na stránkách Last.fm. Domovská stránka této služby se nachází na adrese *http://www.audioscrobbler.net/data/webservices*. Tato webová služba je v podstatě kolekcí kanálů RSS, které spadají do několika kategorií:

- **Data o uživatelském profilu** – největší kolekce kanálů. Tato kategorie poskytuje data o určitém uživateli. Jsou zde např. kanály uvádějící uživatelem nejčastěji poslouchané interprety a skladby.
- **Data o interpretovi** – tato kategorie agreguje data o všech interpretech v databázi Last.fm. Zadáním jména interpreta můžete získat informace jako jsou jeho

nejvíce poslouchané skladby na serveru Last.fm, jejich největší fanoušky, uživatele, kteří ho poslouchali nejvíce atd.

- **Data o albu** – tato kategorie obsahuje informace o albech. S touto kategorií je momentálně spojen pouze jeden kanál. Zadáním názvu alba do kanálu Info je možné získat seznam skladeb tohoto alba.
- **Data o skladbě** – zde je možné získat informace o specifických skladbách.
- **Data o značkách** – stejně jako YouTube umožňuje i Last.fm uživatelům popisovat skladby značkami.
- **Data o skupině** – podobně jako u ostatních stránek s veřejně dostupnou hudbou i stránka Last.fm nabízí skupiny, do nichž se mohou uživatelé přihlašovat. Informace o tom, co členové v těchto skupinách poslouchají, jsou dostupné na kanálech této kategorie.
- **Data z diskusního fóra** – stránka Last.fm nabízí diskusní fórum. Příspěvky jsou dostupné pomocí kanálu RSS.
- **Protokol LiveJournal** – služba Audioscrobbler je schopna komunikovat s webovou službou LiveJournal a umožňuje přístup k protokolu této komunikace.

Při pohledu na domovskou stránku webové služby Audioscrobbler je patrné, že každý kanál může být až ve čtyřech formátech – čistý text, obecné XML, XSPF nebo RSS. Dostupné formáty jsou často dány účelem kanálu. Nedávalo by např. smysl poskytovat informace o uživatelích ve formátu XSPF, formátu vytvořeném speciálně pro prezentaci playlistů.

Tato webová služba není příliš dobře dokumentovaná, ale je velmi jednoduchá a funkci dokumentace přebírají příklady. Na domovské stránce služby se nachází odkazy na kanály v různých formátech, které slouží jako příklady použití kanálů. Ty z těchto příkladů, které získávají uživatelské informace pracují s uživatelem RJ, což je uživatelské jméno jednoho z původních zakladatelů Last.fm. Příklady získávající informace o interpretech pracují s kapelou Metallica. Chcete-li kterýkoli z kanálů použít, jednoduše použijte URL kanálu a formátu, který chcete a nahraďte použité uživatelské jméno resp. jméno interpreta požadovaným uživatelským jménem resp. jménem interpreta.

Pojďme se podívat na některé z příkladů. S kanály služby Audioscrobbler se snadno pracuje i experimentuje, protože jsou pouhými URL a pro jejich použití není zapotřebí žádný identifikátor vývojáře. Domovská stránka služby nabízí odkazy na kanály obsahující uživatelem nejčastěji přehrávané skladby ve formátu čistého textu, ve formátu XML serveru Last.fm a ve formátu XSPF. URL ke kanálu pro získání nejoblíbenějších skladeb uživatele RJ ve formátu XSPF by vypadalo takto: *http://ws.audioscrobbler.com/1.0/user/RJ/toptracks.xspf*.

Pokud si přejete získat informace o mých oblíbených skladbách, stačí uživatelské jméno RJ nahradit mým uživatelským jménem na serveru Last.fm: *http://ws.audioscrobbler.com/1.0/user/ShuTheMoody/toptracks.xspf*.

Příklady s kanály obsahujícími informace o interpretech používají kapelu Metallica. Pro zobrazení seznamu největších fanoušků této kapely na serveru Last.fm stačí použít odkaz na kanál s názvem Top Fans na domovské stránce: *http://ws.audioscrobbler.com/1.0/artist/Metallica/fans.xml*. Pro zobrazení největších fanoušků jiného interpreta stačí adekvátně změnit výraz `Metallica` v URL: *http://ws.audioscrobbler.com/1.0/artist/Donnas/fans.xml*.

Parsing pomocí PEAR

Kdybychom začali s vytvářením mashupu už nyní, museli bychom vytvořit tři různé parsery pro zpracování všech tří formátů odpovědí, které jsme si představili – formát XSPF, odpověď od serveru YouTube ve formátu XML a formát RSS. Museli bychom se prokousat dokumentací a vytvořit flexibilní parsery pro všechny tyto formáty. Pokud se struktura odpovědi kteréhokoli z těchto zdrojů změní, museli bychom změnit programový kód našeho parseru. Nejedná se o těžký úkol, ale je vhodné upozornit na to, že někdo jiný už tuto práci udělal za nás. Někdo jiný už data z kódu XML vyextrahoval. Abychom ušetřili čas, použijeme výsledek této práce v našem mashupu.

PEAR jsme použili již dříve, v kapitole 1, při parsingu XML-RPC. Pro tento projekt opět použijeme PEAR, který nás ušetří před vytvářením parserů pro všechny dokumenty XML, se kterými se setkáme.

Pro potřeby tohoto projektu se podíváme na tři balíky. Balík File_XSPF slouží pro extrakci dat a vytváření playlistů ve formátu XSPF. Balík Services_YouTube je balík vytvořený speciálně pro práci s rozhraním API serveru YouTube. Konečně pak balík XML_RSS slouží pro práci s kanály RSS.

V případě našeho projektu máme štěstí, že jsou k dispozici tři specifické balíky, které vyhovují našim požadavkům na použité formáty. Pokud bychom potřebovali pracovat s dokumenty XML, které nejsou ve formátu jemuž odpovídá některý balík PEAR, bylo by možné použít balík XML_Unserializer. Tento balík vezme dokument XML a vrátí jej jako řetězec.

Je PEAR to pravé pro vás?

Ještě než začnete instalovat balíky PEAR, je vhodné podívat se, zda -li je jejich použití pro konkrétní projekt vůbec vhodné. Balíky PEAR se instalují pomocí správce balíků, nástroje pro příkazový řádek dostupného v každé instalaci jazyka PHP. Abyste mohli nainstalovat balík PEAR, musíte mít na serveru práva administrátora. Pokud se nacházíte v situaci, kdy si hosting platíte a vaše hostingová společnost není změnám příliš nakloněna, anebo se nacházíte v přísném firemním prostředí, ve kterém je prosazení každé změny na serveru větší boj, než stojí za to, nemusí být instalace balíku PEAR možným řešením. Tyto problémy lze obejít stažením souborů balíku PEAR a jejich instalací do vašeho adresáře přístupného přes webový server. Poté byste však museli závislosti mezi balíky a aktualizace balíků řešit sami. Tyto starosti nemusí stát za to a může být vhodnější vytvořit vlastní programový kód pro zajištění požadované funkcionality.

