Příloha č. 8

Základní požadavky systémového integrátora kladené na vytváření portálu esfcr.cz budovaném na portálovém frameworku

Obsah

1. ÚVOD

Tato příloha definuje základní pravidla pro multidodavatelské prostředí portálu esfcr.cz na MPSV. Cílem pravidel je zachovat a zajistit platformu budovaného Portálového frameworku, která by měla být i po rozšiřování maximálně šetrná ke změnovým požadavkům a zároveň udržitelná v dlouhodobém horizontu, bezpečná a výkonnostně vyrovnaná. Dokument je připraven tak, aby navazoval na plánovanou celkovou koncepci multivendor platformního prostředí, se kterým se v případě domény esfcr.cz počítá.

2. CELKOVÉ PROSTŘEDÍ A ZASAZENÍ DO KONTEXTU

Portálový Framework v prostředí MPSV je postavený na technologii Liferay 6.2 CE. Liferay 6.2 CE je bezplatný open-sorce portálový framework založený na jazyce JAVA a distribuovaný pod GNU Lesser General Public Licence a dalšími proprietárními licencemi. Liferay portál podporuje standardy JSR-168, JSR-127, JSR-170, JSR-286 a JSF-314.

Nejčastějším způsobem dodávky řešení je Vývoj. Proto je tento dokument primárně zaměřen na detailní vývojové požadavky, při jejichž dodržení budou výsledná řešení dobře provozovatelná. Dále v dokumentu jsou definované bázové technologie a možnosti použití těchto technologií ve zdrojových kódech.

3. TECHNOLOGIE

Vyvíjená řešení musí být postavena na následujících bázových technologiích. Současně je uveden popis jejich použití v rámci Architektury.

3.1. Přístup k databázi

- Pro přístup k DB musí být vždy využito technologie connection pool. Nesmí použít přímé spojení k DB
- Aplikace nesmí držet vlastní connection pool bez schválení.
- Když spojení již není používáno, musí být řádně uzavřeno a vráceno do poolu.
- Pouze jeden přístup (Hiber, JDBC template...)
- Často opakované dotazy a dotazy do více tabulek musí být prováděny nad indexovanými sloupci.
- Přístup do databáze je podle DAO vzoru.
- Při běžném provozu aplikace nesmí používat zbytečně složité nebo dlouhotrvající dotazy (delší než 1s). Ve stejné době nesmí být vytvořeny nepřímé dotazy do DB v rámci jedné žádosti (max. několik desítek dotazů).

3.2. Hibernate

Při práci s Hibernate ORM, je potřeba dbát na následující vlastnosti:

- Cascades – pouze ve výjimečných případech, neměla by se násobit.
- Eager / lazy fetch – eager fetch by měl být použit pouze v odůvodněných případech.
- Flush & refresh – aplikace by neměla napřímo volat tyto metody.
- Transactions – aplikace musí zpracovávat transakce v jedné cestě, aby bylo zcela jasné jak fungují. Nejlepší možností je transaction control pomocí frameworku Spring level.

3.3. Portlety

3.3.1. Architektura portletové aplikace

Aplikační architektura by měla být navržena podle modelu MVC, kde jsou data, aplikační logika a prezenční logika jasně oddělena. Tento model splňuje Spring portlet MVC framefork, který je doporučen v prostředí MPSV. Nedodržení modelu MVC vede k obtížným údržbám a zvyšuje náklady na změny.

Aplikace by měly být navrhnuty podle IOC (Inversion of Control) modelu architektury, který může být využit pro zabránění vytváření monolitických nezvladatelných aplikací.

Pro větší udržitelnost musí být aplikační servisy, které využívají Liferay API, izolovány od zbývající části aplikace. Při práci s Liferay API by měla být veškerá funkcionalita zahrnuta v rámci jedné třídy nebo balíčku. Použití Liferay API je pak snadno modifikovatelné na jednom místě a nemá vliv na zbývající části aplikace. Je využito Facade design modelu. Nedodržení tohoto pravidla vede ke zvýšení nákladů při migraci portálu na vyšší verzi. Příklad použití PortalUtil z Liferay API:

```
@Component
public class PortalFacade {
      public     HttpServletRequest     getHttpServletRequest(PortletRequest
request){
            return PortalUtil.getHttpServletRequest(request);
      }
   }
```

Společné funkce, které používá řada portletů by měla být soustředěna do knihoven, aby se zabránilo zdvojování kódu.

