

Vývoj pro portál Liferay

Miroslav Ligas <miroslav.ligas@ibacz.eu>

Důvěrné

- Časový rozvrh
 - Přestávky
 - Oběd
- Rozvržení obsahu
- Kooperace a komunikace

- Cíle
- Základní architektonický přehled
- Zdrojový kód portálu Liferay a zásuvných modulů
- Portálové API
- Plugins SDK
- Základy vývoje front-endu

Vývojové prostředí

- Rozbalení
- Nastavení
- Použití z příkazové řádky
- Provázání s IDE

Eclipse

- Unzip Eclipse
- Liferay IDE
- Další moduly
 - Subclipse
 - m2e
- Import projektů
- Vytváření nových projektů

Netbeans

- Install Netbeans
- Portal pack

Zdrojový kód portálu

- Liferay portál je od začátku OpenSource
- Hlavní výhodou otevřenosti zdrojového kódu je rychlost řešení problémů
 - Debug code – Fix problem – Deploy patch
- Zdroje
 - Předpřipravené balíčky (na SourceForge)
 - SVN – <http://svn.liferay.com/repos/public/> (*user=guest, no pass*)
 - Git – <https://github.com/liferay/>

Dávná historie

- Monolitická aplikace obsahující všechny portlety

Historie

- EXT prostředí pro úpravy a rozšiřování
- Zásuvné moduly – portlety, témata, šablony rozložení
- Hooky
- Extlet

Současnost

- Funkcionalita jako pluginy
- Ext Plugin

Budoucnost

- OSGi

- ***portal-service***
 - Jádro systému
 - Rozhraní všech služeb
 - Statické pomocné třídy pro volání služeb
- ***portal-impl***
 - Konfigurace (properties, XMLs)
 - Implementace služeb
 - Implementace portletů

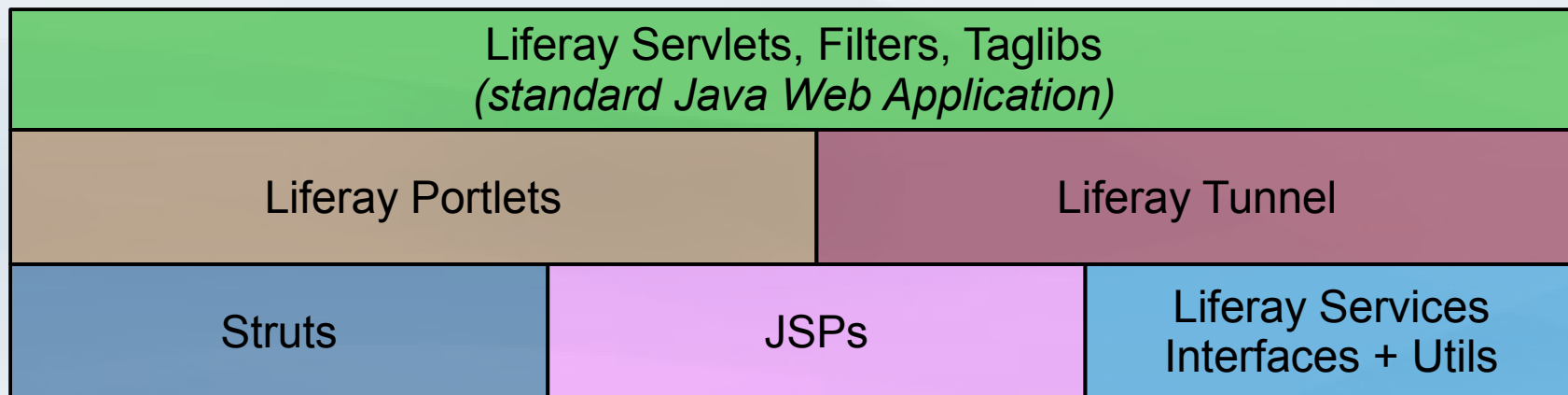
- ***portal-web***
 - Konfigurace webové aplikace
 - JSP stránky
 - Sdílené
 - Portlety
 - Vestavěná témata a šablony rozložení
 - JavaScripts
 - Cascade Style Sheets
 - Taglibs

- ***lib***
 - Všechny potřebné knihovny včetně verzí a licencí
- ***sql***
 - Všechny SQL pro různé DB
- ***support-maven***
 - Podpora pro buildování mavenem
 - Archetypy pro Liferay pluginy
 - Pluginy (ServiceBuilder, témata vzhledu apod.)
- ***portal-client***
 - Knihovna s klientem pro webové služby

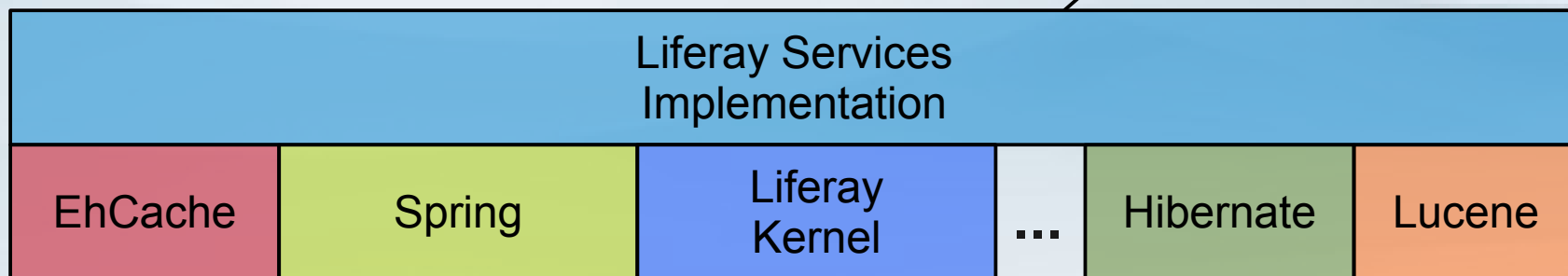
- EXT = Extension environment
 - Původní metoda ohýbání portálu
- Vedle zdrojových kódů portálu existovalo EXT, kde je možné přepsat téměř cokoliv
- V EXT se konfigurační soubory nepřepisují, ale vytváří se přídatný soubor s příponou -ext.
 - Např. *portal.properties* a *portal-ext.properties*
 - Sem se dávají změněné nebo přidané položky
 - Soubory se se potom spojují nebo přepisují na základě logiky portálu
- Tento princip Liferay stále používá na mnoha místech

Architektonický přehled

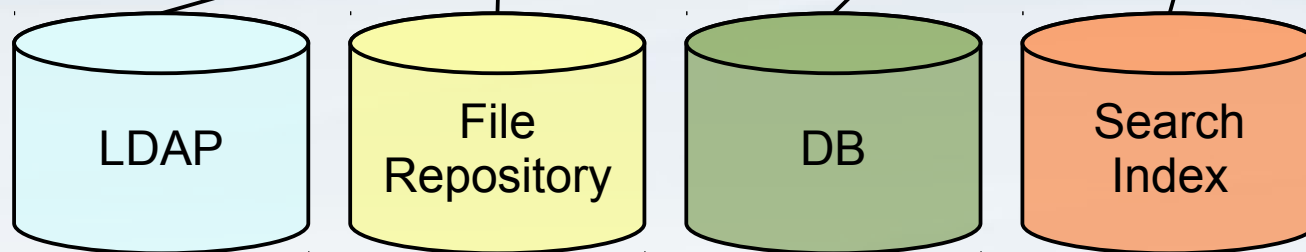
UI



App.
Logic

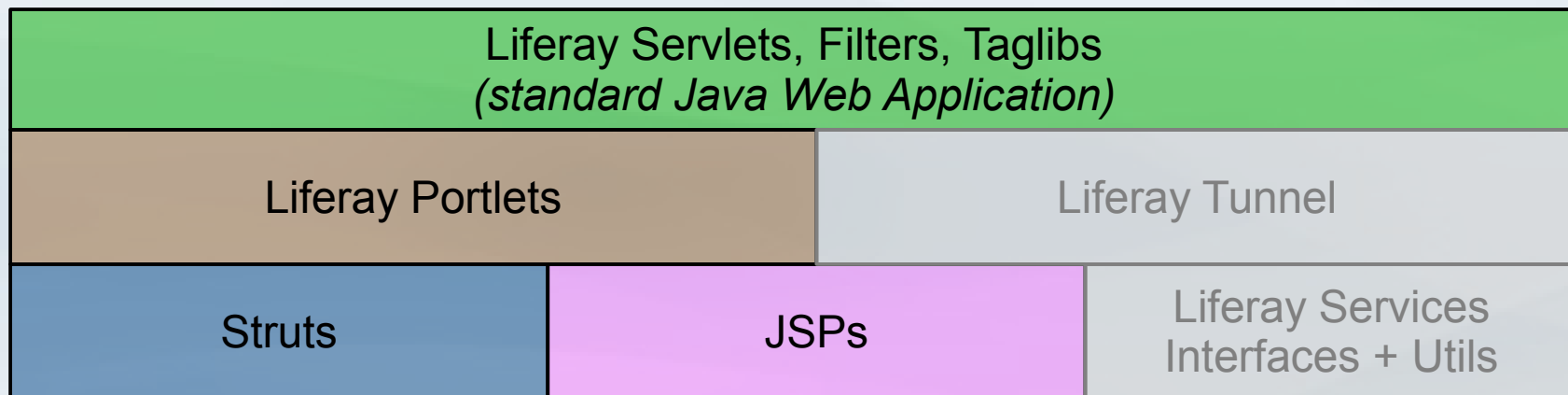


Data

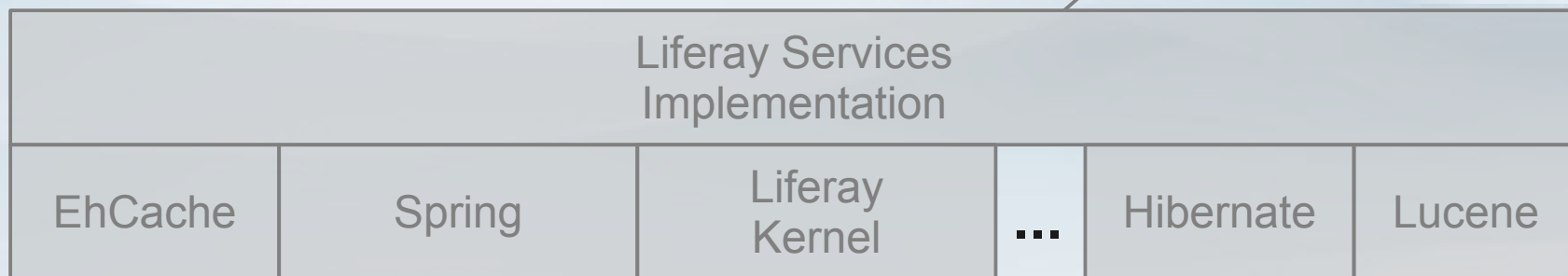


Technologická struktura portálu

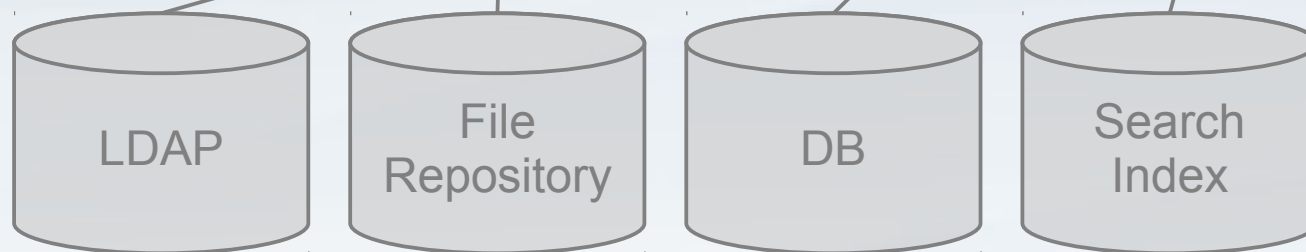
UI

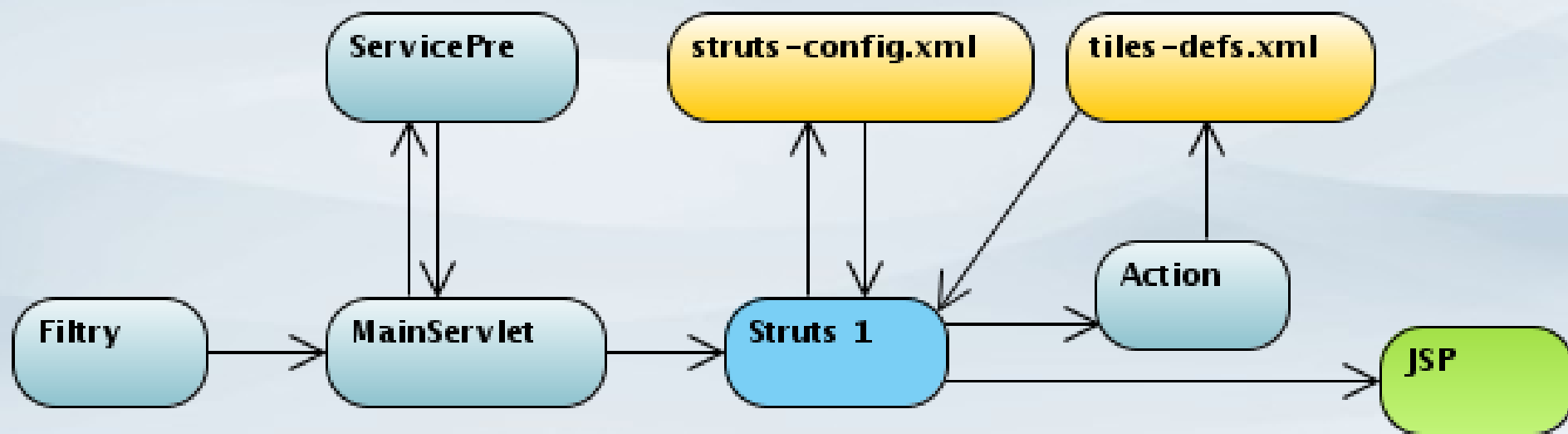


App.
Logic



Data





- Hlavní vstupní bod
- Standardní servlet
 - `init()` – spuštění start-up akcí, inicializace pluginů...
 - `service()`
 - Volá `Service-Pre-Events`
 - Předá řízení frameworku Struts
 - Volá `Service-Post-Events`
 - `destroy()` – uvolnění prostředků a volání `global-shutdown-events`

GlobalStartupAction

- Spuštění vlákna hlídajícího auto deploy
- Registrace tříd HotDeployListener
- Inicializace JCR, JNDI apod.