Na druhou stranu použití balíků PEAR často značně spoří čas. Účelem balíků je buďto zjednodušit náročné úkoly nebo interface pro přístup ke komplexním systémům. Tvůrce balíku

PEAR už udělal obtížnou práci za nás. Navíc protože se balíky vytváří v jazyce PHP a nikoli v jazyce C, jak je tomu v případě rozšíření, měl by být dobrý vývojář v PHP schopen prostudovat programový kód balíku, v případě, že chybí dokumentace. Poslední klíčovou výhodou mnoha balíků, včetně těch, na které se podíváme, je, že jsou to objektově orientované reprezentace čehokoli, s čím komunikují. Hodnoty je možné získat jednoduchým přístupem k atributu objektu a složité operace lze provést jednoduchým zavoláním metody. To pomáhá udržovat programový kód čistý a modulární. To, jestli výhody PEAR převáží nad potenciálními obtížemi s nasazením, závisí na konkrétní situaci.

Instalace a použití balíků

Stejně jako v případě instalace balíku XML-RPC použijeme pro instalaci třech uvedených balíků příkaz `install`. Pokud si vzpomínáte, stačí pro instalaci balíku jednoduše do příkazového řádku zadat příkaz `install` následovaný názvem balíku. V tomto případě však bude zapotřebí nastavit ještě další možnosti, abychom přiměli instalátor k instalaci kódu jenž je ve stadiu beta-testování a ověření všech závislostí.

Pro instalaci balíku File_XSPF se přihlaste k počítači jako správce a zadejte následující příkaz:

```
[Pocitac:~] root# /usr/local/php5/bin/pear install -f --alldeps File_XSPF
```

Tento příkaz zajistí stažení a instalaci balíku (`/usr/local/php5/bin/pear`, případně nahraďte umístěním PEAR na vašem počítači). Možnost `-alldeps` specifikuje, že se mají zkontrolovat závislosti a případně doinstalovat potřebné balíky. O stavu a výsledcích stahování a instalace budete informováni.

Podobným příkazem se nainstaluje balík Services_YouTube:

```
[Pocitac:~] root# /usr/local/php5/bin/pear install -f --alldeps
            Services_YouTube
```

Obyčejně nebude možnost `-f` zapotřebí. Ve výchozím nastavení stáhne PEAR nejnovější stabilní verzi balíku. Možnost `-f`, neboli `force`, přinutí PEAR, aby stáhl nejaktuálnější verzi balíku, bez ohledu na to, jestli byla označena jako stabilní nebo ne. V době psaní této publikace balíky File_XSPF a Services_YouTube nemají stabilní verze, pouze beta, resp. alfa-verzi. Z tohoto důvodu je nutné pro získání a instalaci balíku použít možnost `-f`. V opačném případě bude PEAR hlásit, že nejnovější verze není k dispozici. Pokud je balík, který si přejete stáhnout, dostupný ve stabilní verzi, pak možnost `-f` nebudete potřebovat.

To je také případ balíku XML_RSS, pro který je dostupná stabilní verze.

```
[Pocitac:~] root# /usr/local/php5/bin/pear install --alldeps XML_RSS
```

Po provedení těchto kroků a zadání příkazu `list-all` by měl PEAR zobrazit tři nové balíky společně s dalšími balíky, které jste měli dříve.

Balíky PEAR jsou v podstatě skripty jazyka PHP, které PEAR nainstaluje do adresáře `includes` jazyka PHP. Cestu k tomuto adresáři specifikuje direktiva `include_path` v konfiguračním souboru `php.ini`. Otevřete daný adresář pro zobrazení zdrojových souborů balíků PEAR. Pro použití balíku PEAR je třeba ke skriptu připojit zdrojový soubor balíku, zpravidla hned na začátku skriptu. Pro informace týkající se připojení hlavního souboru balíku prostudujte jeho dokumentaci. Např. balík File_XSPF se aktivuje připojením souboru s názvem `XSPF.php`. PEAR umístil soubor `XSPF.php` do adresáře File, který je podadresářem adresáře `includes`.

```
<?php
require_once 'File/XSPF.php';
// zde je již balík File_XSPF dostupný
```

Balík File_XSPF

Dokumentace nejnovější verze tohoto balíku se nachází na adrese *http://pear.php.net/package/File_XSPF/docs/latest/File_XSPF/File_XSPF.html*.

Balík se velmi snadno používá. Jeho srdcem je objekt s názvem XSPF. Vytvoříme instanci tohoto objektu a použijeme ji pro práci s playlistem. Objekt má metody pro získání i modifikaci hodnot z playlistu, stejně jako obslužné metody pro načtení playlistu do paměti, uložení playlistu z paměti do souboru na disku a konverzi formátu XSPF na jiné formáty.

Získání informací z playlistu se skládá ze dvou snadných kroků. V prvé řadě se umístění souboru XSPF předá metodě `parse` objektu. To zajistí načtení souboru do paměti. Poté, co je soubor načten, je možné použít řadu metod pro získání hodnot z playlistu. Většina z těchto metod slouží pro získání metadat o samotném playlistu. Pro získání informací o skladbách uvedených v playlistu je třeba použít metodu `getTracks`. Tato metoda vrátí pole objektů `XSPF_Track`. Každý objekt `XSPF_Track` v tomto poli reprezentuje jednu skladbu. Následně lze použít metody objektů `XSPF_Track` pro získání informací o jednotlivých skladbách.

Pro ilustraci toho, jak to celé funguje, použijeme playlist ze serveru Last.fm. Webová služba nabízí mimo jiné playlist skladeb nejčastěji přehrávaných určitým uživatelem. Tento playlist s názvem Top Tracks se nachází na adrese *http://ws.audioscrobbler.com/1.0/user/UZIV_JMENO/toptracks.xspf*, kde UZIV_JMENO je uživatelské jméno uživatele serveru Last.fm, který nás zajímá.

Následující skript naleznete mezi příklady jako soubor s názvem `testPearXSPF.php`. Používá balík File_XSPF pro získání playlistu s mými oblíbenými skladbami na serveru Last.fm.

```
<?php
require_once 'File/XSPF.php';
$objXspf =& new File_XSPF();
```

```
// načtení souboru XSPF do paměti
$objXspf->parseFile('http://ws.audioscrobbler.com/1.0/user/
  ShuTheMoody/toptracks.xspf');

// získání všech skladeb v playlistu
$skladby = $objXspf->getTracks();
?>
```

V první části dochází k vytvoření instance objektu `XSPF` a načtení playlistu do paměti. Nejdříve připojíme ke skriptu balík File_XSPF. Poté vytvoříme instanci objektu. Metoda `parseFile` se používá pro načtení souboru XSPF ze sítě. Tím se zvolený playlist načte do objektu `XSPF`. Poté použijeme metodu `getTracks` pro transformaci skladeb uvedených v playlistu na objekty `XSPF_Track`.

