Přeloženo komunitou platformy NetBeans

Předmluva: **Jaroslav Tulach**
původní architekt NetBeans API

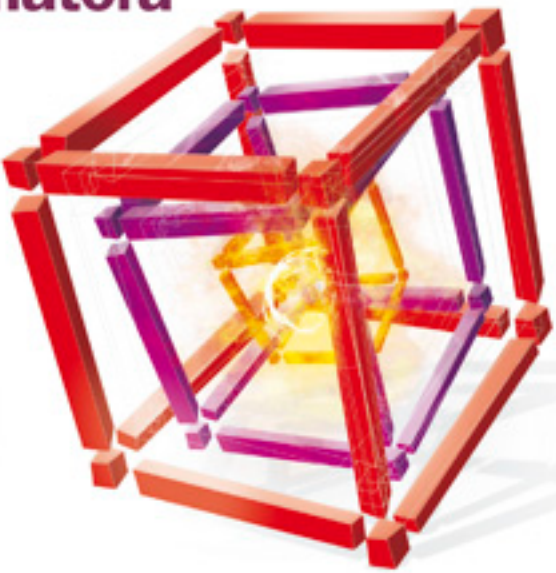
Platforma NetBeans

Podrobný průvodce programátora

Naučte se tvořit Rich Client aplikace v Javě

Ovládněte všechny dostupné moduly a API

Tipy a triky z praxe a vzorové projekty



WWW

Ke stažení zdrojové kódy příkladů

CPRESS Apress

Heiko Böck

Platforma NetBeans
Podrobný průvodce programátora

Computer Press, a. s.
Brno
2010

Platforma NetBeans
Podrobný průvodce programátora

Heiko Böck

Computer Press, a.s., 2010. Vydání první.

Překlad: Jaroslav Bien, Miloš Šilhánek, Michal Petřík
Jazyková korektura: Veronika Macková, Alena Láníčková
Vnitřní úprava: Kateřina Kiszková
Sazba: Kateřina Kiszková
Rejstřík: Kateřina Kiszková

Obálka: Martin Sodomka
Komentář na zadní straně obálky: Martin Domes
Technická spolupráce: Zuzana Šindlerová
Odpovědný redaktor: Martin Domes
Technický redaktor: Jiří Matoušek
Produkce: Petr Baláš

Computer Press, a.s.,
Holandská 8, 639 00 Brno

Objednávky knih:
http://knihy.cpress.cz
distribuce@cpress.cz
tel.: 800 555 513

ISBN 978-80-251-3116-9
Prodejní kód: K1828
Vydalo nakladatelství Computer Press, a.s., jako svou 3650. publikaci.

Obsah

KAPITOLA 3

Systém pro správu modulů 27

KAPITOLA 4

Akce 51

KAPITOLA 5

Design uživatelského rozhraní 65

KAPITOLA 6

Lookup 95

KAPITOLA 12

KAPITOLA 13

KAPITOLA 14

KAPITOLA 15

Rozšiřujeme NetBeans IDE 249

KAPITOLA 16

Od Eclipse RCP k platformě NetBeans 259

KAPITOLA 17

Tipy a triky 267

KAPITOLA 18

Předmluva

Nejlepší způsob, jak zlepšit svoje schopnosti, je potkat někoho, kdo se k vám přidá, začne spolupracovat a dokáže lépe to, co jste předtím dělali sami.

Takové příběhy jsem zažil několikrát. Nebylo tak špatné navrhovat vývojové prostředí NetBeans a platformu NetBeans. Vše ale dostalo nový rozměr, jakmile jsem mohl spolupracovat se spoustou talentovaných vývojářů navrhujících jednotlivé části NetBeans. I psaní dokumentace pro platformu NetBeans se dalo zvládnout, ale od té doby, co se přidaly hordy nadšenců, kteří píší zajímavé tutoriály, blogy a popisují témata, která by mne vůbec nenapadla, jde vše mnohem lépe. To samé se přihodilo i s knihami. Líbilo se mi napsat pár kapitol do knihy o platformě NetBeans, ale je mnohem lepší držet v ruce výtisk Heikovy knihy. Je úplnější, zajímavější a popisuje témata, která by mne ani ve snu nenapadla.

Tato kniha mne provází již dlouho. Poprvé jsem ji viděl v roce 2007, když Heiko dokončil svoji německou verzi a já byl požádán o překontrolování jejího obsahu. Protože však moje znalost němčiny je více než malá, scvrklo se mé čtení na čtení zdrojových kódů v Javě a ověření, že používají postupy vhodné pro platformu NetBeans. Vše bylo samozřejmě v naprostém pořádku. Navíc i jen z pouhých kódů jsem vycítil, že témata knihy jsou skvělá a že přinášejí mnoho nových poznatků do světa platformy NetBeans.