GlobalShutdownAction

- Uvolnění prostředků
- Vypínání procesů
- V případě vynucení ukončuje všechna běžící vlákna

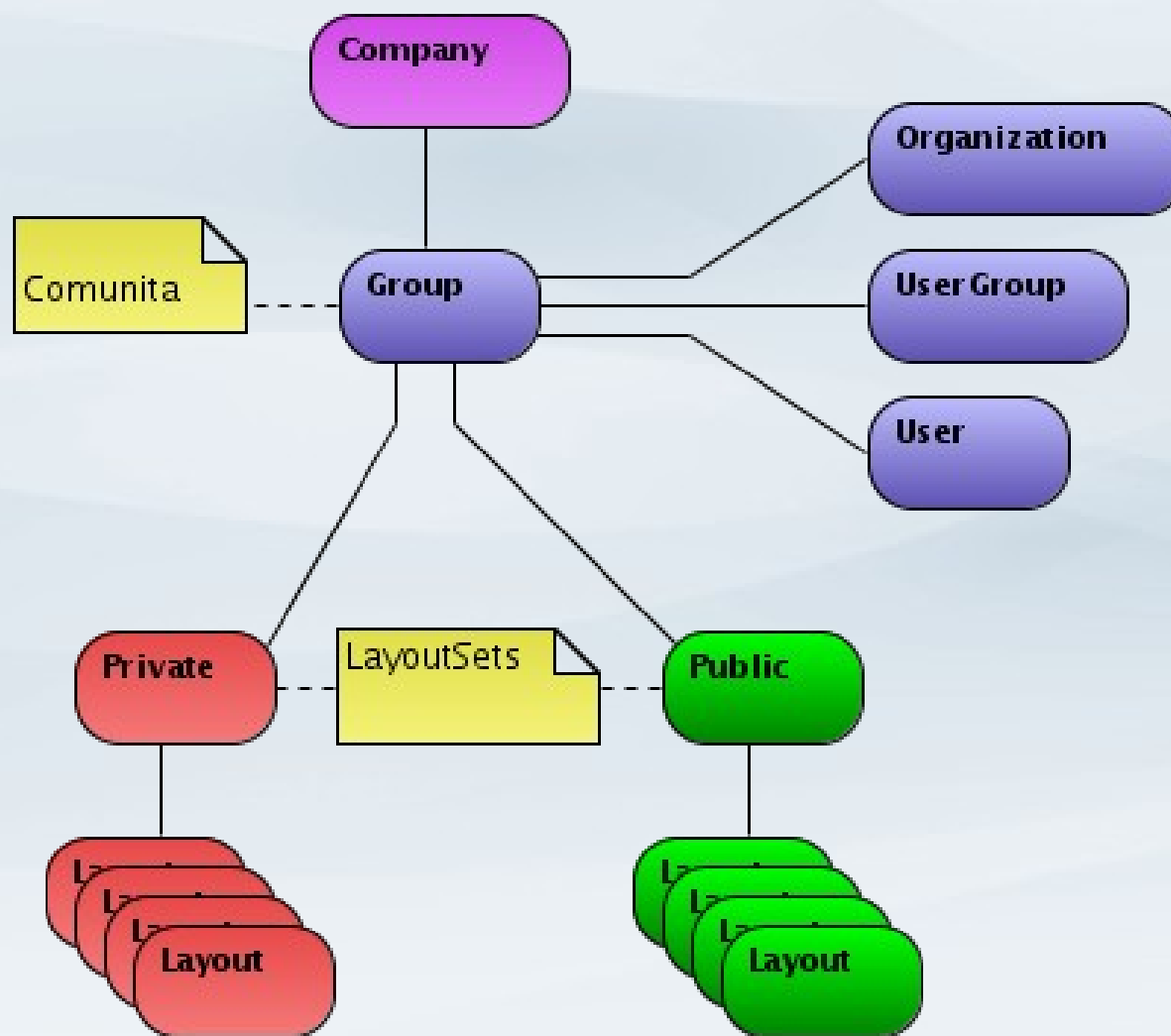
- Spouští se s každým dotazem na server
- Inicializace všech objektů v *HttpServletRequest* a *HttpServletResponse*
- Inicializace informací o
 - Uživateli
 - Aktuální stránce
 - atp.

- Jedna z nejdůležitějších Struts akcí
- Zobrazuje stránku
- Implementuje JSR-286 pro volání jednotlivých portletů
- Spolu s *render_portlet.jsp* zajišťuje vykreslení všech portletů na stránce

Základní entity v portálu

- DB schéma nesplňuje základní normální formy
- Neexistují cizí klíče
 - Možnost DB rozbít ručně přímou změnou dat → používat služby!
- Odkazy jsou tvořeny logicky
 - Podpora v ServiceBuileru
 - Cachování
- Při normalizaci (tabulky pro všechny třídy a jejich potomky) by se DB několikanásobně zvětšila

- Reprezentuje číselník tříd v Liferay.
- Důležitá, při běžném používání neviditelná entita
- Veliké množství dotazů generických entit, aby definovali, jaká implementace je za entitou schovaná
- Často lze ve schématu vidět vektor
[classNameID, primaryKey]
- Např. dříve používaný systém oprávnění



- [illegible]

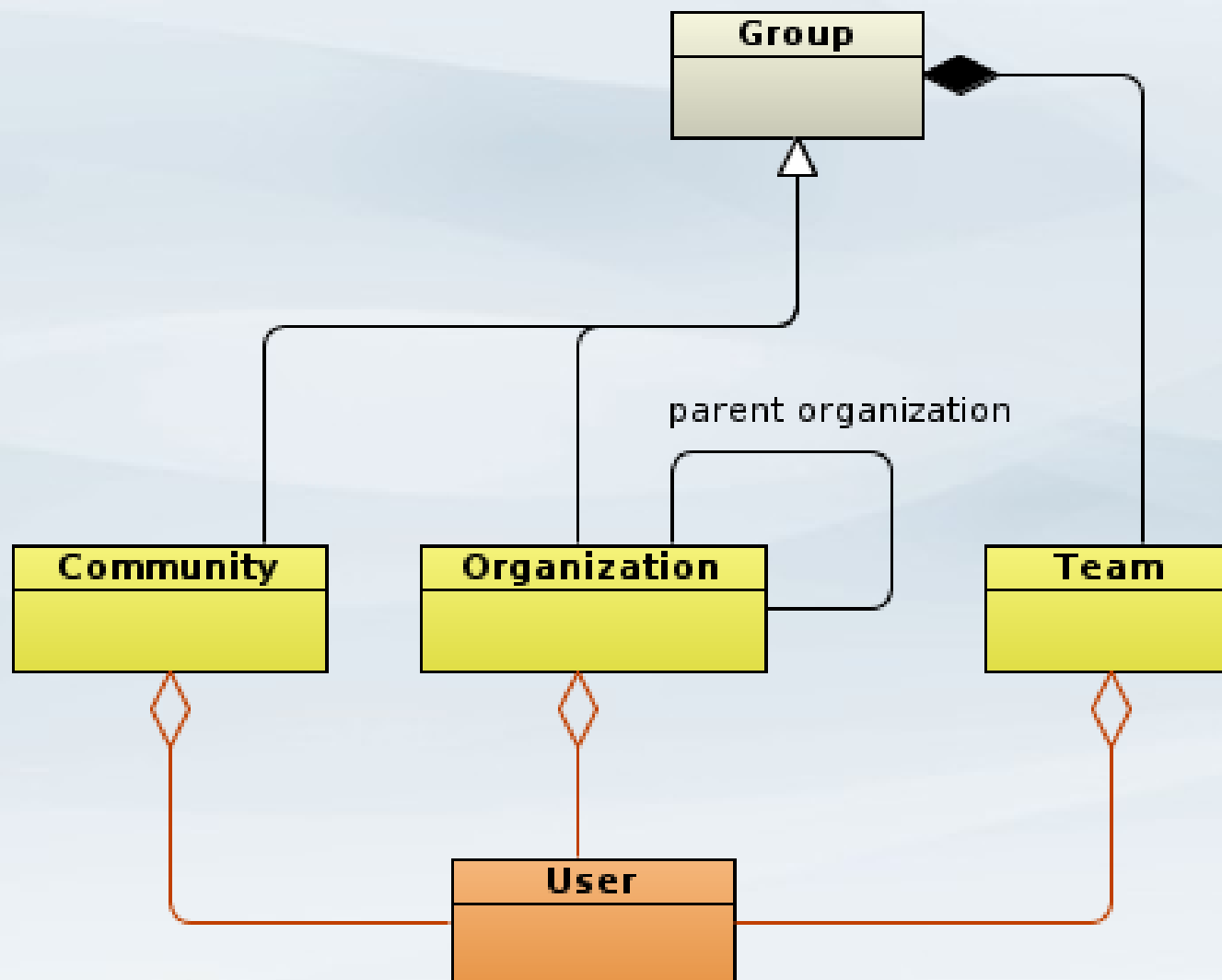
- Reprezentuje sadu portálových stránek
 - Stránky komunity
 - Stránky organizace
 - Stránky uživatelů
 - Šablony webových prezentací
 - Sandboxy stagingu
- Ostatní entity (stránky, portlety ...) jsou navázané na Group
- Group se používá jako kontext pro ověřování práv

- Sada stránek
- Veřejné nebo privátní stránky Groupy
- Obsahuje další informace o prezentaci
 - Virtual host
 - Témata vzhledu
 - Logo

- Portálová stránka
 - Lokalizované informace o stránce
 - Ikona
 - Friendly URL
 - Téma vzhledu
 - Rozložení portletů
 - Další nastavení stránky
 - Nadřazená stránka

- Portlet v rámci company
- Řízení přístupu podle rolí
- Většina informací o portletech z portlet.xml se drží v paměti a nezapisuje se do DB
 - Portlety se nasazují (inicializují) při každém startu portálu

- Reprezentuje nastavení konkrétní instance portletu na stránce
- Společné pro uživatelův
- Preference se v DB drží jako XML
- Cachování

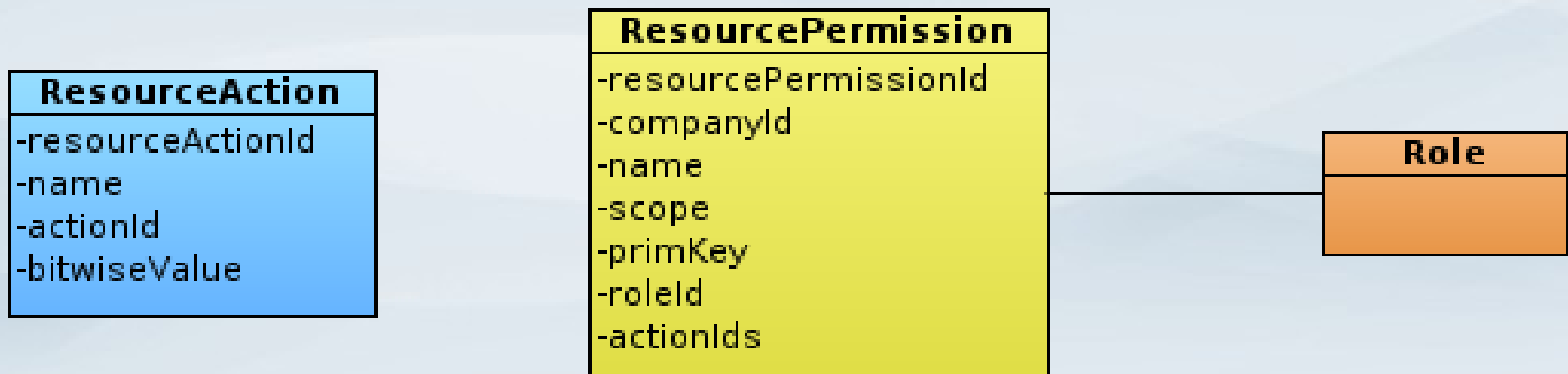


- Má vlastní stránky
- Lze do ní přiřadit uživatele
- Lze řídit práva v rámci komunity (komunitní role)
- Různé restrikce
 - Public – pokud chtějí, mohou se uživatelé stát členy
 - Restricted – o členství se žádá a administrátor jej schvaluje
 - Private – administrátor explicitně definuje členy
- V DB ji nenajdete, jedná se o Group

- Podobné komunitě, ale
 - Uchovávají informace o organizaci (místo, telefony apod.)
 - Možno vytvářet hierarchii
 - Možno definovat typy (Regular Organization, Location, Department...)
 - Členy určuje administrátor

- Reprezentuje skupiny uživatelů
- Často se používá při synchronizaci s LDAPem
- Jinak nemá žádnou speciální funkci

- Uživatel je standardní uživatel v portálu
- Všechny operace v portálu se provádí pod nějakým uživatelem
- Speciální uživatelé
 - *Omniadmins* – neověřují se pro ně práva
 - *Default user* – nepřihlášený uživatel
- Uživatele je možné synchronizovat s LDAPem nebo dalšími externími systémy



- Skupina uživatelů
- Uživatele do role je možné je přiřazovat
 - Přímo
 - Prostřednictvím UserGroup
 - Prostřednictvím Group (organizací nebo komunit)

- **Regular Role**

- Viditelnost všude v portálu
- Platnost napříč všemi komunitami a organizacemi
- Např. *Administrator* může spravovat všechny komunity a jejich členy

- **Community Role a Organization Role**

- Viditelnost v každé komunitě resp. organizaci
- Platnost pouze v kontextu dané komunity
- Např. *Community Administrator* v komunitě XY může uživatelům přiřazovat členství v komunitě XY

- **Team**

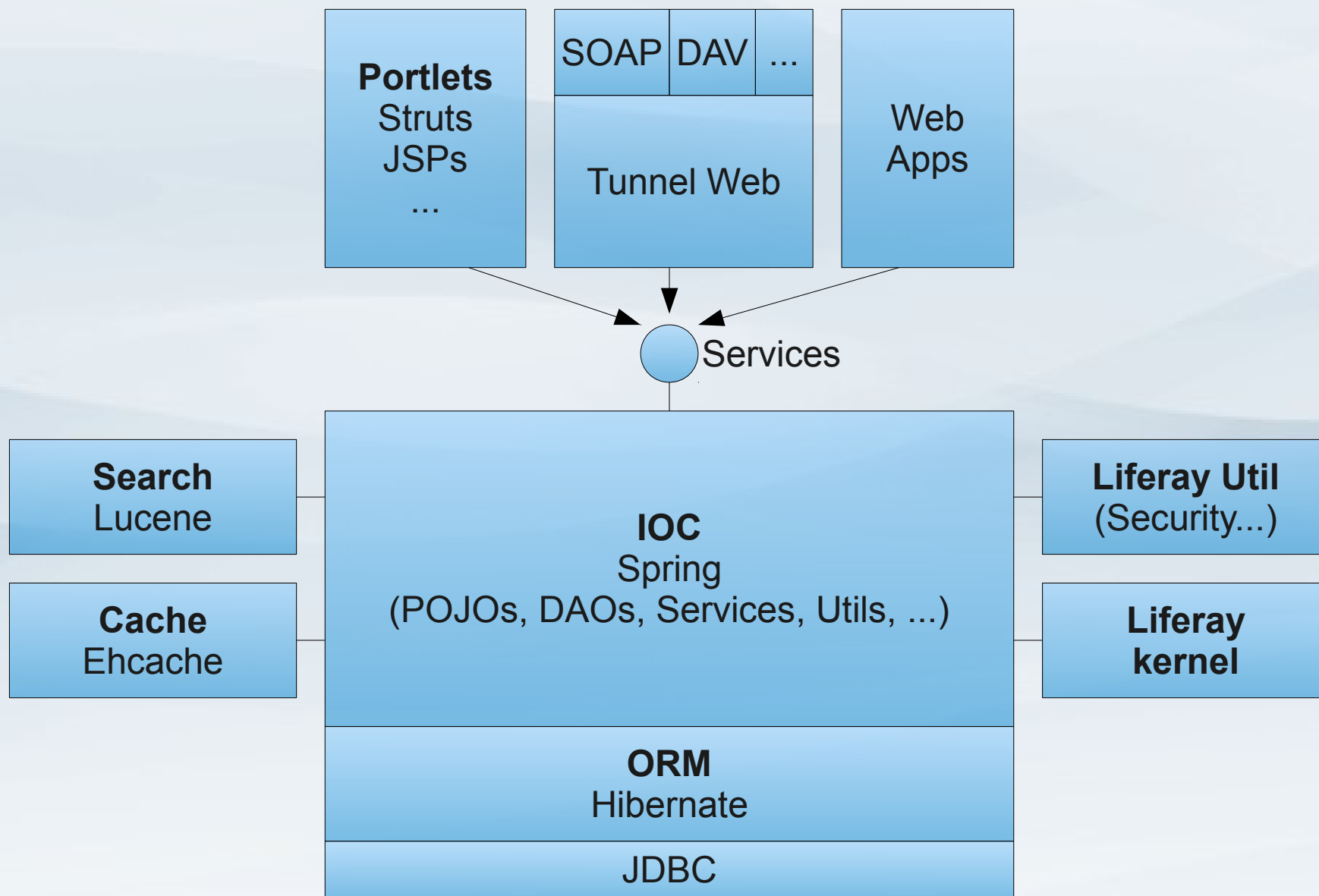
- Viditelnost i platnost pouze v jedné organizaci / komunitě

- Definuje, které akce je možné provádět s nějakou entitou
- Např. uživatele je možné vytvořit, zobrazit, editovat, smazat, přiřadit do komunity, upravit oprávnění apod.