JavaScriptové knihovny, které jsou používány v celém portálu, musí být umístěny do tématu, aby se zabránilo duplicitě. Portál řeší zdvojení JavaScriptu pouze na úrovni WAR souborů. Vývojář musí řešit konflikt umístění JavaScriptu mezi aplikační knihovny, téma a portál.

Chyba v portletu nesmí rušit provoz portálu a jiných portletů. Jestliže nastane porucha v některém z portletů, Liferay portál zajistí správné generování chybové zprávy, kterou musí portlet zobrazit. Problém nastane, když portlet nekončí chybou, ale čeká na pomalé vzdálené zdroje. Volání vzdálených zdrojů by tedy měla mít časové limity. Pro urychlení zobrazení stránky, může být použito AJAX volání pomalých portletů. Liferay nabízí vestavěné funkce pro tento účel. Více informací je dostupné v Liferay-portlet.xml DTD, render-weight setting.

3.3.2. Meziportletová komunikace (IPC, Inter-portlet communication)

Inter-portlet komunikace je definována ve standardu JSR-286. Tento standard musí být dodržen.

3.4. Servlets

Aplikace může používat servlety pouze po schválení systémovým integrátorem. Servlety nesmí být řízeny na úrovni portálu. Místo servletů může být využito kombinace funkcionalit getResource a processAction metod, které jsou součástí Portlet 2.0 API.

3.5. Spring

Spring framework je v současnosti jedním z nejrozšířenějších frameworků v oblasti vývoje webových aplikací. Nabízí velké množství funkcionality, která urychluje, zefektivňuje a zpřehledňuje vývoj aplikací. Existuje velká množina knihoven, které mají podporu pro Spring. Integrování těchto aplikací je potom velmi jednoduché a často založené jen na správné konfiguraci XML souborů frameworku Spring. Tím, že je celá aplikace postavena na frameworku Spring, je umožněno její konfigurování z jednoho místa pomocí konfiguračních souborů frameworku. Spring je velmi intuitivní, čímž máme zabezpečenou vysokou efektivitu i u programátorů, kteří s frameworkem Spring teprve začínají. Samotný Liferay využívá Spring na úrovni business logiky.

Spring je tvořený různými moduly. Udržuje se tím kompaktní velikost frameworku a aplikace neobsahuje knihovny, které nepotřebuje. Jedním z populárních modulů je Spring MVC, který slouží k usnadnění vývoje webových aplikací. Z tohoto modulu vychází i modul Spring Portlet MVC, který přidává ke klasickému MVC portletové specifika. Portletová funkcionalita specifikovaná v JSR-286 je v modulu Spring Portlet MVC plně nativně podporovaná (bez využití bridge). Zabezpečuje se tím maximální výkon.

Spring Portlet MVC podporuje různé skriptovací jazyky, které zabezpečují vykreslování výsledků. Pro efektivnost, rychlost a jednoduché používání doporučujeme pro generování stránek používat JavaServerPages (JSP). Jedná se o Java standard. Spring nabízí tagliby, které zjednodušují a

urychlují práci s JSP. Volba jiné technologie musí být v rámci návrhu architektury explicitně konzultována se systémovým integrátorem.

Výhody používání Spring Portlet MVC:

- jednotný framework napříč celou aplikací, kód je tím čitelnější, uniformnější a lépe udržovatelný, nevznikají konflikty mezi frameworky
- léty ověřený a dobře fungující koncept
- plně podporuje JSR-286
- podpora pro různé knihovny, které jsou snadno integrovatelné
- jednotná konfigurace
- rychlé zaučení vývojářů a zefektivnění práce
- rozsáhlé použití frameworku Spring a velká komunita pomáhá k rychlému řešení problémů
- podpora v různých IDE

3.6. Liferay Portal

3.6.1. Vývoj

Funkcionalitu v portálu Liferay je možné rozšiřovat a upravovat pomocí zásuvných modulů (pluginů). Jedná se webové aplikace s definovanou strukturou a obsahem. Ty je možné vytvářet pomocí Liferay Plugins SDK. Ten je v současnosti k dispozici ve dvou variantách

- Plugins SDK (postavený na nástroji Apache Ant, často používaný v nejrůznějších tutorialech)
- Maven SDK (sada Maven archetypů)

Pro potřeby vývoje pro portál navrhujeme využívat výhradně Maven SDK.

Různé pluginy mají různé použití. V základě se pluginy rozdělují podle toho, zda umožňují funkcionalitu přidávat nebo měnit.

Pro přidávání nové funkcionality slouží:

- Portlety,
- Layout Template,
- Theme,
- Web Application.