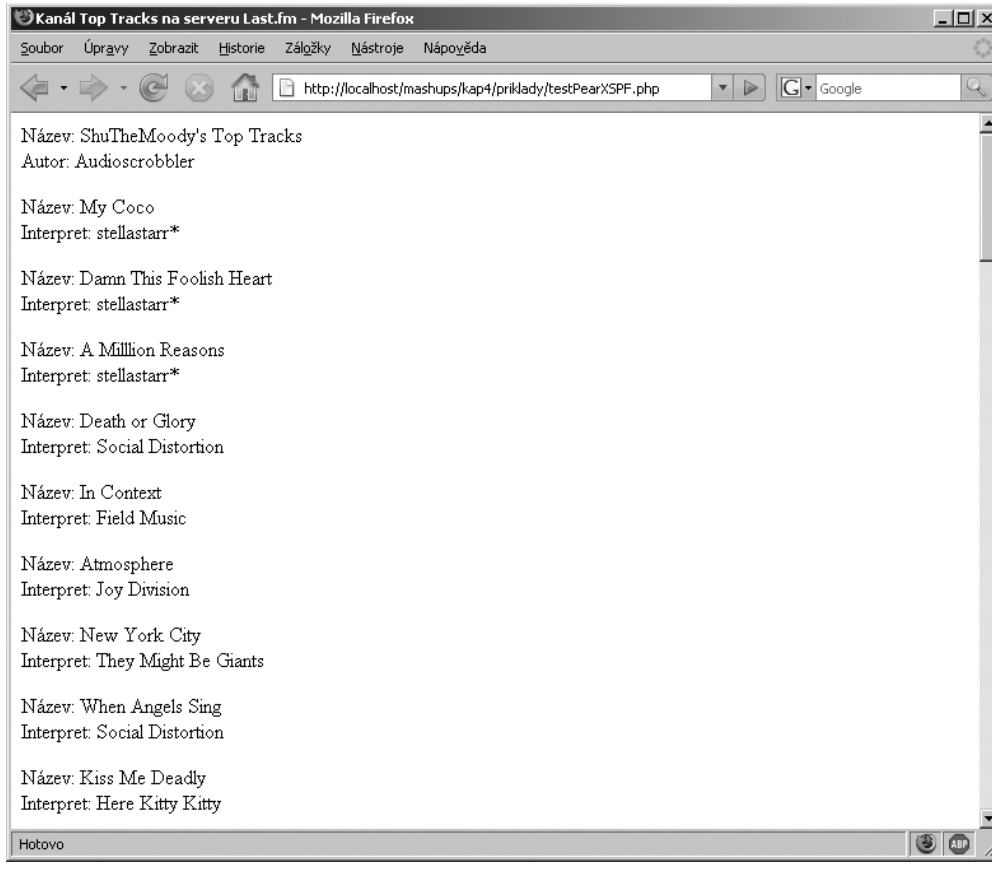
```
<html>
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <title>Kanál Top Tracks na serveru Last.fm</title>
</head>
<body>
  Název: <?= $objXspf->getTitle() ?><br />
  Autor: <?= $objXspf->getCreator() ?>
```

Dále se připravíme na zobrazení obsahu playlistu. Ještě než tak ale učiníme, získáme nějaké informace o samotném playlistu. Metoda `getTitle` objektu `XSPF` vrací element `title` souboru XSPF. Metoda `getCreator` vrací element `creator` souboru.

```
<?php foreach ($skladby as $skladba) { ?>
  <p>
    Název: <?= $skladba->getTitle() ?><br />
    Interpret: <?= $skladba->getCreator() ?><br />
  </p>
<?php } ?>
</body>
</html>
```

Na závěr pak cyklem projdeme polem skladeb. Jednotlivé prvky pole skladeb, objekty `XSPF_Track`, v cyklu postupně přiřadíme do proměnné `$skladba`. Objekt `XSPF_Track` má také metody `getTitle` a `getCreator`. Na rozdíl od stejnojmenných metod objektu `XSPF` metoda `getTitle` v tomto případě vrací název skladby a metoda `getCreator` vrací jméno interpreta skladby.

Pokud tento skript otevřete ve webovém prohlížeči, zobrazí se seznam skladeb získaných ze serveru Last.fm.



Balík Services_YouTube

Balík Services_YouTube pracuje na velmi podobném principu jako balík File_XSPF. Stejně jako balík File_XSPF je i balík Services_YouTube objektově orientovanou abstraktní vrstvou nacházející se nad komplexnějším systémem. V tomto případě je tento systém rozhraní API serveru YouTube.

Používání balíku Services_YouTube je v mnohém stejné jako používání balíku File_XSPF. Stačí připojit balík ke skriptu, vytvořit instanci objektu `Services_YouTube` a použít metody tohoto objektu pro práci se službou. Oficiální dokumentace poslední verze balíku Services_YouTube se nachází na adrese *http://pear.php.net/package/Services_YouTube/docs/latest/*. K balíku jsou dostupné také příklady nacházející se na adrese *http://pear.php.net/manual/en/package.webservices.services-youtube.php*.

Mnoho metod slouží pro získávání informací o uživatelích, jako jsou informace z jejich profilu a o videích, která uploadovali. Menší, avšak velmi důležitá, množina metod se používá pro přístup k videím na serveru YouTube. Tyto metody použijeme v našem mashupu. Konkrétně použijeme metodu `listByTag` pro získání seznamu videí označených specifickou značkou.

Metoda `listByTag` se dotáže serveru YouTube a odpověď ve formátu XML uloží do paměti. Nevrací pole video objektů, se kterými bychom mohli přímo pracovat. Toho však lze dosáhnout pomocí jednoho dalšího volání. Následně můžeme cyklem toto pole videí projít, podobně jako tomu bylo u skladeb XSPF.

Příklad uložený v souboru `testPearYouTube.php` ilustruje tento postup.

```
<?php
  require_once 'Services/YouTube.php';
  $idVyv = 'Váš identifikátor vývojáře pro přístup k YouTube';
  $znacka = 'Social Distortion';
  $youtube = new Services_YouTube($idVyv, array('usesCache' => true));
  $videa = $youtube->listByTag($znacka);
?>
```

Nejdříve připojíme ke skriptu balík Services_YouTube. Protože webová služba serveru YouTube vyžaduje pro přístup identifikátor vývojáře, uložíme jej do lokální proměnné s názvem `$idVyv`. Poté do další lokální proměnné, s názvem `$znacka`, uložíme značku, s jejíž pomocí budeme vyhledávat. V tomto příkladu budeme na YouTube hledat videa jedné z mých nejoblíbenějších skupin s názvem Social Distortion. Prvním parametrem konstruktoru třídy `Services_YouTube` je identifikátor vývojáře, který se pak použije při každém dotazu na webovou službu YouTube. Druhým parametrem konstruktoru je pole možností. Jednou z možností použitých v tomto příkladu je použití vyrovnávací paměti pro dotazy. Obecně se považuje použití vyrovnávací paměti za vhodné, aby nedocházelo k přetěžování serveru YouTube a zbytečně jste si nevyčerpávali váš limit na požadavky.

Další možností, v poli možností reprezentovanou klíčem `driver`, je specifikace protokolu – REST nebo XML-RPC. Ve výchozím nastavení používá objekt `Services_YouTube` REST. Pokud nemáte nezvladatelnou touhu použít XML-RPC, můžete ponechat toto nastavení tak, jak je.

Po vytvoření instalace objektu můžeme zavolat jeho metodu `listByTag` pro získání odpovědi od serveru YouTube. Metoda `listByTag` bere pouze jediný parametr, kterým je požadovaná značka.