- Používá se pro Algoritmus 6 (poslední algoritmus pro ověřování oprávnění)
- Definuje oprávnění mezi subjektem a objektem pomocí bitového pole
- Např. role *Editor* (subjekt) má na *článku* (objekt) oprávnění provést akce *vytvořit*, *zobrazit*, *editovat* a *publikovat*

- Zástupná entita za vše možné
 - Záznam v blogu
 - Soubor
 - Web Content Article
- Pracuje s nimi
 - Asset Publisher
 - Workflow
 - Tagy a kategorie
 - Od verze 6.1 je možné je navzájem prolinkovat
- Je možné si vytvářet vlastní

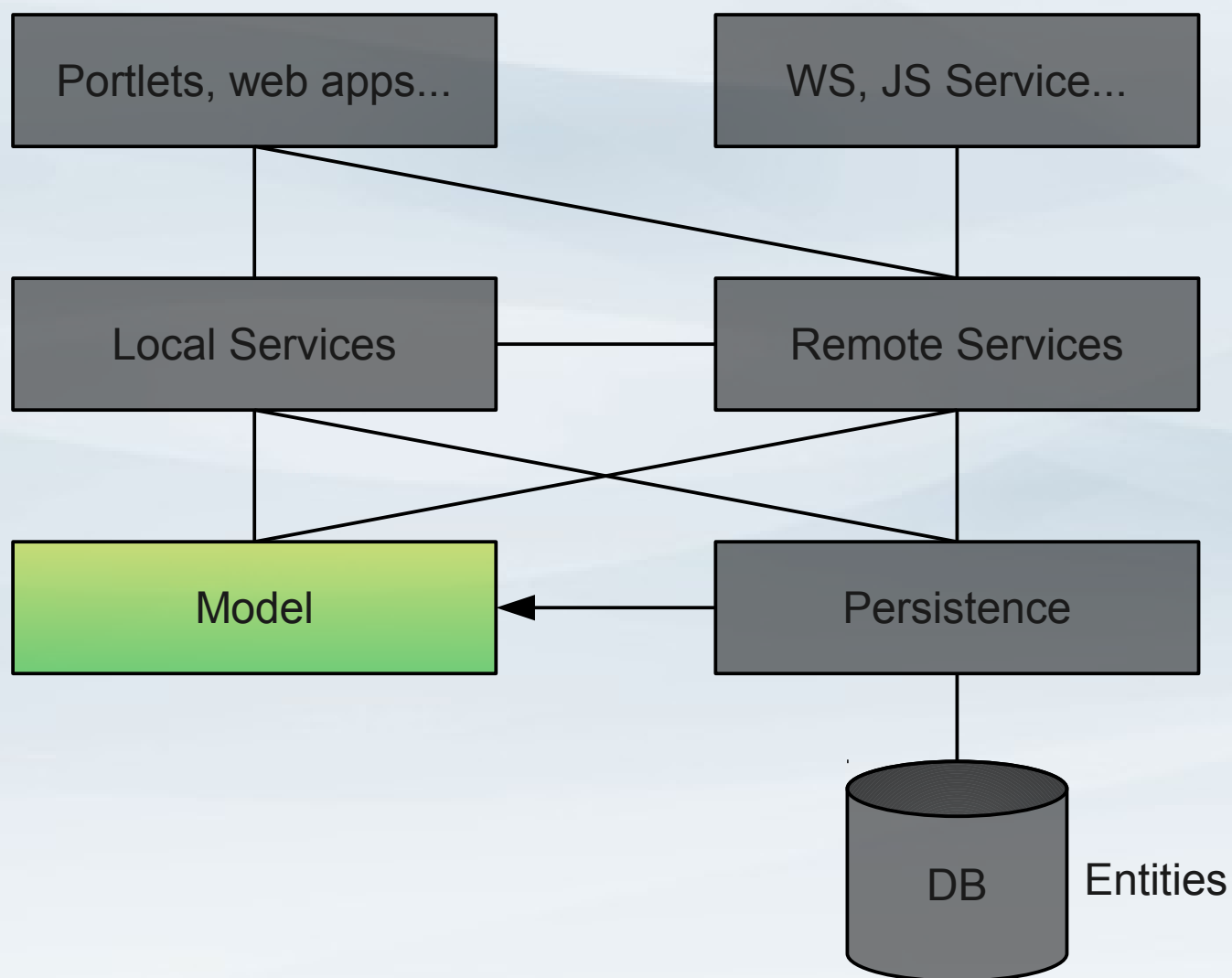
Service Builder
&
Liferay Services



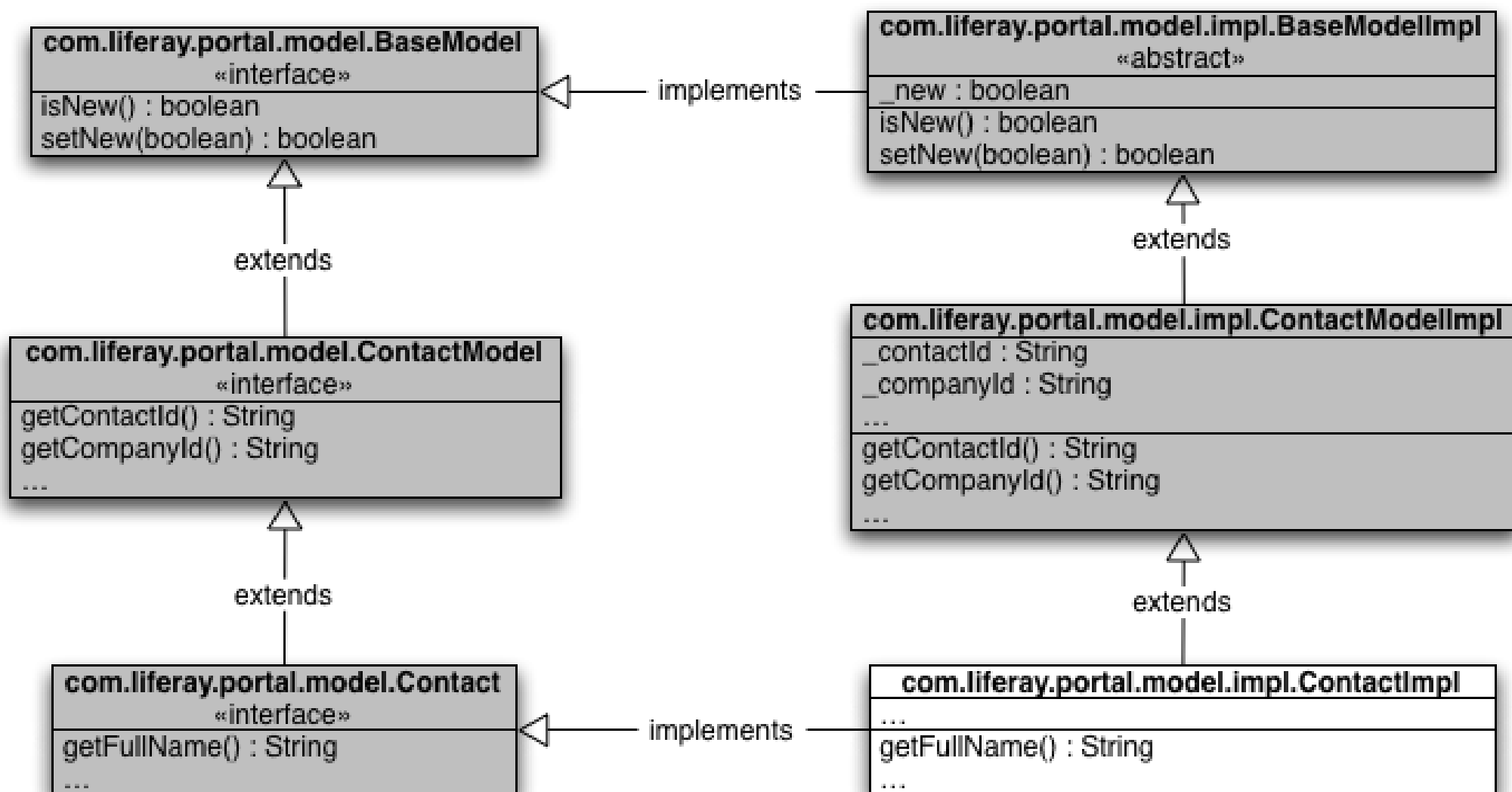
- K čemu slouží?
 - Automatické vytváření tříd a rozhraní
 - Odstranění vysoké režeie při psaní
 - Zachování jednotnosti stylu a architektury napříč celým portálem
- Generuje třídy a rozhraní pro
 - Model
 - DAO a persistenci
 - Služby
 - Util třídy pro volání služeb

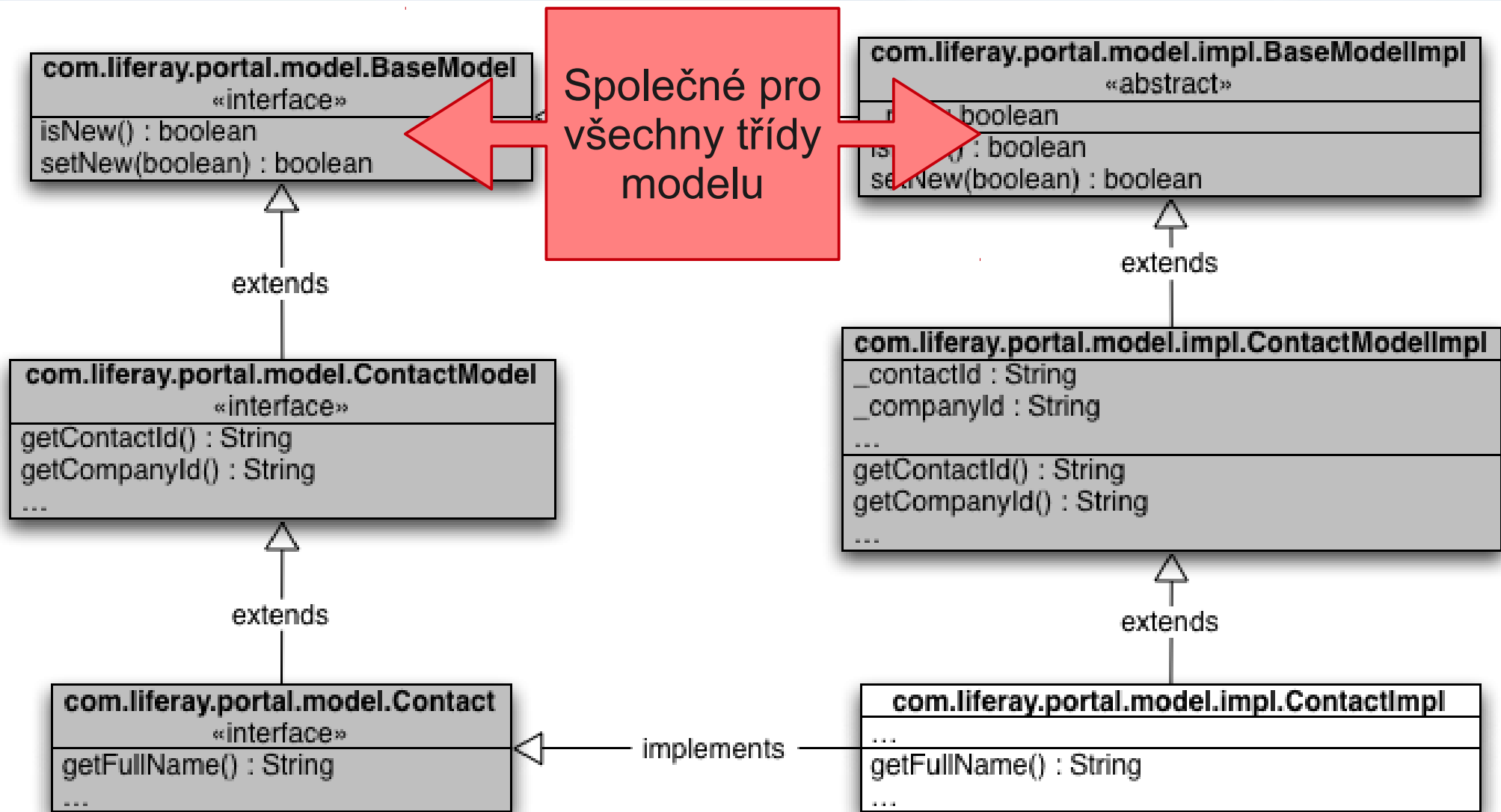
- Konfigurace pomocí service.xml
 - http://www.liferay.com/dtd/liferay-service-builder_6_0_0.dtd
- V konfiguraci lze nastavit
 - Informace o službě
 - Entity (tabulky)
 - Atributy entity (sloupce tabulek)
 - Řazení
 - Indexování pro vyhledávání
 - Výjimky
 - Transakce

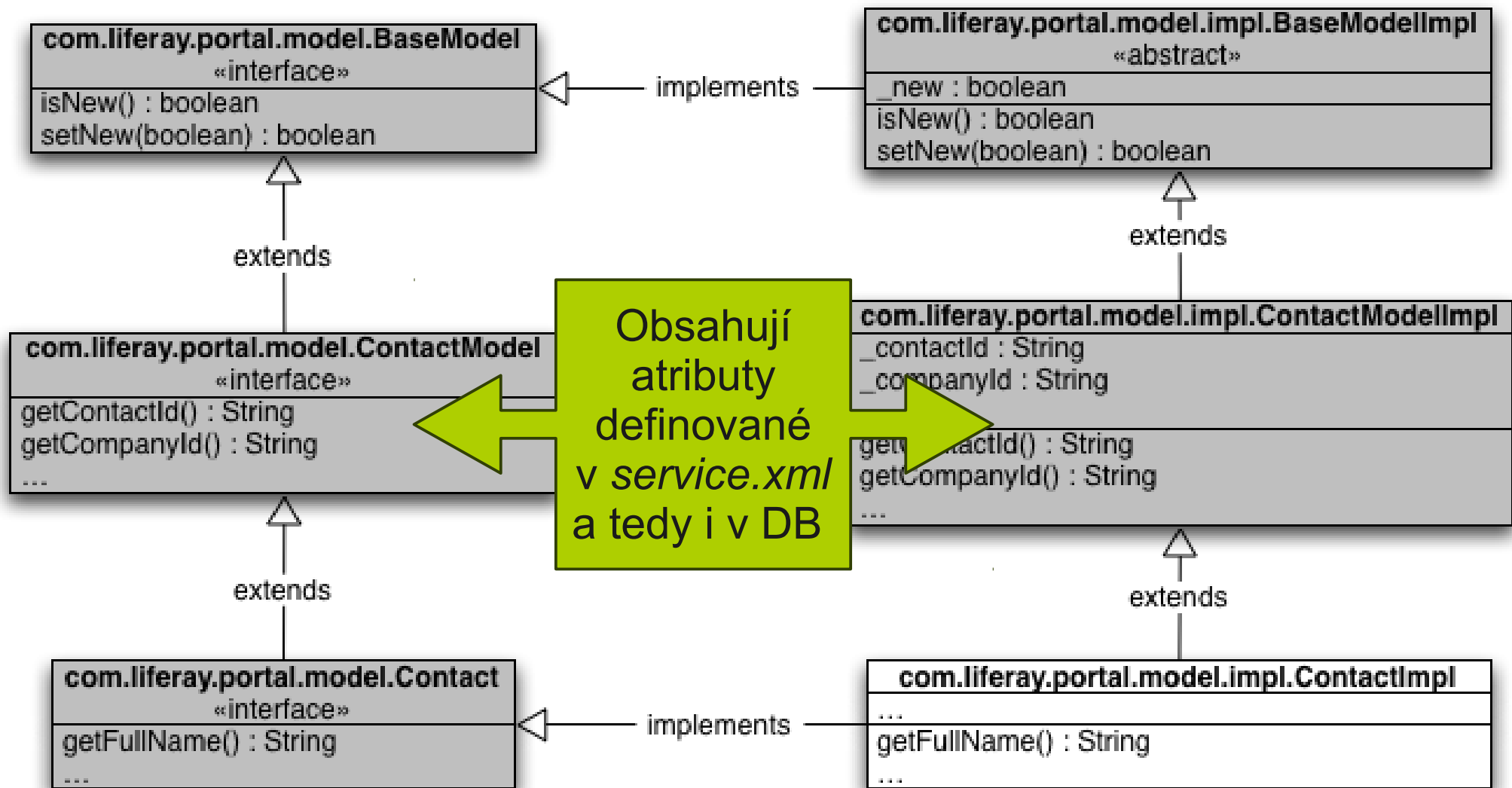
- Service Builder nepoužívá vzdálené klíče
- Service Builder nepoužívá DTO objekty
- Service Builder se využívá k generování
 - Všech servisních tříd v Liferay
 - Uživatelé
 - Organizace
 - Oprávnění
 - ...
 - Všech servisních tříd pro Liferay portlety
 - Blogy
 - Message Boards
 - atd.