Pro změnu chování implementace Liferay:

- Hook,
- Ext Plugin.

Ve zkratce:

- pro implementaci funkcionality využíváme Portlety,
- pro vzhled portálu využíváme Témata a Layout Templates,
- pro změnu chování portálu využíváme Hook,
- jako poslední možnost využíváme Ext Plugin, pomocí kterého můžeme zasáhnout do implementace portálu.

Vytváření Hooků a Ext pluginů je podmíněno povolením ze strany systémového integrátora a po dodání budou revidovány.

3.6.2. Theme

Jedná se o nástroj pro definování vzhledu portálových stránek. Pro každý web a každou stránku je možné definovat, které téma má používat a jak má tedy vypadat. Téma definuje vzhled portálové stránky na úrovni CSS stylů, JavaScriptu, záhlaví, zápatí. Dále umožňuje definovat především vzhled navigace (seznam stránek) a vzhled portletu.

3.6.3. Layout Template

Je nástroj na definování rozložení stránky - sloupců. Jedná se o doplněk témat, pomocí kterého můžeme nastavit rozložení sloupců a řádků, do kterých je možné vkládat portlety.

3.6.4. Portlety

Jsou základní stavební jednotkou portálu a jsou primárním způsobem přidávání nové funkcionality do portálu. V prostředí MPSV se používají JSR-286 portlety, které využívají Spring portlet MVC framework. Při vytváření portletů není doporučeno používat Liferay Service Builder.

3.6.5. Ext Plugin

Jedná se o nástroj, pomocí kterého je možné modifikovat jakýkoliv soubor v portálu. Jedná se tedy o velmi silnou zbraň, která na druhou stranu nese rizika s migrací. Často přímým uzamčením na implementaci portálu dochází k nepřenositelnosti kódu na vyšší verze, neboť neexistuje kompatibilní rozhraní mezi těmito úpravami, které by se neměnilo.

Samozřejmě je možné psát nové portlety, témata a vše i za pomocí těchto nástrojů. Vývoj je ale mnohem pomalejší než za použití ostatních typů pluginů, které jsou k tomu určený.

Zásady při psaní Ext pluginů:

- změna musí být co nejméně invazivní,
- izolovat změny do nových tříd a nepřepisovat přímo Liferay kód,
- v případě přepisování Liferay kódu třeba změnu provést v nové metodě a tu volat z kódu,
- důkladně změnu okomentovat, zachytit důvod změny a způsob řešení,
- nepoužívat Ext Plugin pokud je možné použít Hook.

3.6.6. Hook

Je to nejčastěji používaný plugin pro změnu fungování portálu. Obsahuje definované body rozšíření, pomocí kterých lze upravit konkrétní funkčnost portálu, takže je zajištěna vyšší přenositelnost mezi verzemi.

Hook umožňuje úpravu:

- JSP souborů,

- vybraných řádků z portal.properties,
- události + model listeners,
- lokalizace (language.properties),
- Spring bean,
- Servlet filtry (od Liferay 6.1),
- Struts akcí (od Liferay 6.1),
- Indexer post-processor (od Liferay 6.1),
- Application Adapter (od Liferay 6.1).

3.6.7. Webová aplikace v kontextu Liferay

Tento plugin stojí trochu mimo základní pluginy, neboť se jedná o možnost rozšířit funkcionalitu o celou webovou aplikaci. I když webová aplikace funguje mimo samotný portál, může využívat API portálu a změnit jeho chování. Vnitřně ale portál neobsahuje žádnou logiku pro podporování webových aplikaci, až na portlet WAI (Web Application Interface), který vloží do stránky IFrame zobrazující danou aplikaci.

Webové aplikace, které stojí mimo Liferay portál, nejsou pokryty tímto dokumentem a pokud by měly existovat a napojovat se na esfcr.cz, bude vždy nutná konzultace se zadavatelem.

4. POVINNÝ OBSAH

Pro různé druhy externě připravovaných balíčků jsou definovány následující povinné soubory, které musí být průběžně aktualizovány:

- Pluginy pro Liferay portál
  - ***liferay-plugin-package.properties***
- Portletové aplikace
  - ***portlet.xml***
  - ***liferay-portlet.xml***
  - ***liferay-display.xml***

Součástí musí být i eventuální sada SQL skriptů pro tvorbu všech potřebných databázových objektů a základních údajů. V průběhu dalších dodávek, musí existovat přírůstkové SQL skripty, které můžou být aplikovány na systém, který je již v provozu.