Objekt `Services_YouTube` nyní má výsledky od serveru YouTube. Můžeme začít s jejich zobrazováním.

```
<html>
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <title>Videa skupiny Social Distortion</title>
</head>
```

```
<body>
<h1>Výsledky hledání skupiny Social Distortion na serveru YouTube</h1>
```

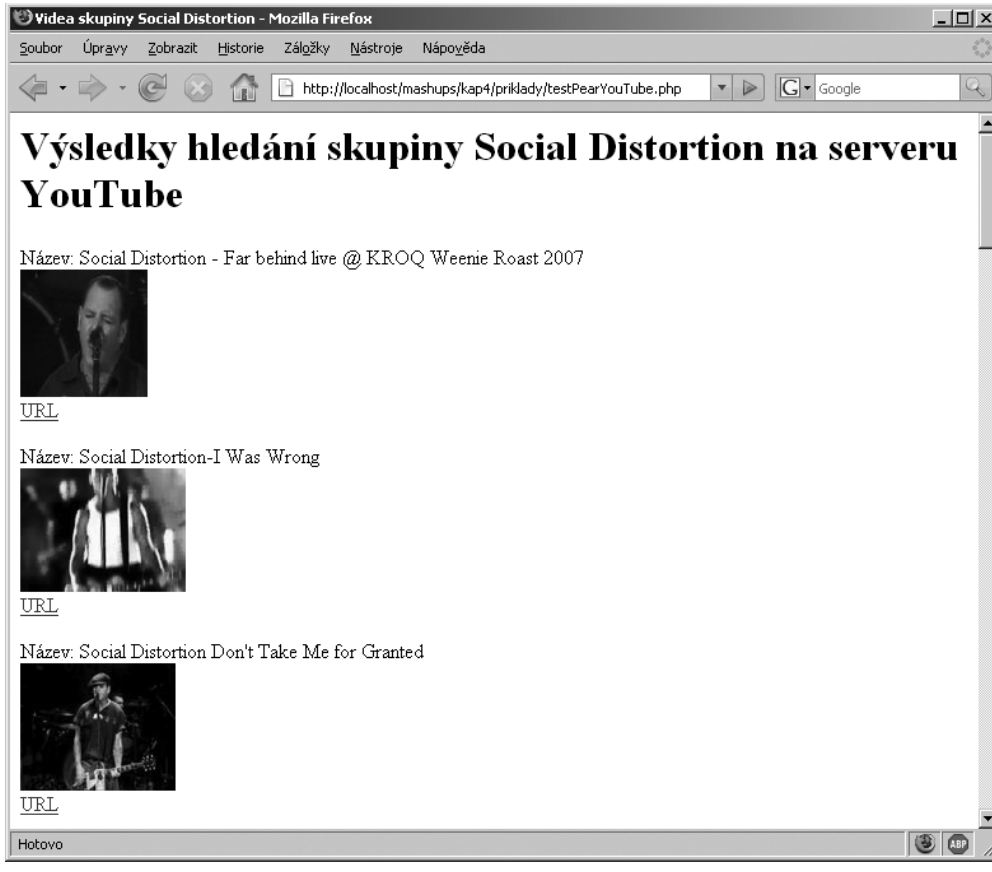
Následně projdeme v cyklu polem s videi. Abychom získali toto pole s video objekty, musíme nejdřív zpracovat odpověď ve formátu XML od serveru. K tomu použijeme metodu `xpath` objektu `Services_YouTube`, která využívá mocný jazyk XPATH pro průchod dokumentem XML a jeho konverzi na objekty jazyka PHP. Dotaz jazyka XPATH předáme jako parametr metody `xpath`, která vrátí pole odpovídajících objektů. V dalším projektu se podíváme na jazyk XPATH a dotazy v tomto jazyce. Prozatím postačí důvěřovat tomu, že dotaz `//video` vrátí pole video objektů, se kterými můžeme dále pracovat.

V cyklu zobrazíme název každého videa, jeho náhled a odkaz na video samotné.

```
<?php foreach ($videa->xpath('//video') as $i => $video) { ?>
<p>
  Název: <?= $video->title ?><br />
  <img src='<?= $video->thumbnail_url ?>' alt='<?= $video->title ?>'
    /><br />
  <a href='<?= $video->url ?>'>URL</a>
</p>
<?php } ?>
</body>
</html>
```

Otevření tohoto skriptu ve webovém prohlížeči zobrazí stránku s výsledky vyhledávání videí podle vyhledávacích parametrům, jež jsme specifikovali.

Poznámka: Před spuštěním skriptu `testPearYouTube.php` je zapotřebí nainstalovat `CURL`.



Balík XML_RSS

Podobně jako předešlé balíky PEAR mění balík XML_RSS něco velmi složitého, RSS, na něco velmi jednoduchého a snadno použitelného, objekty jazyka PHP. Kompletní dokumentace tohoto balíku se nachází na adrese *http://pear.php.net/package/XML_RSS/docs/XML_RSS*.

V porovnání s balíky Services_YouTube a File_XSPF je zde však drobná změna ve filozofii. První dva zmíněné balíky načtou informace odkudkoli jim řekneme a uloží je do atributů objektů jazyka PHP.

Např. balík File_XSPF uloží název skladby, společně s dalšími informace o skladbě, do objektu `XSPF_Track` a poté můžeme použít metodu `getTitle` pro získání názvu skladby. V balíku Services_YouTube se setkáváme se stejným principem, ale atributy jsou zde veřejné, takže nejsou zapotřebí metody, které by data vracely. K informacím o videu přistupujeme přímo pomocí atributů objektu.

V případě balíku XML_RSS jsou hodnoty, o které máme zájem, uloženy v asociativních polích. Dostupné metody objektu vrací tato pole, se kterými pak přímo pracujeme. Je to malý rozdíl, ale v případě, že vás programový kód zajímá, je třeba ho mít na paměti. Také je třeba prostudovat dokumentaci balíku, abychom věděli, jaké klíče jsou v polích dostupné.

Pojďme si na příkladu ukázat, jak to celé pracuje. Tento příklad naleznete mezi ukázkovými kódy jako soubor s názvem `testPearRSS.php`. Jeden z kanálů služby Audioscrobbler podává informace o skladbách nedávno přehrávaných uživatelem ve formátu RSS. Tento kanál někdy neobsahuje žádná data, protože po několika hodinách již nejsou přehrané skladby považovány za nedávno přehrávané. Jinými slovy informace o přehraných skladbách z kanálu jednoduše zmizí, protože jsou již příliš zastaralé. Z tohoto důvodu je pro náš příklad vhodné prozkoumat obsah kanálu pro uživatele doslova závislého na poslechu hudby ze serveru Last.fm.

Uživatel RJ je pro naše potřeby jako dělaný. Zdá se, jako by pořád něco poslouchal. Získáme obsah jeho kanálu služby Audioscrobbler:

```
<?php
  include ("XML/RSS.php");
  $rss =& new XML_RSS("http://ws.audioscrobbler.com/1.0/user/RJ/
    recenttracks.rss");
  $rss->parse();
```

Začneme připojením balíku ke skriptu a vytvořením objektu `XML_RSS`. Je to právě objekt XML_RSS, ve kterém se nachází všechny metody pro přístup k asociativním polím a je srdcem celého balíku. Konstruktor bere jediný parametr – cestu k souboru RSS. Při vytváření instance objektu se načte soubor RSS do paměti.