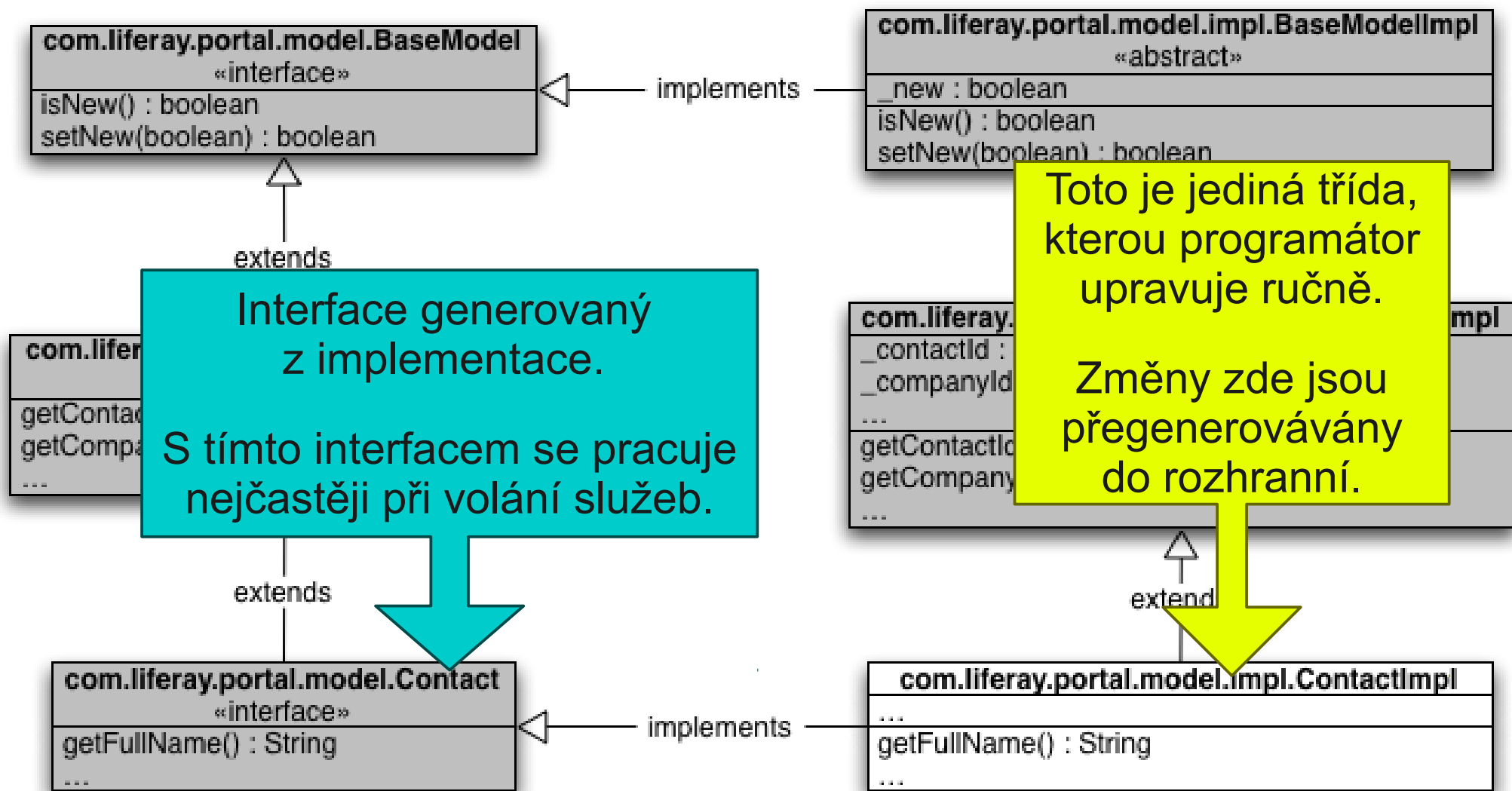


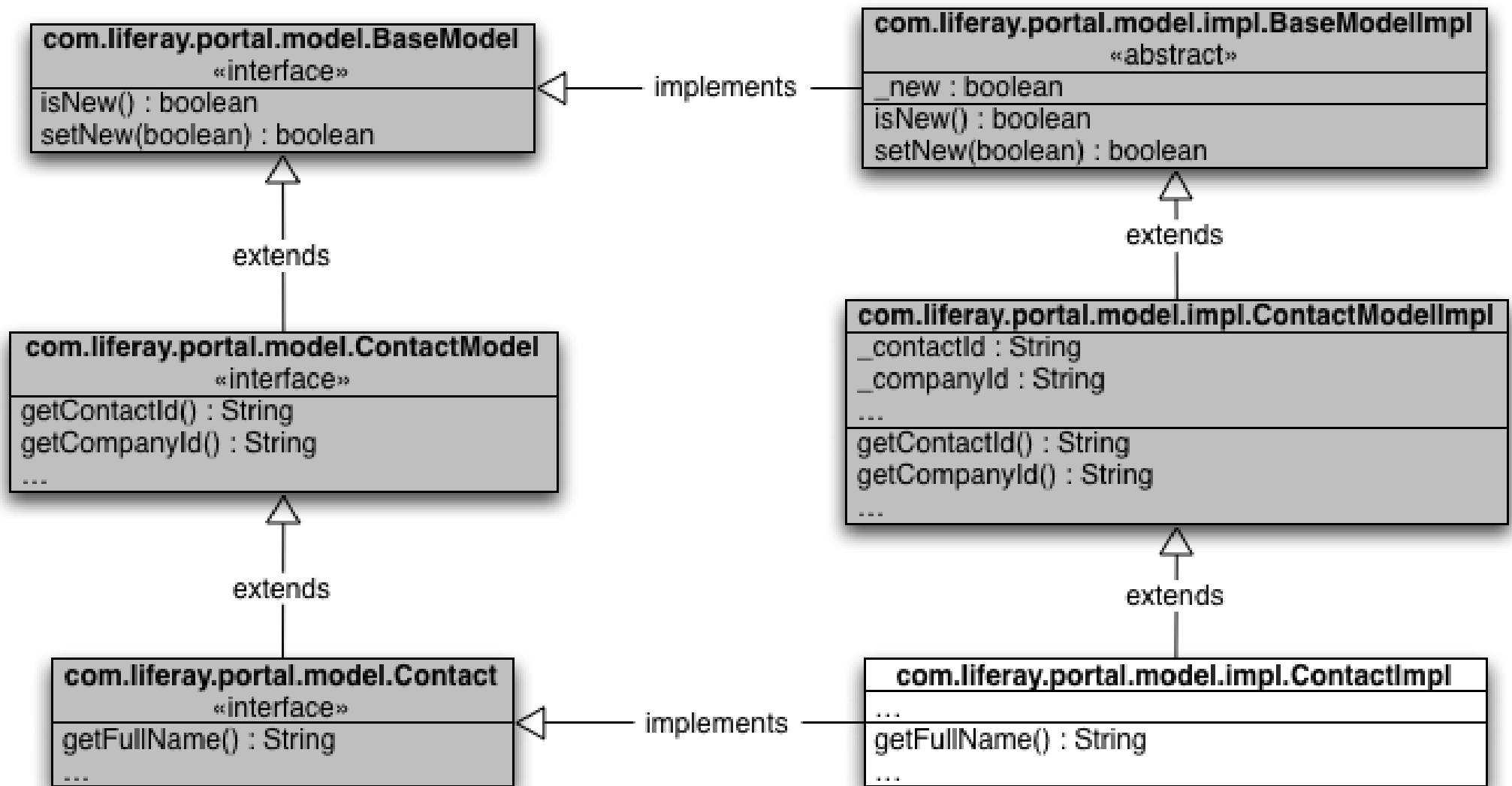
- Reprezentuje datový model aplikace
- Odráží přesně data v DB

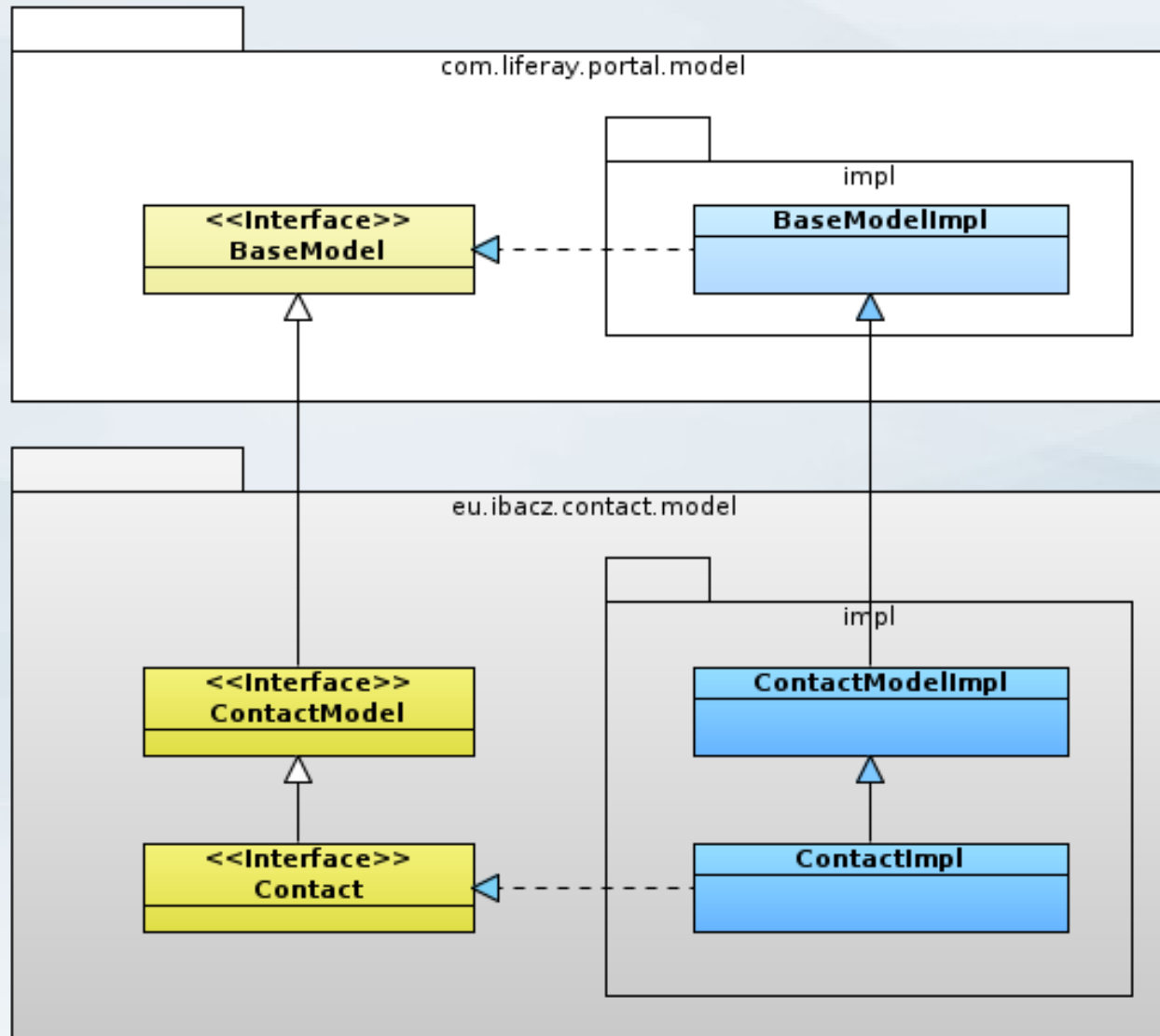


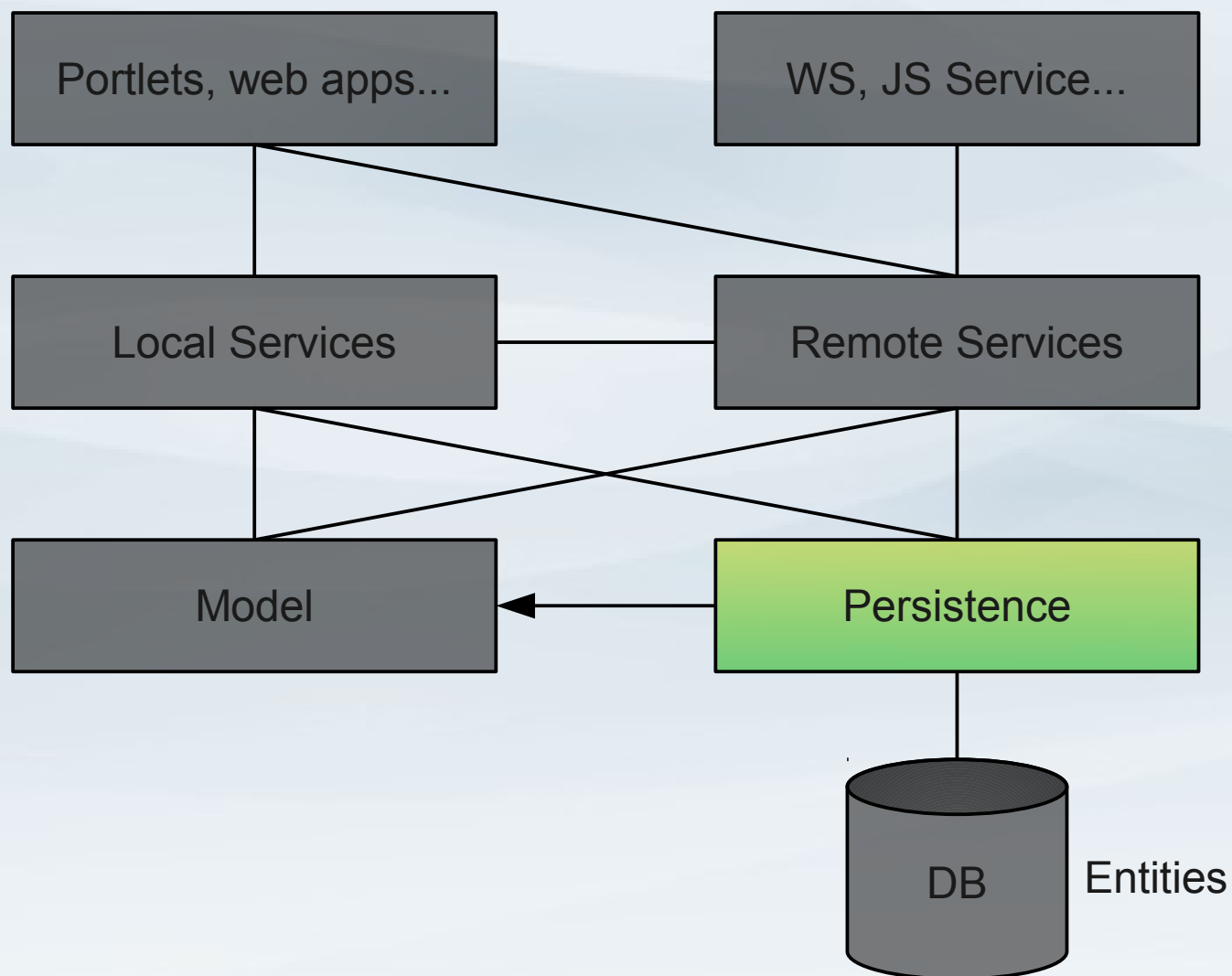




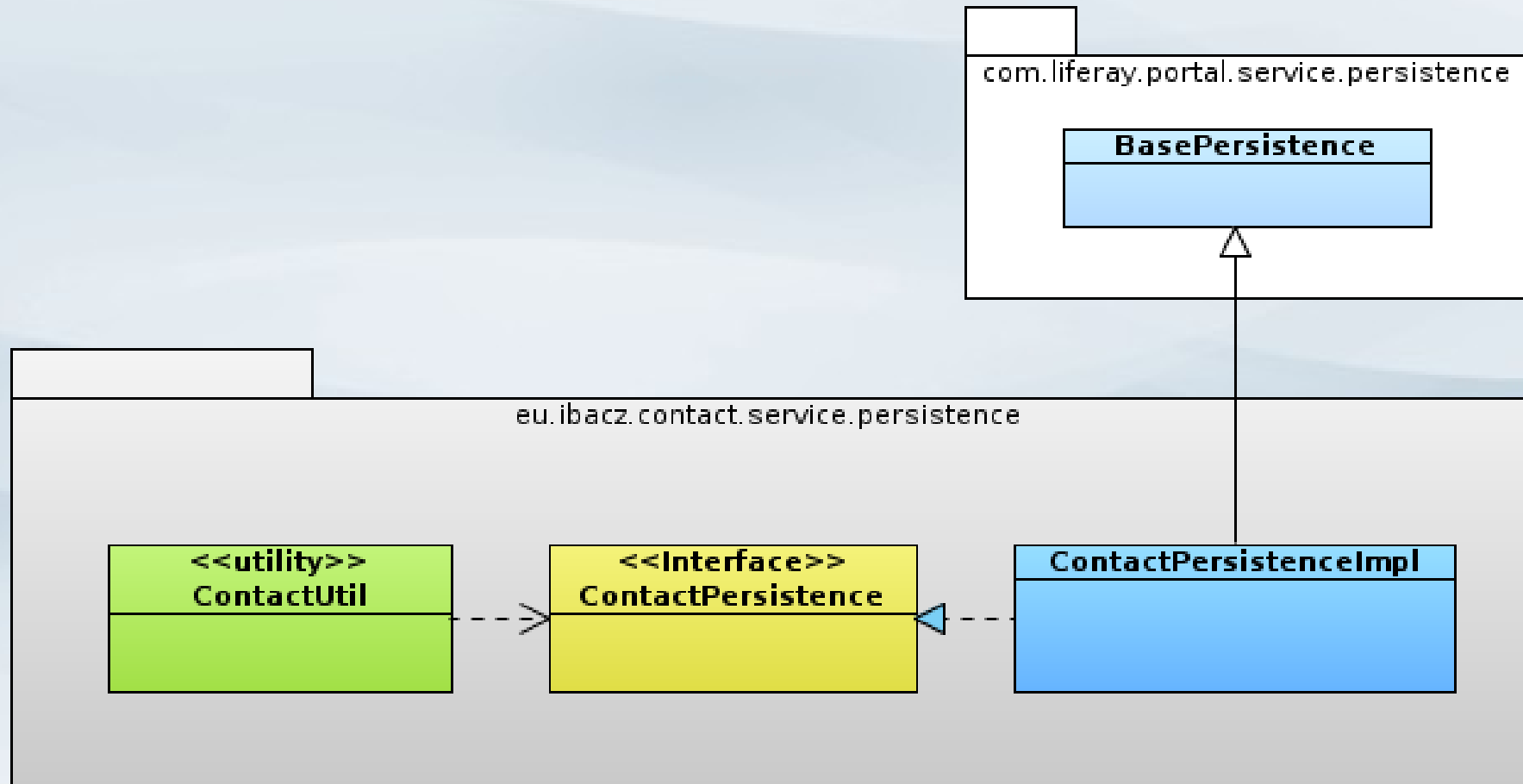


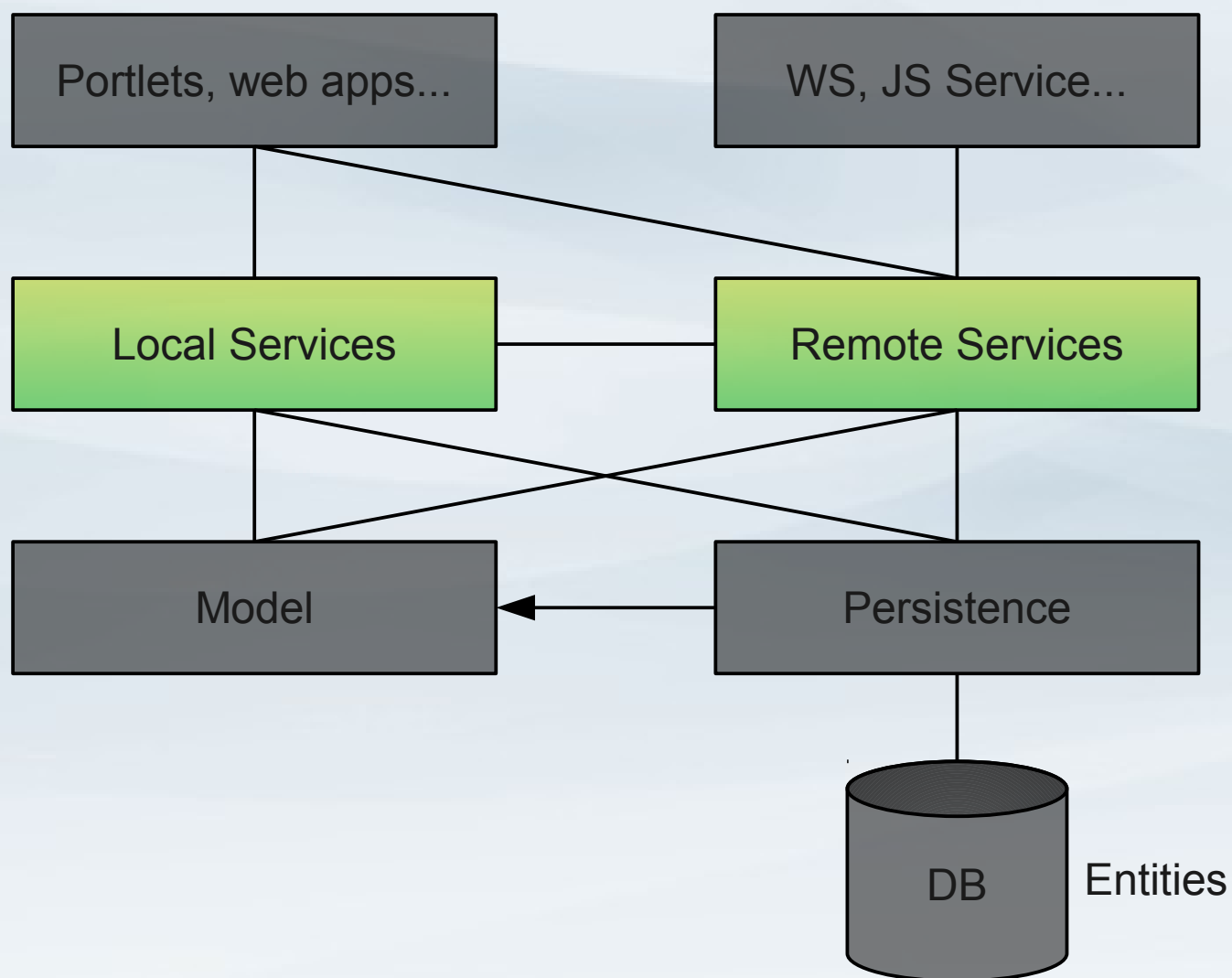




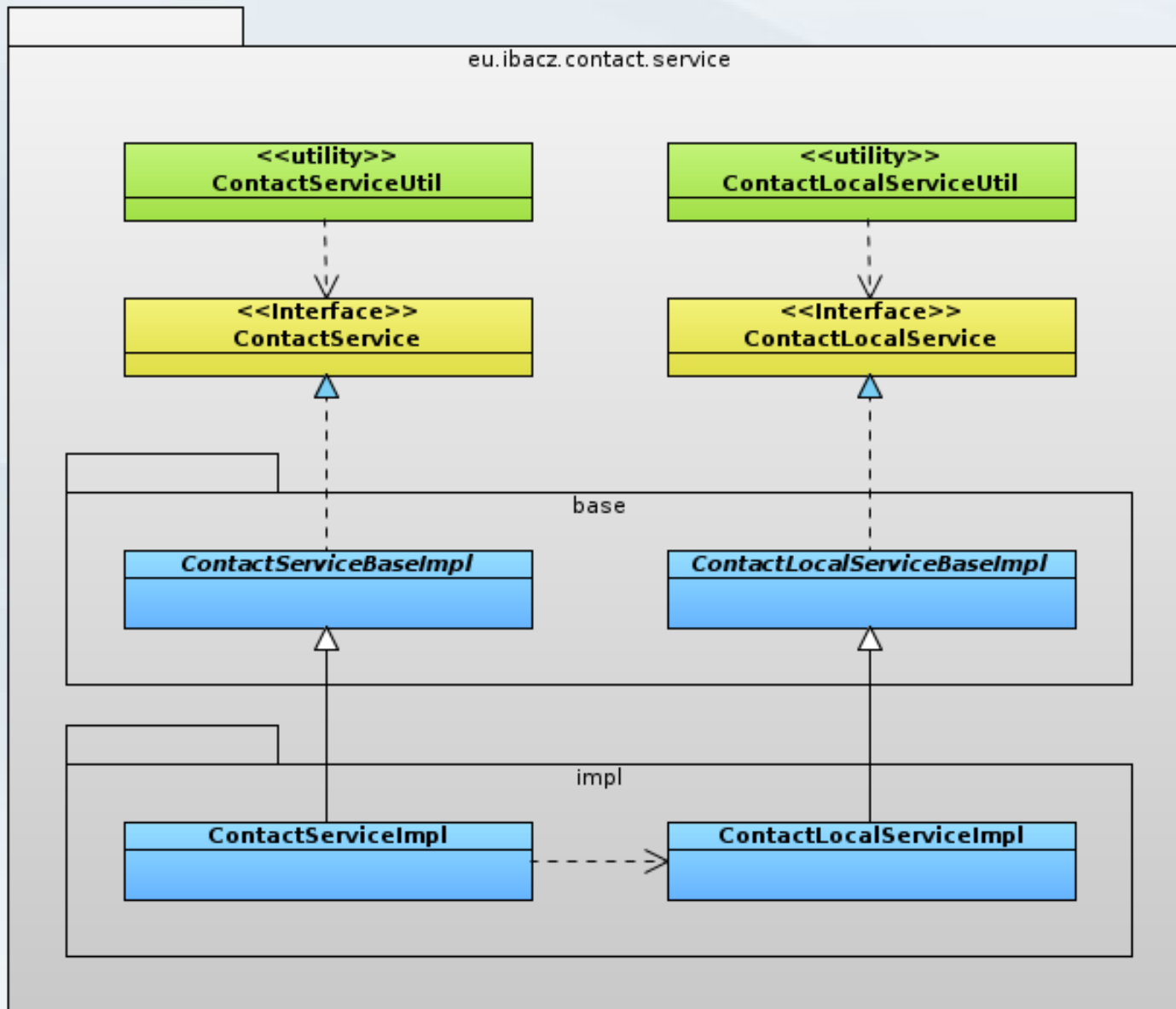


- Postavena nad Hibernate
- Zajišťuje
- Komunikaci s DB přes sdílenou Liferay Hibernate session
- CRUD akce
- Vyhledávání entit podle definovaných kritérií
- Většinou plně generována Service Builderem,
- Netřeba cokoliv implementovat – při opětovném přegenerování tříd jsou změny zahozeny

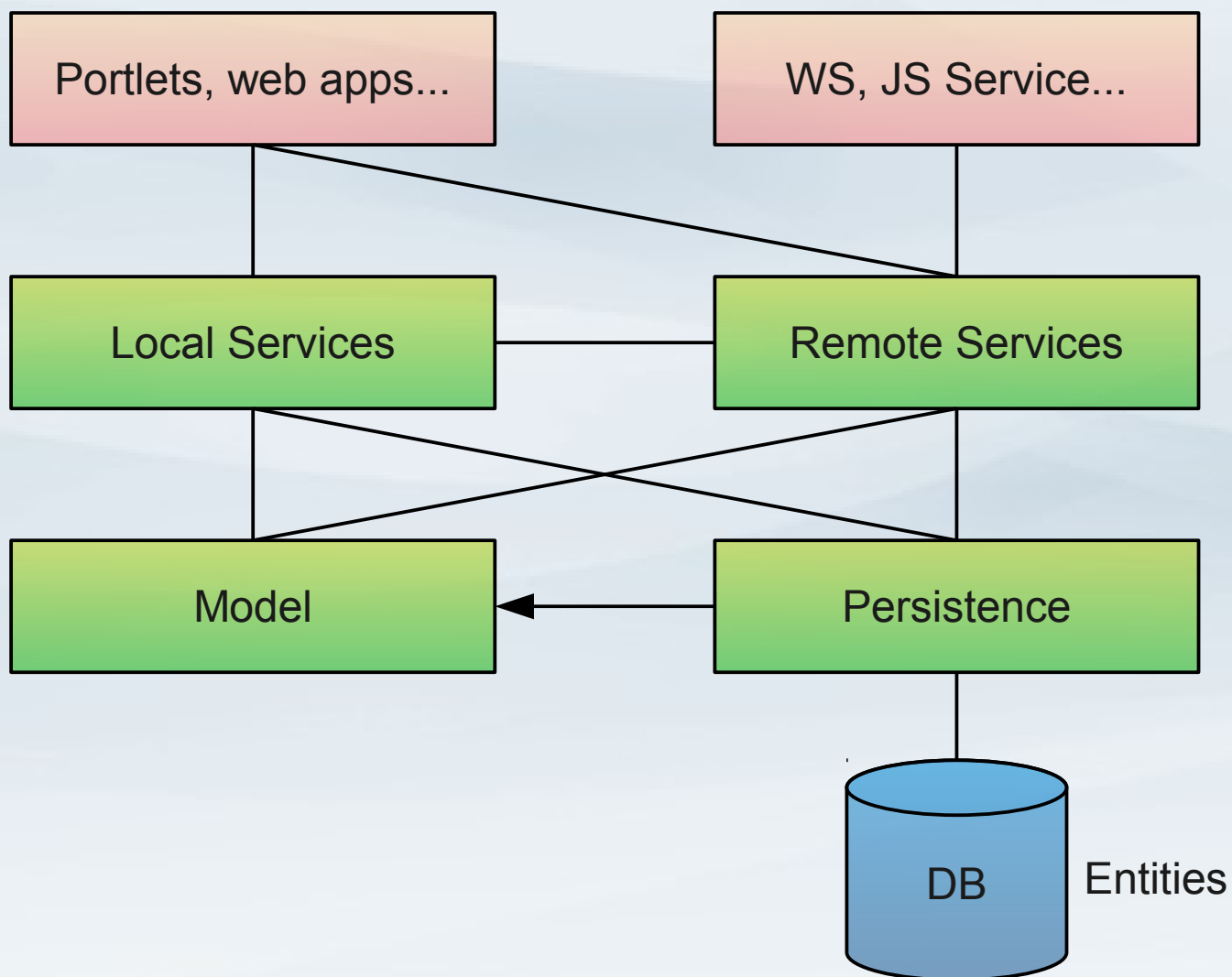




- Obsahují business logiku
- Většinou jsou k dispozici 2 verze služeb
 - **Local**
 - Většinou obsahuje přímo logiku
 - Neví, který uživatel metody volá a jestli je volat může
 - Pokud toto není přímo součástí aplikační logiky a jako argument metody
 - **Remote**
 - Většinou tvoří fasádu k local service
 - Musí řešit autorizaci k volání daných metod (v kontextu volání jsou k dispozici informace o aktuálním uživateli)
 - Bývají vystaveny jako webové služby, JSON apod.