Metoda `parse` je metodou, která provede vlastní parsing dokumentu RSS. Poté je možné volat další metody pro získání dat z kanálu. Jistě tedy není třeba dodávat, že je metodu `parse` nutné zavolat předtím než lze se souborem provádět něco konstruktivního.

```
  $kanalInfo = $rss->getChannelInfo();
?>
```

Metoda `getChannelInfo` vrací pole obsahující informace o kanálu. Prvky tohoto pole obsahují elementy `title`, `description` a `link` souboru RSS. Každý z těchto elementů je v poli uložen pod klíčem se stejným názvem jako je název elementu.

```
<?= "<?xml version=\"1.0\" encoding=\"UTF-8\" ?>" ?>
```

Data, která se vrací, jsou ve znakové sadě UTF-8. Proto nesmíme zapomenout zobrazit stránku s tímto kódováním. Tento řádek vypíše deklaraci XML na začátek webové stránky, aby se zajistilo správné zobrazení. Pokud bychom použili běžnou deklaraci `<?xml`, pak by jazyk PHP tuto deklaraci zpracoval jako svůj vlastní kód. To by však vedlo k chybě, protože se nejedná o platný programový kód jazyka PHP.

```
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
    <title><?= $kanalInfo['title'] ?></title>
  </head>
  <body>
  <h1><?= $kanalInfo['description'] ?></h1>
```

Zde začínáme s vlastním zobrazením stránky. Začneme použitím pole vráceného metodou `getChannelInfo` pro zobrazení elementů `title` a `description` kanálu.

```
<ol>
  <?php foreach ($rss->getItems() as $polozka) { ?>
    <li>
      <?= $polozka['title'] ?>:
      <a href="<?= $polozka['link'] ?>"><?= $polozka['link'] ?></a>
    </li>
  <?php } ?>
</ol>
```

Poté vypíšeme položky uvedené v souboru RSS. Pro získání informací o položkách v souboru RSS použijeme metodu `getItems`. Ta vrací pole, kterým projdeme pomocí příkazu `foreach`. V těle cyklu přistupujeme k elementům `title` a `link`. Vypíšeme název skladby a poté vytvoříme hypertextový odkaz na stránku skladby na serveru Last.fm. Elementy `description` a `pubDate` ze souboru RSS jsou v poli vráceném metodou `getItems` také dostupné.

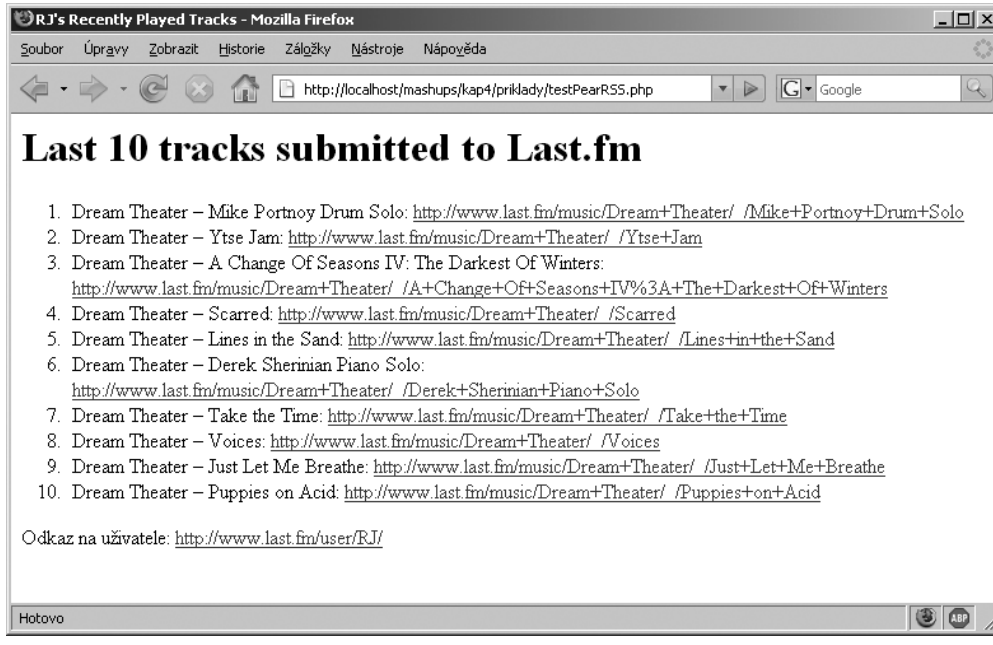
```
    Odkaz na uživatele:
    <a href="<?= $kanalInfo['link'] ?>"><?=
      $kanalInfo['link'] ?></a>
  </body>
</html>
```

Na závěr, ještě než stránku ukončíme koncovými značkami elementů `body` a `html`, použijeme klíč `link` pole s informacemi o kanálu pro vytvoření hypertextového odkazu na stránku uživatele na serveru Last.fm.

Použití dalších elementů

V tomto příkladu je v polích s informacemi o kanálu a položkách počet dostupných elementů omezený. Metoda `getChannelInfo` vrací pole obsahující pouze prvky `title`, `description` a `link`. Pole vracené metodou `getItems` obsahuje pouze prvky `title`, `description`, `link` a `pubDate`. Je to způsobeno tím, že používáme nejnovější stabilní verzi balíku XML_RSS. V době psaní této publikace se jedná o verzi 0.9.2. Novější verze balíku XML_RSS, momentálně ve stadiu beta-testování, umožňuje přístup k mnoha dalším elementům. Elementy RSS 2.0 jako je `category` a `authors` jsou také dostupné. Přejete-li si aktualizovat na beta-verzi balíku XML_RSS je třeba do příkazového řádku zadat příkaz `PEAR upgrade -f XML_RSS`. Použitá možnost `-f` je tatáž, kterou jsme použili pro vynucení instalace beta a alfa-verze balíku Services_YouTube, resp. File_XSPF. Alternativně můžete hned na začátku, s využitím možnosti `-f`, nainstalovat beta verzi balíku XML_RSS.

Pokud tento skript otevřete ve webovém prohlížeči, můžete se přesvědčit o úspěšnosti našeho postupu.



V tomto okamžiku již víme, jak se používají kanály služby Audioscrobbler pro získání informací. Většina kanálů je buďto ve formátu XSPF, nebo RSS. Známe také základy práce s rozhraním API serveru YouTube. A co je nejpodstatnější, víme, jak použít příslušné balíky PEAR pro získání informací z jednotlivých webových služeb. Je na čase začít programovat naši aplikaci.

Vytvoření mashupu

Pokud jste tak doposud neučinili, měli byste si přinejmenším vytvořit svůj účet na serveru YouTube a získat svůj identifikátor vývojáře. Také byste si měli vytvořit účet na serveru Last.fm, nainstalovat klientský software a začít na počítači přehrávat oblíbenou hudbu. To upraví obsah videotéky podle vašich osobních hudebních choutek. Veškeré zde uvedené příklady předpokládají, že používáte svůj vlastní identifikátor vývojáře pro přístup k YouTube. V příkladech používám vlastní účet na serveru Last.fm. Jelikož jsou kanály otevřené a volně dostupné, můžete použít stejné kanály jaké já, pakliže se rozhodnete nezakládat si vlastní účet na serveru Last.fm.

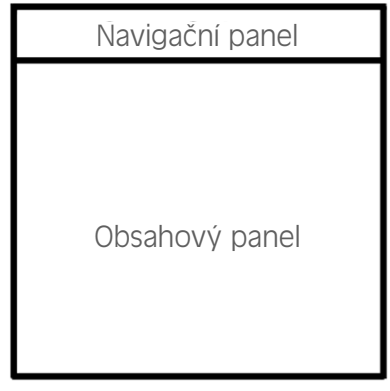
Architektura mashupu

Existuje bezesporu mnoho způsobů, jak by mohla naše aplikace vypadat. My se však pokusíme udržet aplikaci velmi jednoduchou.