- Nutné implementovat
 - XYLocalServiceImpl
 - případně XYServiceImpl
- Pro změnu rozhraní stačí přidat/změnit/odebrat public metodu v -ServiceImpl a přegenerovat pomocí ServiceBuilderu
- Volání metod se provádí nejčastěji pomocí Util tříd a jejich statických metod.
 - Portálové services jsou k dispozici ve všech aplikacích nasazených v aplikačním serveru s portálem (portal-service.jar je ve sdíleném classloaderu)



Portálové Java API

- Java API
 - Definováno pomocí rozhraní (*portal-service*)
 - Implementace (*portal-impl*)
 - Dostupné pomocí Util tříd a statických metod v nich
- Dělí se na 2 části
 - Jádro portálu
 - Doplnkové portlety

Entitní služby

- Služby obalující entity portálu
 - *com.liferay.portal.service*
- CompanyLocalService
- GroupLocalService
- UserLocalService
- ContactLocalService
- OrganizationLocalService
- LayoutLocalService
- PortletPreferencesLocalService
- ResourceLocalService
- RoleLocalService
- TeamLocalService
- UserGroupLocalService
- UserGroupRoleLocalService
- *com.liferay.counter.service.*
CounterLocalService
- A další

Liferay Logging

- Výhody
 - Možnost nastavit úroveň logování přímo z Control panelu
 - Jednoduchý interface založený na Log4j
- Nevýhody
 - Nedá se oddělit log naší aplikace a Liferay (ani prefixem)
 - Nemáme přímo kontrolu nad tím, jak se loguje

```
import com.liferay.portal.kernel.log.Log;
import com.liferay.portal.kernel.log.LogFactoryUtil;
...
public class Foo {

    private static final Log _log = LogFactoryUtil.getLog(Foo.class);

    public void method(...) {
        ...
        if(_log.isDebugEnabled()) {
            _log.debug("Toto "+je+" velice narocny"+foo.foo()+
                " text na sestaveni "+f+
                " a je zbytecne, aby se provadel vzdy,"+neco+
                " i pri vyplem logovani.");
        }
        ...
        _log.info("A toto je bezna jednoduchucha hlaska.");
    }
}
```

Friendly URLs

- Termíny
 - Friendly URL stránky
 - Oddělovač `/-/`
 - Friendly URL portletu
- Pro podporu Friendly URL v portletu musíme:
 - Zdefinovat generickou implementaci `DefaultFriendlyURLMapper` v deskriptoru *liferay-portlet.xml*
 - Zdefinovat a vytvořit *friendly-url-routes.xml*

Zadefinování v *liferay-portlet.xml*

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE liferay-portlet-app PUBLIC "-//Liferay//DTD Portlet Application ...
<liferay-portlet-app>
  <portlet>
    <portlet-name>HelloPortlet</portlet-name>
    <friendly-url-mapper-class>
      com.liferay.portal.kernel.portlet.DefaultFriendlyURLMapper
    </friendly-url-mapper-class>
    <friendly-url-mapping>muj-portlet</friendly-url-mapping>
    <friendly-url-routes>
      eu/ibacz/projekt/portlets/mujportlet/friendly-url-routes.xml
    </friendly-url-routes>
    <instanceable>>false</instanceable>
  </portlet>
</liferay-portlet-app>
```

```
<?xml version="1.0"?>
<!DOCTYPE routes PUBLIC "-//Liferay//DTD Friendly URL Routes 6.0.0//EN"
"http://www.liferay.com/dtd/liferay-friendly-url-routes_6_0_0.dtd">
<routes>
  <route>
    <pattern>/{instanceId}/view/{primaryKeyId:\d+}</pattern>
    <overridden-parameter name="p_p_lifecycle">0</overridden-parameter>
  </route>
  <route>
    <pattern>/{instanceId}/edit/{primaryKeyId:\d+}/{name}</pattern>
    <overridden-parameter name="p_p_lifecycle">1</overridden-parameter>
  </route>
  <route>
    <pattern>/{instanceId}/pdf/{primaryKeyId:\d+}/{name}</pattern>
    <overridden-parameter name="p_p_lifecycle">2</overridden-parameter>
  </route>
</routes>
```

Mail API

```
InternetAddress from = new InternetAddress(  
    fromAddress, fromName);  
  
InternetAddress to = new InternetAddress(  
    user.getEmailAddress(), user.getFullName());  
  
String curSubject = "Message subject";  
String curBody = "...";  
  
MailMessage message = new MailMessage(  
    from, to, curSubject, curBody, htmlFormat);  
  
MailServiceUtil.sendEmail(message);
```

Často používané třídy v portálu Liferay

Mnoho užitečných funkcí pro práci s portálem

- Získání Company z požadavku (HttpReq., PortletReq.)
- Získání mnoha URL ke stránkám
- Získání originálního HttpServletRequest
 - před posláním do portletu Liferay požadavek obalí
- Globální cesty
- Práce s portlety
- Získání aktuálního uživatele z požadavku
- CDN
- Mnoho dalšího

- Je v portálu nositelem kontextových informací v cyklu request – response
- Informace vázané k requestu
 - User
 - Company
 - Layout
 - URLs
 - a mnoho dalšího

```
import com.liferay.portal.theme.ThemeDisplay;  
import com.liferay.portal.util.WebKeys;
```

```
...
```

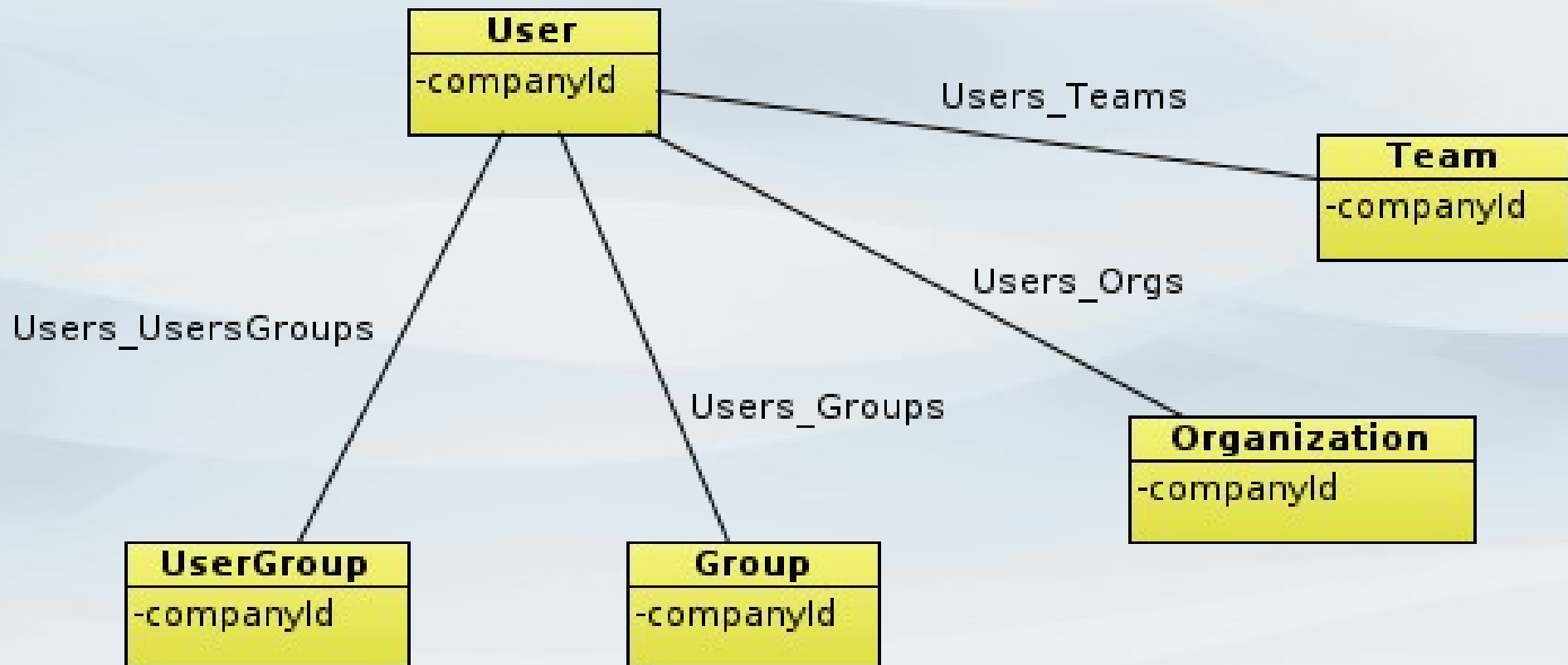
```
ThemeDisplay themeDisplay =  
    (ThemeDisplay)request.getAttribute(  
        WebKeys.THEME_DISPLAY);
```


Permissions

- Role Based Access Control
- Pojmy
 - **Subjekt operace**
 - většinou uživatel
 - **Akce**
 - např. View, Edit, Configure apod.
 - **Oprávnění**
 - oprávnění provést nějakou akci
 - např. uklízet
 - **Role**
 - souhrn oprávnění
 - Administrátor, HR manager, uklízečka

- Typy rolí (Regular, Community, Organization, Team)
- Resources
 - Liferay rozšiřuje RBAC o možnost definovat, nad jakým zdrojem je akce uplatňována
 - Jedná se tedy o **objekt autorizace**
 - Např. uklízečka už neuklízí obecně, ale může uklízet třeba jen toalety
- Většina zdrojů je v kontextu komunity / organizace
 - Výjimku tvoří systémové zdroje jako jsou role, skupina apod.
 - Na zdroj v komunitě resp. organizaci pak platí oprávnění z
 - Regular Roles
 - Community Roles resp. Organization Roles
 - Teams

Organizace uživatelů do skupin



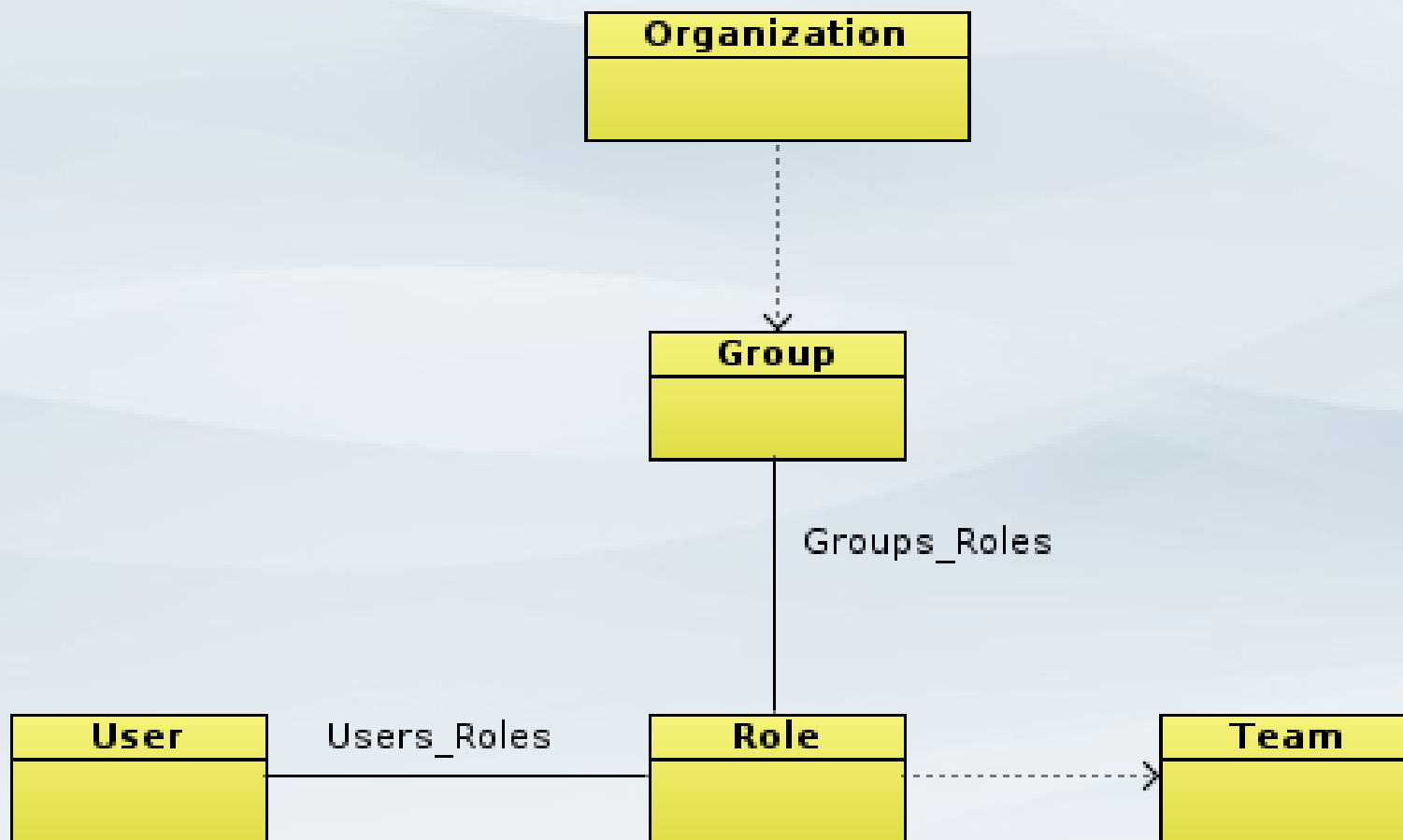
```
UserLocalServiceUtil  
    .addGroupUsers(groupId, userIds);
```

```
UserLocalServiceUtil  
    .addOrganizationUsers(organizationId, userIds);
```

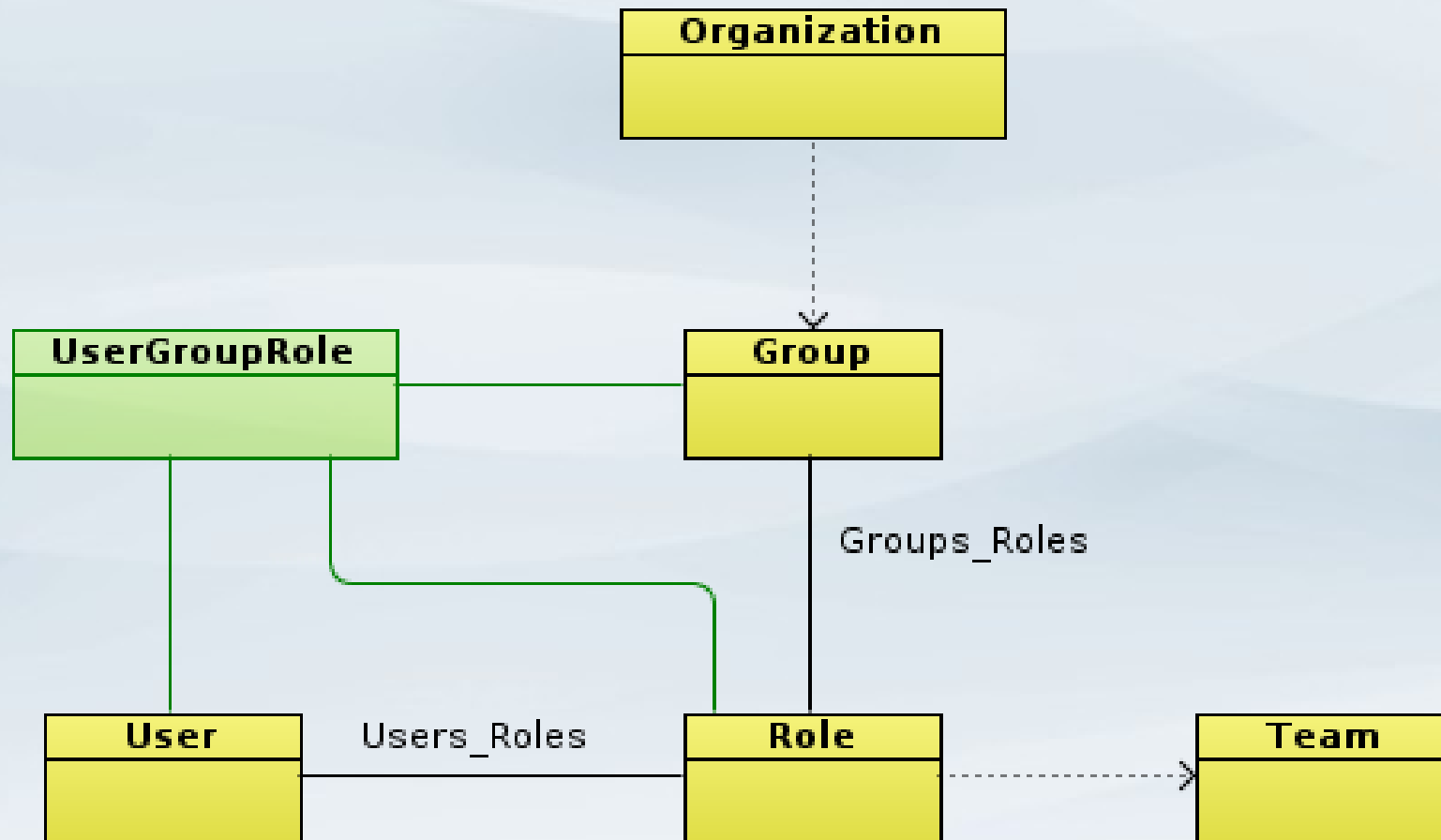
```
UserLocalServiceUtil  
    .addUserGroupUsers(userGroupId, userIds);
```

```
UserLocalServiceUtil  
    .addTeamUsers(teamId, userIds);
```

```
UserLocalServiceUtil  
    .unsetGroupUsers(userGroupId, userIds);
```



```
String roleName = RoleConstants.ADMINISTRATOR;  
  
Role role = RoleLocalServiceUtil  
    .getRole(companyId, roleName);  
  
UserLocalServiceUtil  
    .addRoleUsers(role.getRoleId(), userIds);
```




```
String communityRoleName =  
    RoleConstants.COMMUNITY_ADMINISTRATOR;
```

```
Role communityRole = RoleLocalServiceUtil  
    .getRole(companyId, communityRoleName);
```

```
String communityName = "My Community";  
// put something smart ^^^^here^^^ ; -)
```

```
Group community = GroupLocalServiceUtil  
    .getGroup(companyId, communityName);
```

```
UserGroupRoleLocalServiceUtil.addUserGroupRoles(userIds,  
    community.getGroupId(),  
    communityRole.getRoleId());
```

```
String orgRoleName = RoleConstants
    .ORGANIZATION_ADMINISTRATOR;

Role orgRole = RoleLocalServiceUtil
    .getRole(companyId, orgRoleName);

String orgName = "My Lovely Organization";

Organization organization = OrganizationLocalServiceUtil
    .getOrganization(companyId, orgName);

long orgGroupId = organization.getGroup().getGroupId();

UserGroupRoleLocalServiceUtil.addUserGroupRoles(
    UserIds, orgGroupId, orgRole.getRoleId());
```

- Používá se pro ověření práv na konkrétní entitě
- Různé implementace
 - *SimplePermissionChecker*
 - Vrací *TRUE* pro přihlášeného uživatele
 - *AdvancedPermissionChecker*
 - Posbírá všechny role přiřazené uživateli v rámci portálu
 - Kontroluje, zda některá z nich nemá právo provést požadovanou akci
 - viz. *PermissionLocalServiceImpl*
- Možnost vytvořit vlastní implementaci
 - *portal.properties* → `permissions.checker=...`

- Řízení práv z Liferay lze využít i u portletů mimo portál
 - Řízení práv nad custom portlety
 - Řízení práv nad custom entitami
- Co je zapotřebí udělat?
 - Vytvořit *portlet.properties* s umístěním XML konfigurace akcí
 - Vytvořit XML deskriptor s konfigurací akcí a definovat
 - Portlety
 - Entity
 - Povolené akce
 - Výchozí hodnoty

- Soubor musí
 - být mezi zdrojovými soubory
 - obsahovat klíč *resource.actions.configs*

resource.actions.configs=resource-actions/my-resource-actions.xml

```
<?xml version="1.0"?>
<resource-action-mapping>
  <portlet-resource>
    ...
  </portlet-resource>
  <portlet-resource>
    ...
  </portlet-resource>

  <model-resource>
    ...
  </model-resource>
  <model-resource>
    ...
  </model-resource>
</resource-action-mapping>
```

```
<?xml version="1.0"?>
<resource-action-mapping>
  <portlet-resource>
    <portlet-name>my-portlet</portlet-name>
    <permissions>
      <supports>
        <action-key>VIEW</action-key>
        <action-key>CONFIGURATION</action-key>
      </supports>
      <community-defaults />
      <guest-defaults />
      <guest-unsupported>
        <!--
        <action-key>VIEW</action-key>
        <action-key>CONFIGURATION</action-key>
        -->
      </guest-unsupported>
    </permissions>
  </portlet-resource>
  ...

```

```
...  
<model-resource>  
  <model-name>eu.ibacz.portlet.model.MyObject</model-name>  
  <portlet-ref>  
    <portlet-name>my-portlet</portlet-name>  
  </portlet-ref>  
  <permissions>  
    <supports>  
      <action-key>DELETE</action-key>  
      <action-key>UPDATE</action-key>  
      <action-key>VIEW</action-key>  
    </supports>  
    <community-defaults/>  
    <guest-defaults />  
    <guest-unsupported>  
      <action-key>VIEW</action-key>  
      <action-key>DELETE</action-key>  
      <action-key>UPDATE</action-key>  
    </guest-unsupported>  
  </permissions>  
</model-resource>  
</resource-action-mapping>
```


- Pro každou entitu → resource
 - Portál práva nastavuje a ověřuje práva oproti resource

```
String modelName = MyObject.class.getName();  
long entityPK = myObject.getId();
```

```
// create resource when creating entity
```

```
ResourceLocalServiceUtil.addResources(companyId, groupId,  
    userId, modelName, entityPK, true, true, true);
```

```
// delete resource when removing entity
```

```
ResourceLocalServiceUtil.deleteResource(companyId, modelName,  
    ResourceConstants.SCOPE_INDIVIDUAL, entityPK);
```

```
String modelName = MyObject.class.getName();  
long entityPK = myObject.getId();  
String actionName = "UPDATE";
```

```
PermissionChecker permissionChecker =  
    PermissionThreadLocal.getPermissionChecker();
```

```
boolean hasPermission =  
    permissionChecker.hasPermission(groupId, modelName,  
    primaryKey, actionName);
```

Expando API

- Expando slouží k vytváření a ukládání dynamických dat
 - Custom atributy
 - Web form
- Výhody
 - Vysoká flexibilita a rychlost implementace
 - Snadné použití
- Nevýhody
 - Výkonnost
 - Špatná udržitelnost

- Každý atribut má
 - **Key**
 - Bez mezer, možno použít podtržítka, pomlčku nebo CamelCase
 - **Type**
 - Nelze po vytvoření změnit
 - **Hidden**
 - Hodnoty se standardně nezobrazují v UI
 - **Searchable**
 - Definuje, zda se hodnoty indexují a je možné je pak prohledávat

```
user.getExpandoBridge()  
    .setAttribute("company-name", "My Company");
```

```
String companyName = (String) user.getExpandoBridge()  
    .getAttribute("company-name");
```

```
public Hits search(  
    long companyId, String keywords, Boolean active,  
    LinkedHashMap<String, Object> params,  
    int start, int end, Sort sort)  
    throws SystemException;
```

```
LinkedHashMap userParams = new LinkedHashMap();
userParams.put("company-name", "My Company");

boolean asc = true;
Sort sort = new Sort("lastName", Sort.STRING_TYPE, asc);

Hits hits = UserLocalServiceUtil.search(companyId, null, true,
    userParams, 0, 20, sort);

List<User> users = new ArrayList<User>();
List<Document> hitsList = hits.toList();
for (Document doc : hitsList) {
    long userId = GetterUtil.getLong(doc.get(Field.USER_ID));
    users.add(UserLocalServiceUtil.getUserById(userId));
}
```


Social APIs

- Social Activity API
 - Zaznamenávání aktivit uživatelů v portálu
 - Tyto aktivity lze poté zobrazovat v portletech
 - V profilu uživatele
 - Na stránkách komunit či organizací
 - Jako aktivity kamarádů apod.
- Social Equity API
 - Ohodnocení chování uživatelů v portálu
 - Identifikace aktivních a užitečných uživatelů

- Skládá se ze dvou částí
 - Zaznamenávání aktivit
 - Voláním metod *SocialActivityLocalService*
 - ***addUniqueActivity()*** – při vytváření záznamu
 - ***deleteActivities()*** – při mazání položky
 - Pro aktivitu se zaznamenává uživatel, group, class, druh aktivity a další
 - Interpretace aktivit
 - Třída zaregistrovaná v *liferay-portlet.xml*
`<social-activity-interpreter-class>`
 - Vytváří ze *SocialActivity* textovou reprezentaci

- Provádí se zaznamenávání aktivit
- Aktivity jsou různě ohodnocené
- Hodnoty se mění v průběhu času
- Sumy se přepočítávají pravidelně v noci nebo manuálně
- Nastavení koeficientů v Control Panelu

- Zaznamenávání akcí

```
SocialEquityLogLocalServiceUtil  
    .addEquityLogs(userId,  
                    BlogsEntry.class.getName(),  
                    entryId, ActionKeys.ADD_ENTRY);
```

- Výchozí hodnoty pro výpočty → resource actions XML

```
<resource-action-mapping>
...
<model-resource>
  <model-name>com.liferay.portlet.blogs.model.BlogsEntry</model-name>
  <portlet-ref>
    <portlet-name>33</portlet-name>
  </portlet-ref>
  <permissions>
    ...
  </permissions>
  <social-equity>
    <social-equity-mapping>
      <action-key>ADD_ENTRY</action-key>
      <information-value>10</information-value>
      <information-lifespan>365</information-lifespan>
      <participation-value>5</participation-value>
      <participation-lifespan>365</participation-lifespan>
    </social-equity-mapping>
    ...
  </social-equity>
</model-resource>
</resource-action-mapping>
```

Assets

- Původně vyvinut pro tagování různých typů obsahu
- Využitím frameworku získáme
 - Možnost tagovat a kategorizovat obsah
 - Spravovat tagy z Control panelu
 - Přehled o počtu zobrazení assetů
 - Zobrazení pomocí Asset Publisher portletu
 - Workflow

- Použití = synchronizace naší entity s assetem
 - Vytváření
 - Editace
 - Mazání
- Vše jsou volání *AssetEntryLocalService*
 - *updateEntry()*
 - *deleteEntry()*
 - a další

- ***userId*** – ID uživatele, který vytvořil obsah
- ***groupid*** – scope, v němž byl obsah vytvořen (může být 0)
- ***className*** – nejlépe java class
- ***classPK*** – jedinečný identifikátor v rámci className
- ***assetCategoryIds* a *assetTagNames*** – seznam kategorií a tagů, které byly vybrány pro asset
- ***visible*** – má se zobrazovat v Asset Publisher portletu
- ***title*, *description* a *summary*** – popisující pole
- ***publishDate* a *expirationDate*** – kdy má Asset Publisher položku zobrazovat
- Další pole mohou být *null*

- Zobrazení
 - `<liferay-ui:asset-tags-selector ... />`
 - `<liferay-ui:asset-categories-selector ... />`
- Editace
 - `<liferay-ui:asset-tags-summary ... />`
 - `<liferay-ui:asset-categories-summary ... />`

- Zobrazení v Asset Publisher portletu
 - *AssetRendererFactory*
 - Umí získat asset podle
 - PK
 - groupId a řetězce z pěkné URL
 - Umí vytvořit URL pro přidání nového assetu
 - Často se vytváří rozšířením *BaseAssetRendererFactory*
 - *AssetRenderer*
 - Vrací metadata o konkrétním assetu
 - Kontroluje práva na zobrazení editaci assetu
 - Zajišťuje vykreslení obsahu (plný, abstrakt)
 - Často se vytváří rozšířením *BaseAssetRenderer*

Captcha

- Ověření uživatele kódem v obrázku
- Sestává se ze 3 kroků
 - Vygenerování obrázku (uložení informace do session)
 - Zobrazení obrázku uživateli
 - Validace opsaného kódu

```
import com.liferay.portal.kernel.captcha.CaptchaUtil;

...

@Override
public void serveResource(ResourceRequest resourceRequest,
    ResourceResponse resourceResponse) throws IOException,
    PortletException {
    try {
        CaptchaUtil.serveImage(resourceRequest, resourceResponse);
    } catch (Exception e) {
        log.error(e);
    }
}
```

- Metoda *serveImage* zařídí
 - vygenerování obrázku
 - přenos do response včetně potřebných hlaviček
 - Vložení potřebných informací do session

```
<portlet:resourceURL var="captchaURL"/>  
<liferay-ui:captcha url="<%= captchaURL %>" />
```



```
import com.liferay.portal.kernel.captcha.CaptchaUtil;
```

```
...
```

```
CaptchaUtil.check(actionRequest);
```

- V případě, že uživatel zadá kód špatně, vypadne *CaptchaException*

Search

- Každý portlet může nabízet rozhraní pro vyhledávání a indexaci
- Definice v deskriptoru *liferay-portlet.xml*
 - `<open-search-class />`
 - Pro komunikaci se využívá formát OpenSearch
 - `<indexer-class />`
- Liferay Search portlet
 - Vezme všechny portlety, které poskytují hledání pomocí OpenSearch
 - Všem zašle dotaz pro vyhledání
 - Výsledky parsuje a zobrazí
 - Viz. `/html/portlet/search/search.jsp`

- Implementace rozhraní *Indexer*, nejčastěji pak rozšíření abstraktní třídy *BaseIndexer*
- Viz. *BlogsIndexer*
 - Převádí *BlogsEntry* na search *Document* a ten pak posílá k indexaci
 - Ruční udržování aktuálního indexu z *BlogsEntryLocalService*
- Záznamy je poté možné vyhledat pomocí *SearchEngineUtil.search(...)*

- Nejjednodušší implementace pomocí *HitsOpenSearchImpl*
 - Závislost na portálových třídách, která se poměrně složitě řeší
- Možnost implementovat vlastní rozhraní *OpenSearch* a poskládat XML ručně

Co je zapotřebí rozlišovat?

- Indexace a vyhledávání
 - Může být implementována pomocí Liferay rozhraní
 - Výchozí indexing engine je Lucene, nicméně existují i další implementace, které lze snadno vyměnit
 - Lze se inspirovat v existujícím kódu Liferay
 - Může být držena zcela mimo portál (např. při integracích)
- Vyhledávání pomocí OpenSearch
 - Zajišťuje komunikaci od našeho portletu směrem k Liferay Search portletu
 - Vyhledávání samotné nemusí být postaveno na Lucene, ale je možné využít např. Dotaz do DB

Liferay Message Bus

- Sběrnice pro výměnu zpráv
 - v rámci portálu Liferay
 - mezi portletovými aplikacemi
 - jinými typy pluginů
- Výrazně jednodušší implementace než např. JMS

- Zaregistruje se cílová adresa pro zprávy (*Destination*)
 - Každá *destination* je identifikovaná jménem
 - Pro každou *destination* je možné zaregistrovat *listenery* (MessageListener)
- Klient (producent) pomocí API odešle zprávu na určitou cílovou adresu
- Sběrnice předá zprávu všem listenerům (konzument), které jsou zaregistrované pro danou cílovou adresu

- Zprávy se posílají
 - **Asynchronně** – klient odešle zprávu a ppracuje, zatímco zpráva je zpracována v jiném vlákně
 - **Synchronně** – klient odešle zprávu a jeho vlákno čeká na odpověď
- Zprávy se zpracovávají
 - **Sériově** – v jednom vlákně (SerialDestination)
 - **Paralelně** – ve více vláknech (ParallelDestination)

```
import com.liferay.portal.kernel.messaging.Destination;

...