Interface bude tvořit webová stránka složená z rámců. Horní rámec představuje navigační panel a bude sloužit pro výběr skladby. Dolní rámec představuje obsahový panel, ve kterém budeme zobrazovat a přehrávat video.

V navigačním panelu vytvoříme nabídku se všemi našimi skladbami. Hodnota o popisek každého prvku nabídky bude jméno interpreta následované pomlčkou, následovanou názvem skladby (např. April Smith – Bright White Jackets). Tím, že uvedeme oba důležité střípky informace o hledaném videu, pomůžeme zpřesnit výsledky hledání na YouTube.

Když uživatel vybere skladbu a stiskne tlačítko Vyhledat, načte aplikace stránku s obsahem do obsahového panelu. Tento formulář předá stránce s obsahem informace o interpretovi a skladbě ve formě parametru `GET`. Stránka s obsahem použije tento parametr pro vyhledání videa na serveru YouTube. Stránka vybere ze seznamu výsledků vyhledávání první, nejvíce relevantní, video a zobrazí ho.



Hlavní stránka

Hlavní stránka je uložena v souboru s názvem `videoteka.html`. Zde je definováno rozdělení stránky na rámce. Tato stránka bude velmi jednoduchá, pouze definuje rozložení rámců, které použijeme.

```
<html>
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <title>Má videotéka</title>
</head>
  <frameset rows="10%,90%">
```

```
    <frame src="navigace.php" name="Navigace" />
    <frame src="" name="Obsah" />
  </frameset>
</html>
```

Tento kód definuje, jak bude naše stránka vypadat. Použijí se dva rámce. Rámec s navigační stránkou, s názvem `Navigace`, zaujímá 10 % z celkové výšky stránky. Rámec s obsahovou stránkou, s názvem `Obsah`, zaujímá zbylých 90 % z celkové výšky. Při prvním otevření stránky se vždy načte pouze seznam skladeb do navigační stránky.

Navigační stránka

Navigační stránka je uložena v souboru s názvem `navigace.php`. Tato stránka bude používat balíky File_XSPF a XML_RSS pro získání informací o skladbách z kanálů Top Tracks a Recent Tracks serveru Last.fm. Jejich spojením vytvoříme obsah nabídky navigační stránky.

```
<?php
  require_once ('File/XSPF.php');
  require_once ('XML/RSS.php');
```

Na začátku se ke skriptu připojí balíky File_XSPF a XML_RSS.

```
$poleSkladeb = array();
```

Vytvoříme pole pro uložení všech skladeb. Potřebujeme ho, protože pracujeme se dvěma kanály – Top Tracks a Recent Tracks. Oba kanály jsou v jiném formátu. Z prvního z nich získáme s využitím balíku File_XSPF pole objektů skladeb. Z druhého pak pomocí balíku XML_RSS získáme asociativní pole hodnot, v němž názvy elementů figurují jako klíče prvků pole.

```
// kanál Top Tracks
$objXspf =& new File_XSPF();
$objXspf->parseFile('http://ws.audioscrobbler.com/1.0/user/
  ShuTheMoody/toptracks.xspf');
$kanalTopTracks = $objXspf->getTracks();
```

Nejdříve vytvoříme pole objektů skladeb z kanálu Top Tracks (jenž je ve formátu XSPF) pomocí balíku File_XSPF.

```
// kanál Recent Tracks
$rss =& new XML_RSS('http://ws.audioscrobbler.com/1.0/user/
  ShuTheMoody/recenttracks.rss');
$rss->parse();
$kanalRecentTracks = $rss->getItems();
```

Následně vytvoříme asociativní pole z druhého kanálu, Recent Tracks (jenž je ve formátu RSS), pomocí balíku XML_RSS.

```
foreach ($kanalTopTracks as $skladba) {
  $poleSkladeb[] = $skladba->getCreator() . " - " . $skladba->getTitle();
}
```

Použijeme metody `getCreator` a `getTitle` objektů skladeb pro vytvoření řetězce ve tvaru „Interpret – Skladba“, a uložíme ho do pole `$poleSkladeb`.

```
foreach ($kanalRecentTracks as $polozka) {
  $tempSkladba = htmlentities($polozka['title'], ENT_COMPAT, 'UTF-8');
  $poleSkladeb[] = str_replace('&ndash;', "-", $tempSkladba);
}
```

Ve výchozím stavu tento kanál RSS obsahuje elementy `title`, jejichž hodnoty již jsou ve tvaru „Interpret – Skladba“. Můžeme tedy tyto hodnoty přímo použít a vložit je do pole `$poleSkladeb`. Přesto musíme nejdříve provést drobné úpravy. Protože data pochází z dokumentu XML, jsou některé znaky zakódovány. Jedním z takovýchto znaků je také znak pomlčky, nacházející se mezi jménem interpreta a názvem skladby. Z tohoto důvodu zavoláme funkci jazyka PHP s názvem `htmlentities`, která převede speciální znaky použité v dokumentu XML na ekvivalentní entity jazyka HTML. V tomto případě se z pomlčky uvedené v souboru RSS stane entita `&nd ash;`. Takto upravená hodnota se uloží do proměnné s názvem `$tempSkladba`. Na následujícím řádku se nahradí entity `&ndash;` běžným znakem pomlčky a výsledek se uloží do pole `$poleSkladeb`.

```
$poleSkladeb = array_unique($poleSkladeb);
sort($poleSkladeb);
```

Konečně pak upravíme seznam do podoby již je možno prezentovat. Funkce `array_unique` eliminuje duplicitní skladby nacházející se v obou kanálech, Top Tracks i Recent Tracks. Funkce `sort` setřídí seznam podle abecedy. Tím jsme dokončili potřebný úvodní programový kód v jazyce PHP a můžeme začít s kódem v jazyce HTML.

```
?>
<?= '<?xml version="1.0" encoding="UTF-8" ?>' ?>
<html>
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <title>Výběr</title>
</head>
<body>
  <form method="GET" action="obsah.php" target="Obsah">
  <select name="vyber">
  <?php foreach ($poleSkladeb as $klic => $hodnota) { ?>
    <option value="<?= $hodnota ?>"><?= $hodnota ?></option>
  <?php } ?>
  </select>
  <input type="submit" value="Vyhledat" />
  </form>
</html>
```

Před samotným začátkem kódu HTML opět vypíšeme deklaraci XML pro specifikaci kódování této stránky. Následující kód HTML vytvoří formulář. Tento formulář používá metodu `GET` pro předání vybrané položky nabídky skriptu obsah.php. Cílovým rámcem tohoto formuláře je rámec s názvem `Obsah`. Pro vytvoření nabídky použijeme pole `$poleSkladeb`, které jsme vytvořili dříve.

Na závěr přidáme tlačítko, které zahájí načtení videa. Po něm následují koncové značky elementu `form` a `html`.