// create destination
Destination destination = new
    SerialDestination("myapp/serialdestination1");

// destination settings done
destination.afterPropertiesSet();

// register destination
MessageBusUtil.addDestination(destination);
```

```
package eu.ibacz.myapp;

import com.liferay.portal.kernel.messaging.Message;
import com.liferay.portal.kernel.messaging.MessageListener;

public class Listener1 implements MessageListener {

    public void receive(Message message) {
        String attr1 = message.getString("attribute1");
        ...
    }

}
```

```
// directly into message bus
MessageBusUtil.registerMessageListener(
    "destinationName", myListener);

// if we still have destination
destination.register(myListener);
```

- Vytvoření objektu typu *Message*
 - Je možné nastavit atributy včetně typu *Object*
 - Pozor na classloadery
- Zaslání pomocí *MessageBusUtil*
 - *sendMessage()*
 - *sendSynchronizedMessage()*
 - Výchozí timeout je 1 sekunda

```
Message myMessage = new Message();  
myMessage.put("attribute1", "test 123");  
myMessage.put("attribute2", "test 123");  
MessageBusUtil.sendMessage(  
    "myapp/serialdestination1", myMessage);
```

Caching

- Liferay silně cachuje
 - DB entity
 - Výsledky vyhledávání
 - Portlety
 - Stránky
- Standardní implementace
 - EhCache
- Na co si dávat pozor
 - Clustering
 - Problémy se synchronizací
 - Přístupy přímo do DB
 - Optimalizace cachí
 - Bezpečnost při zapnutí
 - Výkon po vypnutí

- portal.properties
 - XML soubory s nasavením
 - Pro jednu instanci
 - Pro cluster
 - Pro Hibernate
- XML konfigurační soubory
 - Clustering
 - Výchozí nastavení pro nové cache
 - Nastavení velikostí a typů cachí

```
public static final String CACHE_NAME = MyObject.class.getName();

public MyObject getMyObject(String objectId) {
    String key = CACHE_NAME.concat(StringPool.POUND)
        .concat(objectId);

    PortalCache cache = MultiVMPoolUtil.getCache(CACHE_NAME);
    MyObject myObject = (MyObject)cache.get(key);

    if (myObject == null) {
        // expensive operation
        myObject = MyObjectLocalServiceUtil
            .prepareMyObject(objectId);

        // store object into cache
        cache.put(key, myObject);
    }

    return myObject;
}
```

Liferay Web Content

- Historický název WCM = Journal
 - Tudíž balík *com.liferay.portlet.**journal***
- Možno pracovat standardně pomocí služeb v balíku *com.liferay.portlet.journal.service*
- Standatdně WCM používá pro ukládání dat XML a pro zobrazení různé transformace
- Pro zobrazování obsahu uživatelům používat *com.liferay.portlet.journalcontent.util.**JournalContentUtil***
 - Obsahuje metody, vracející cachovaný obsah
 - Nemusí se pokaždé parsovat a zpracovávat XML data

- Využití služeb v balíku
*com.liferay.portlet.**documentlibrary**.service*
 - **Složky** – *DLFolderLocalService*
 - **Soubory** – *DLFileEntryLocalService*
 - **Linky** – *DLFileShortcutLocalService*

Plugins SDK

- Možnost vývoje a provádění změn bez nutnosti měnit přímo kódy portálu
 - Modularita
 - Znovupoužitelnost
 - Přenositelnost
- Repositáře zásuvných modulů
 - Liferay official
 - Liferay community
 - Marketplace

- Sada Ant skriptů a šablon pro generování, vývoj, sestavování a nasazování pluginů
- Bokem existuje i podpora pro Maven
 - Archetypy
 - Pluginy

Plugins se dělí dle toho, zda slouží k přidávání či změně existující funkcionality

Přidání nové funkcionality

- Portlety
- Layout Template
- Theme
- Web Application

Změna funkcionality

- Hook
- Extension Environment
- Ext Plugin
- Extlet
 - Neoficiální vyvinutý v IBA CZ
 - Open-source

- Nástroj pro definování vzhledu portálových stránek
- Pro každý web (komunitu/organizaci/...) a každou stránku lze definovat, jaké téma má používat a tudíž jak má vypadat.
- Téma definuje vzhled portálové stránky na úrovni
 - CSS stylů
 - JavaScriptů
 - Šablon
 - Záhloví, zápatí
 - Vzhled navigace (seznam stránek)
 - Vzhled portletů

- Nástroj na definování rozložení portletů na stránce
- Jedná se o doplněk tématu
- Můžeme nastavit rozložení sloupců a řádků, do kterých lze vložit portlety

- Základní stavební jednotkou portálu
- Primární způsob přidávání nové funkcionality do portálu
- Liferay podporuje
 - Portlety podle JSR-168
 - Od verze 5 i podle JSR-286
- Liferay obsahuje několik druhů *Bridge pro implementaci portletů pomocí Struts, PHP, Faces a další
- Liferay MVC portlet založený na JSP

- ServiceBuilder lze použít jako součást portletové aplikace
- Po vytvoření konfiguračního service.xml se použít
`ant build-service`
- Více viz. sekce ServiceBuilder and Liferay Services

- Portletové deskriptory
 - ***portlet.xml***
 - Standardní portletový deskriptor podle specifikace
 - ***liferay-portlet.xml***
 - Parametry portletové aplikace specifické pro Liferay
 - ***liferay-display.xml***
 - Parametry zobrazení portletů pro Liferay

Podporovaná nastavení:

- Ikona portletu
- Indexace a vyhledávání
- Časované procesy
- Vytváření pěkných URL
- Využití sociálních API
- Přístup z Control Panelu
- Chování portletových preferencí
- Podoba ID uživatele
- Chování portletu na stránce
 - Maximalizace
 - Neoprávněný přístup
 - apod.
- Ajaxové načítání portletů
- Cachování
- Sdílené atributy
- JavaScripty a CSS

- Nejčastěji používaný plugin pro změnu fungování portálu
- Extension points
 - lze upravit konkrétní fungování portálu
 - je zajištěna vyšší přenositelnost mezi verzemi
- Co lze změnit
 - jakékoliv JSP
 - implementaci Spring beanů (od verze 6 CE, 5.2 EE)
 - Vybraná nastavení z portal.properties
 - a další

- Stojí trochu mimo základní pluginy
- Možnost rozšířit funkcionalitu o celou webovou aplikaci
 - Funguje mimo samotný portál
 - Může využívat API portálu a měnit jeho chování
- Portál vnitřně neobsahuje žádnou logiku pro podporu webových aplikací
- WAI portlet (Web Application Interface),
 - Vloží do stránky IFrame zobrazující danou aplikaci

- Nástroje, pomocí nichž lze modifikovat jakýkoliv soubor v portálu → mocná zbraň
- Rizika
 - Migrace
 - přímé uzamknutí na implementaci portálu
 - nepřenositelnost kódu na vyšší verzi (neexistuje kompatibilní rozhraní mezi těmito úpravami, které by se neměnilo)
- Kde to jde, použít hook
- Ext nástroje umožňují vyvíjet nové portlety, témata atd.
 - Pomalejší vývoj

- Extension environment (do verze 5.2.3)
 - Merge souborů Liferay a EXT
 - Nová instalace portálu
 - Časově náročný proces
- Extlet (5.2.3)
 - Vyvinula IBA CZ jako open-source
 - Stejné principy jako EXT
 - Lightweight plugin
- Ext plugin
 - Inspirovaný extletem

- Pro implementaci funkcionality využíváme **Portlety**
- Pro vzhled portálu využíváme **Témata a Layout Templates**
- Pro změnu chování portálu využíváme **Hooky**
- Pro nasazení komplexního systému (např. takový, který již má implementováno nějaké UI) užívá se **Web Plugin**
- Jako poslední možnost využíváme **Extlet/Ext Plugin**, pomocí kterého můžeme zasáhnout do implementace portálu

Front-end development

Tag Libraries

- UI Liferay vyvíjeno v JSP
- Sada knihoven značek pro rychlý přístup k připravené funkcionalitě
- Zdrojové kódy
 - *util-taglib*
 - JSP obsahující výkonný kód – *portal-web/docroot/html/taglib*
 - DTD – *util-taglib/src/META-INF*
- Nejpoužívanější
 - *liferay-ui*
 - *liferay-aui*
 - *liferay-security*

- ***liferay-portlet***
 - Implementace taglibu dle <http://java.sun.com/portlet>
 - Dostupné pod namespace *portlet*
- ***liferay-portlet-ext***
 - Rozšíření standardní knihovny pro portlety
 - Obsahuje tagy
 - Pro tvorbu URL s implementačně závislými parametry
 - ID stránky
 - ID portletu
 - Pro zobrazení různých ikon

- Obsahuje sadu silných tagů
 - *buffer* – umožňuje renderovaný obsah tagu umístit do proměnné a s tou dále pracovat
 - *get-url* – umožňuje vložit do cache obsah URL
 - *html-top* – umožňuje vložit obsah do elementu `<head>`
 - *html-bottom* – umožňuje vložit obsah do na konec HTML stránky
 - *include* – umožňuje zavolat `RequestDispatcher.include` na kterýkoliv portlet (se zvolenými parametry `page` a `portletId`)
 - *param* – ve spojení s tagem `include` umožňuje nasatvit parametry do requestu

- *permissionsURL* – Vytvoří URL pro nastavení oprávnění ke konkrétní entitě
- *encrypt* – Vytvoření URL obsahující šifrované atributy
- *doAsURL* – Vytvoření URL pro impersonalizaci

- Obsahují komponenty, které se používají v portálu
- Široký výběr z nejrůznějších komponent
- Vhodné použít
 - Výsledný L&F bude jednotný
 - Vstupní pole obsahují kontroly a jsou ošetřeny proti některým útokům, jako např. XSS

Liferay JavaScript API

- Adresář s JS: */portal-web/docroot/html/js*
- JavaScriptové API lze rozdělit do několika kategorií:
 - JS API pro volání entitních služeb – *liferay/service.js*
 - JS definující chování a zpřístupňující funkce portálu – *liferay/*.js*
 - AlloyUI - základ, na kterém je postaven Liferay v.6+
 - JS ostatních komponent – *calendar/, editor/, firebug/, misc/**
- Portlety nejvíce využívají entitních služeb a JS portálu, které se také občas v projektech mění
- API je přístupné z JS objektu ***Liferay***

- <http://localhost:8080/api/jsonws>

- Mladý framework založený na moderních technologiích
 - CSS3
 - JavaScript
 - HTML5
- Staví se něm UI Liferay portálu
- S Liferay portálem od verze 6
- Nahrazuje dříve používané jQuery
 - Nad ním Liferay neměl kontrolu a špatně se jim dodávala podpora, aniž by přepisovali celé jQuery pod sebe
- Postavený na Yahoo User Interface 3 (YUI3), který rozšiřuje

ALLOY

USERINTERFACE

<http://alloy.liferay.com/demos.php>

- Dependency mgmt. & resources lazy loading
 - Více instancí v sandboxech používající různé knihovny
 - Systém widgetů s životním cyklem, atributy, událostmi atd.
 - Práce s poli
 - Framework pro práci s CSS
 - Jmenné prostory pro pluginy
 - OOP features
-
- jQuery tak již není součástí Liferay, nicméně jej lze použít bez rizika konfliktu s AUI

- Získat alespoň základní představu o tom, jak pracuje YUI3
 - AUI z něj vychází a prakticky lze vše co začíná na YUI nahradit za AUI
- Přečíst si blog Nathana Cavanaugh, který vede a podílí se na vývoji Alloy v Liferay
 - <http://www.liferay.com/web/nathan.cavanaugh/blog/>
- Stáhnout aktuální AUI, v adresáři demos je jich více než na webu

- Sandbox je prostředí, v němž script poběží
- Nebude kolidovat s dalšími skripty
- Bude mít k dispozici potřebné zdroje
 - Balíčky knihoven
 - Načtenou stránku
 - apod.

```
AUI().use(function(A) {  
    // Your code goes here  
});
```

- **AUI()** – znamená, že se bude používat Alloy
- **use()** – dostane 1..n argumentů, kde poslední je callback funkce
- **Argument A** – Alloy objekt se všemi objekty a třídami

```
AUI().use('event', 'node', function(A) {  
    A.one('body');  
});
```

- **AUI().use(...)**
 - počká, až se načtou všechny potřebné knihovny, poté spustí obsah sandboxu
- **AUI().ready(...)**
 - počká, až se načtou všechny potřebné knihovny a je načtena celá stránka, poté spustí obsah sandboxu

```
AUI().ready('node-base', function(A) {  
    // vyber podle ID  
    var div = A.one("#myMainDiv");  
    var div = A.all("#myMainDiv"); // all – velmi drahé, nepoužívat  
  
    // výběr podle typu elementu  
    var p = A.one("p");  
    var p = A.all("p");  
  
    var empty = A.one("#notExistingElem"); // empty == null  
    var empty = A.all("#notExistingElem"); // empty prázdné pole  
  
    // výběr v rámci elementu  
    var span = A.one("#footer").one("#spanText");  
});
```

```
AUI().ready('node-base', function(A) {  
    var a = A.one("#click");  
    a.on("click", function(e) {  
        alert("Button clicked");  
    });  
  
    // funkce fn pro testování událostí  
    function fn(e){alert("test");}  
  
    // více funkcí na různé události  
    a.on({  
        click      : fn,  
        focus      : fn,  
        blur       : fn,  
        mouseout   : fn,  
        mouseover  : fn  
    });  
  
    // jedna funkce více událostem  
    a.on(['click', 'focus', 'blur', 'mouseout', 'mouseover'], fn)  
});
```



```
AUI().ready('node-base', function(A) {  
    // nastaví jednu CSS vlastnost  
    A.one("#footer").setStyle("backgroundColor", "red");  
  
    // nastaví víc CSS vlastností  
    // různé metody setStyle/setStyles  
    A.one("#footer").setStyles({  
        "backgroundColor": "red",  
        "textAlign": "right"  
    });  
});
```

```
AUI().ready('node-base', function(A) {  
    A.one("#footer").show()  
    A.one("#footer").hide()  
  
    // pokud je zavedený modul na přechody,  
    // bude s nastaveným true fungovat  
    // fade-in a fade-out  
    A.one("#footer").show(true)  
    A.one("#footer").hide(true)  
});
```

```
AUI().ready('io', function(A) {  
    A.io("***URL_STRING***", {  
        data: {  
            param1: "val_1",  
            param2: "val_2"  
        },  
        timeout : 10000  
        on: {  
            success: function(transactionid, o, arguments) {  
                A.log(eval('o.responseText'));  
            },  
            failure: function() {  
                A.log("Ajax can not get data from server");  
            }  
        }  
    });  
});
```

Závěr

- <http://www.liferay.com/>
- Dokumentační projekt
 - <http://www.liferay.com/documentation/liferay-portal>
- Komunitní stránky – <http://liferay.org>
 - Fóra – <http://www.liferay.com/community/forums>
 - Wiki – <http://www.liferay.com/community/wiki>
 - IRC – <http://webchat.freenode.net/?channels=liferay>
- Jira – <http://issues.liferay.com>
- Zdrojové kódy
 - <https://github.com/liferay>
- AlloyUI – <http://alloy.liferay.com>