Obsahová stránka

Naše obsahová stránka, s názvem `obsah.php`, se načte při odeslání dat formulářem v navigační stránce. Formulář odešle jméno interpreta a název skladby skriptu `obsah.php` jako parametr s názvem `vyber`. Stránka `obsah.php` načte tento parametr, předá ho rozhraní API serveru YouTube a zobrazí výsledek.

```
<?php
  require_once ('Services/YouTube.php');

  // parametry pro YouTube
  $idVyv = 'VÁŠ IDENTIFIKÁTOR VÝVOJÁŘE';
  $znacka = $_GET['vyber'];

  // výsledek dotazu na server YouTube
  $poleVidei = array();
  $prvniVideo = null;
```

Začneme jednoduchou inicializací. Ke skriptu připojíme balík Services_YouTube, inicializujeme proměnnou `$idVyv`, obsahující identifikátor vývojáře nutný pro přístup k rozhraní API serveru YouTube, a proměnnou `$znacka`, do které uložíme hodnotu parametru `vyber` získaného z formuláře.

Následně vytvoříme pole `$poleVidei` pro uložení výsledků vyhledávání. Poté se deklaruje proměnná s názvem `$prvniVideo`. Vyhledávání může přinést, a často také přinese, více výsledků s nejvíce relevantním výsledkem uvedeným jako prvním. Proměnná `$prvniVideo` bude obsahovat nejvíce relevantní výsledek.

```
$youtube = new Services_YouTube($idVyv, array('usesCache' => true));
$videa = $youtube->listByTag($znacka);

$poleVidei = $videa->xpath('//video');
$prvniVideo = $poleVidei[0];
```

První tři řádky tohoto kódu pracují stejně jako v případě příkladu s balíkem Services_YouTube – vytvoří se objekt `Services_YouTube`, zavolá se jeho metoda `listByTag` a výsledky se analyzují pomocí jazyka XPATH.

Nyní máme pole všech výsledků. Protože se zajímáme pouze o nejvíce relevantní výsledek, vybereme ho z pole coby prvek s indexem 0 a uložíme ho do proměnné `$prvniVideo`. Kód v jazyce PHP je připravený a můžeme začít s kódem jazyka HTML.

```
?>
<?= '<?xml version="1.0" encoding="UTF-8" ?>' ?>
<html>
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <title>Obsah</title>
</head>
<body>
<h1>Výsledky vyhledávání <?= $_GET['vyber'] ?> na serveru YouTube</h1>
<?php if ($prvniVideo) { ?>
```

V kódu jazyka HTML narazíme na příkaz `if` jazyka PHP. Tento podmíněný blok `if` kontroluje, jestli je proměnná `$prvniVideo` nastavena.

```
Název: <?= $prvniVideo->title ?><br />
<object width="425" height="350">
<param name="movie" value="http://www.youtube.com/v/<?=
  $prvniVideo->id ?>"></param>
<param name="wmode" value="transparent"></param>
<embed src="http://www.youtube.com/v/<?= $prvniVideo->id ?>"
  type="application/x-shockwave-flash" wmode="transparent"
  width="425" height="350"></embed>
</object>
<p>
<a href="<?= $prvniVideo->url ?>"><?=
  $prvniVideo->title ?> na YouTube</a>
</p>
```

Pokud je proměnná `$prvniVideo` nastavena, použijeme ji pro vytvoření obsahu stránky. Jedná se o objekt, který má tři atributy, jež v této části použijeme:

- `title` – název videa na serveru YouTube.
- `id` – unikátní identifikátor videa na serveru YouTube.
- `url` – URL stránky s videem na serveru YouTube.

Atribut `id` je obzvláště důležitý. Každá stránka s videem na serveru YouTube obsahuje vzorový kód, který můžeme zkopírovat do naší webové stránky a vložit tak do ní video. Video se tak stane součástí stránky a bude ho možné přímo z ní přehrávat. Vzorový kód se skládá z elementu `object`, dvou elementů `param` a elementu `embed`. Tento kód jsme přenesli do naší aplikace. Přepíšeme však pevně vložené identifikátory hodnotou atributu `id` našeho objektu.

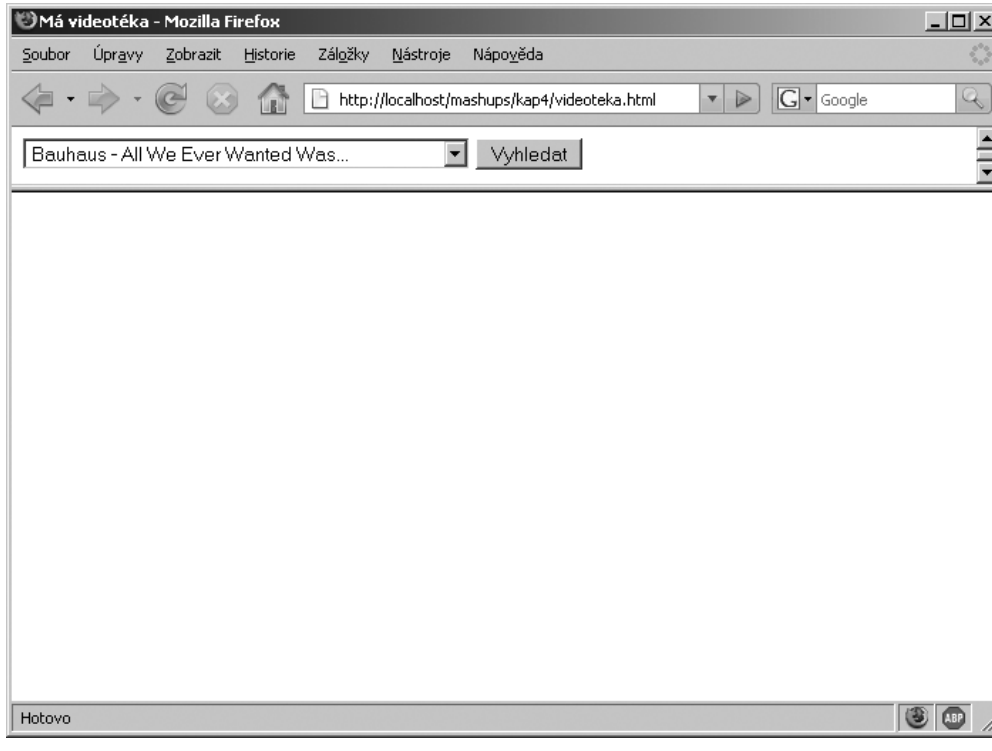
```
<?php } else { ?>
  Nebyly nalezeny žádné výsledky
<?php } ?>
</body>
</html>
```

Přidáme blok else pro zobrazení hlášky `Nebyly nalezeny žádné výsledky`, pro případ, že YouTube nenajde žádné video odpovídající zadaným kritériím. Na závěr uzavřeme elementy `body` a `html`.

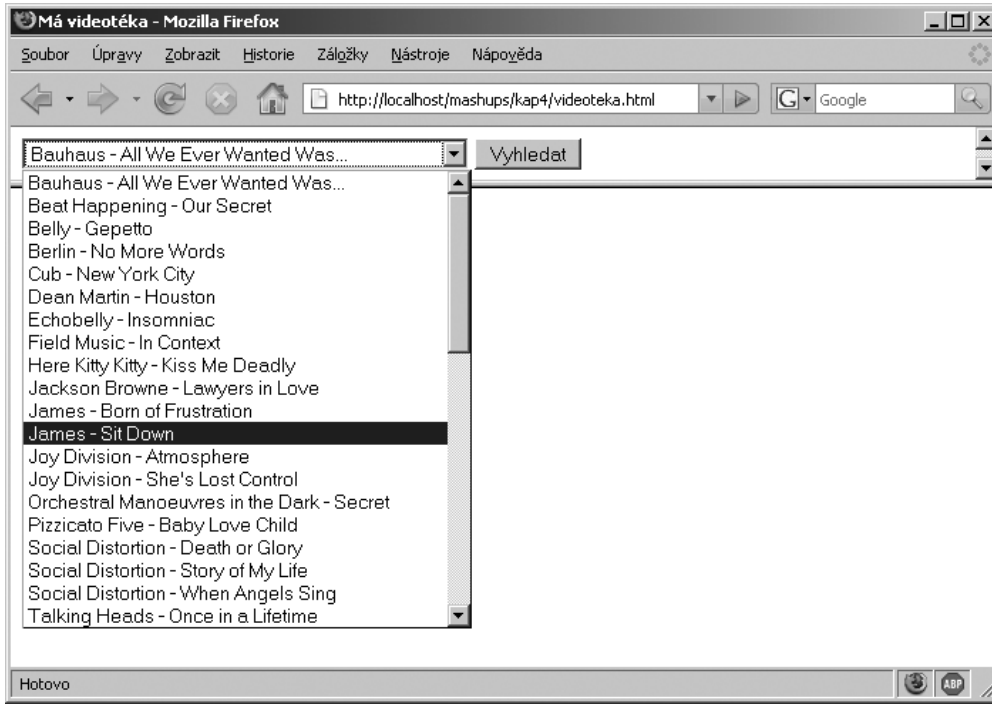
Všechny tři zmíněné soubory uložte do přístupné oblasti webového serveru. Nyní můžete otevřít v prohlížeči stránku `videoteka.html` a začít mashup používat.

Používání mashupu

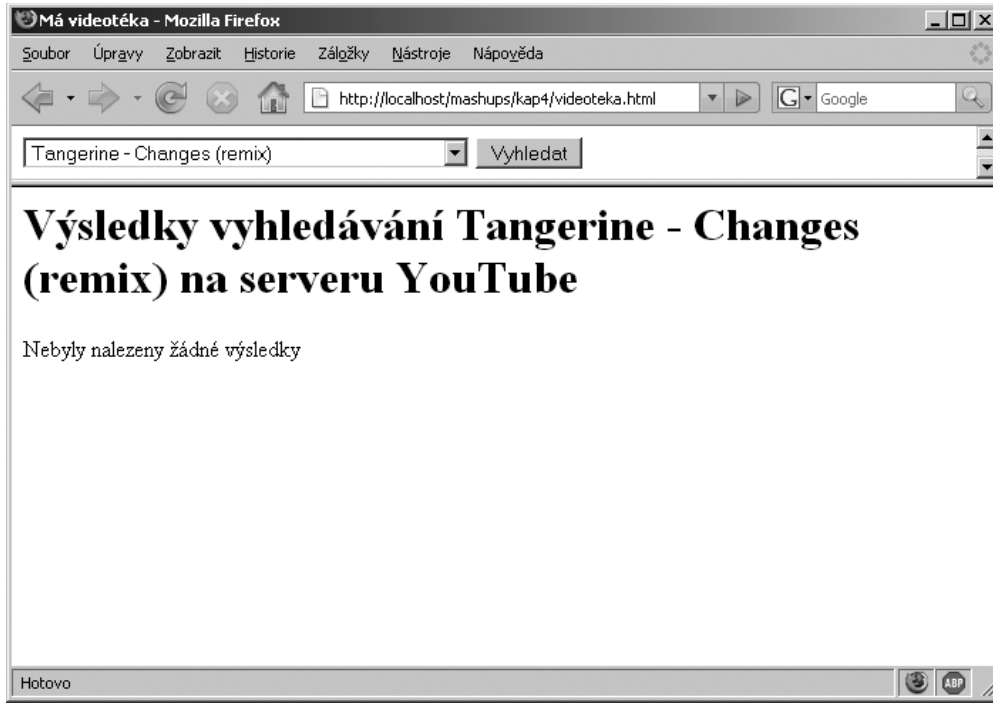
Používání mashupu je velmi jednoduché. Otevřete v prohlížeči stránku `videoteka.html`.



Poté, co stránku `videoteka.html` otevřete, zobrazí se stránka podobná té znázorněné na předchozím obrázku. V horním rámci se načte stránka `navigace.php` a na základě obsahu daných kanálů ze serveru Last.fm se vytvoří nabídka.



Klepněte na nabídku. Zobrazí se seznam skladeb ze dvou uvedených kanálů. Vyberte požadovanou skladbu a stiskněte tlačítko `Vyhledat`.



Můj první výběr, skladba skupiny Tangerine, nevedl k žádným výsledkům. V obsahové stránce se proto zobrazí chybová zpráva. Nejedná se o nic překvapivého, protože se jedná o malou, začínající kapelu z Pittsburghu, Pennsylvanie. Jakmile budou populárnější, počet jejich fanoušků vzroste a některý z nich snad něco přidá na YouTube.

Pojďme vybrat o něco známější skupinu a skladbu.

Náš další výběr, skladba Atmosphere, jejímž autorem je Joy Division, byl úspěšnější. V tomto případě někdo na YouTube uploadoval videoklip k této skladbě. Stisknutím tlačítka `Play` zahájíte přehrávání videa, aniž byste opustili mashup.

Skvělou věcí na tomto mashupu je uživatelská podstata serveru YouTube. Nelze si být jist tím, jaký výsledek bude vyhledávání mít. V případě předchozího hledání jsme mohli vidět oficiální videoklip, který na YouTube někdo nahrál. V mnoha ostatních případech uvidíme živá vystoupení, která někdo nahrál videokamerou a poté uploadoval. Někdy se setkáme s kreativním uživatelem, který si ke skladbě již hledáme, vytvořil své vlastní video. Jindy nemusí mít video se skladbou žádnou spojitost, pouze tu, že skladba hraje na jeho pozadí. Např. pokud jsem zadal klasiku v podobě skladby Death or Glory skupiny Clash, bylo nalezeno jediné video, slideshow fotek Daniela Radcliffa, představitele Harryho Pottera, které někdo vytvořil.

Shrnutí

V tomto mashupu jsme použili dvě rozdílná rozhraní API – jedno serveru pro ukládání videí s názvem YouTube, druhé hudebního serveru Last.fm. Podívali jsme se na tři rozdílné formáty souborů založené na formátu XML těchto stránek – formát XSPF pro playlisty, RSS pro publikaci často aktualizovaných informací a vlastní formát na bázi XML serveru YouTube. Vytvořili jsme mashup, který získal informace o skladbách ze serveru Last.fm a na jejich základě vyhledal video na serveru YouTube.

Pokud bychom se rozhodli vytvořit naše vlastní parsery pro tyto tři formáty, trvalo by to mnohem více času než tomu bylo teď. Ukázali jsme si, že PEAR (PHP Extension and Application Repository) již nabízí parsery, které můžeme použít. Jeden pro každý ze tří formátů. Pomocí balíků PEAR jsme byli schopni vytvořit objektově orientované abstrakce těchto formátů, s jejichž pomocí jsme snadno dokončili naši aplikaci.