Jsem rád, že tato kniha doputovala ke svému dalšímu milníku. Jsem rád, že se našla parta nadšenců, jež ji přeložila do češtiny. Nyní již nic nebrání, aby platforma NetBeans vtrhla na naše vývojová pracoviště, do univerzitních poslucháren českých vysokých škol či do školních škamen! A nebo, alespoň abych si, poprvé v životě, mohl přečíst knihu o svém osudovém projektu ve svém rodném jazyce. Děkuji všem, jejichž tvrdá práce toto vše umožnila.

Jaroslav Tulach, zakladatel a původní architekt platformy NetBeans

Předmluva k českému vydání

Rád jsem se zúčastnil překladu této knihy, protože popisuje moje oblíbené téma a překládání mě bavilo. Během překladu knihy jsem k ní získal velkou úctu. Čtenáři předkládám pár svých postřehů:

- Autor velice pečlivě naplánoval postup výkladu a knihu velmi dobře napsal.
- Vzorový příklad (MP3 manažer s přehrávačem) pokrývá všechna témata, se kterými se při učení platformy setkáte. Buduje se krok za krokem a v poslední kapitole se všechny znalosti integrují do aplikace.
- Kniha seznámí se základní filozofií a použitými principy platformy NetBeans.
- Poskytne mnoho detailů a specialit, které byste jinak museli hledat v diskuzích a v mailinglistech.

- Varuje před obvyklými chybami implementace. Namátkou – že je třeba poprvé zavolat metodu lookup, aby se aktivoval posluchač na Lookupu. Abyste nezapomněli nastavit ActionMap do Lookupu, když pro vaši top-komponentu vytváříte vlastní Lookup. Nebo že pro funkčnost akce DropAction na Widgetu musíte nastavit AbsolutLayout. Výborný je úvod do použití Hibernate a JPA.
- Zastoupí spoustu tutoriálů na netbeans.org – ty si ovšem zkuste taky :-).
- Pěkně popisuje základní úlohy a vzory používané při práci s platformou. Stačí si úsek kódu půjčit, zbytek dohledáte v textu nebo v dokumentaci.
- Ušetří spoustu času, protože je ucelená a nepotřebujete tuny jiné dokumentace.
- Příloha obsahuje nejpoužívanější body rozšíření modulů platformy.
- Až se s platformou seznámíte hlouběji, stále se ke knize budete vracet jako k referenční příručce.

Miloš Šilhánek, překladatel a člen komunity NetBeans

Když jsem pracoval na univerzitě na své diplomové práci postavené mimo jiné na platformě NetBeans, tak k dispozici tato skvělá kniha nebyla. Za několik let práce s platformou jsem se již mnohému naučil, na spoustu věcí si přišel sám čtením dokumentace, článků, archivu mailing listů či procházením zdrojových kódů platformy. Ovšem moje znalosti mohly být získány mnohem rychleji, kdyby existovalo kompletní shrnutí veškerých typických postupů, základů práce, vzorů a tipů pro použití platformy NetBeans. A tím je právě tato kniha. Dnes mohou začínající vývojáři vzít do ruky tuto knihu, a do světa platformy tak proniknout daleko rychleji a využít zkušenosti jiných. Kniha je však vhodná i pro zkušenější vývojáře, neboť z vlastní zkušenosti mohu říct, že v knize je spousta užitečných příkladů a vychytávek, které můžou použít při vývoji aplikací na platformě postavených. Proto knihu doporučuji všem vývojářům pracujícím s platformou NetBeans.

Kniha pokrývá kompletně veškerá používaná API, stejně tak jako vhodné postupy, vzory a způsob práce při vývoji, a konečně dává i spoustu užitečných tipů použitelných při programování desktopové aplikace postavené na platformě NetBeans. Knihu lze opravdu považovat za kompletního průvodce platformou NetBeans. Nyní je tato kniha k dispozici pro většinu programátorů již i v angličtině, stejně tak je překládána do dalších jazyků. Jsem rád, že jsem mohl být součástí týmu, který tuto knihu přeložil do jazyka komunity vývojářů v zemi původu platformy NetBeans – do češtiny.

Děkuji autorovi knihy za skvělou práci pro komunitu, stejně tak týmu spolupracovníků na překladu a zaměstnancům české pobočky společnosti Sun Microsystems, Inc., resp. Oracle Corporation, za veškerou podporu a spolupráci při překladu a vydání knihy. Doufám, že kniha bude přínosem pro českou komunitu vývojářů a pomůže jejímu dalšímu růstu a rozšíření.

Jaroslav Bien, překladatel a člen komunity NetBeans

S platformou NetBeans jsem se poprvé setkal díky NetBeans IDE verze 4. V té době jsem si vůbec nedovedl představit, jaký má tato platforma potenciál, kolik šikovných lidí kolem sebe v budoucnu soustředí a jak neuvěřitelný pokrok za pár let zaznamená. Ze studentského projektu se díky vizi a nadšení vyvinulo jedno z nejlepších Java IDE na světě. Jsem velmi rád, že jsem mohl přispět k popularizaci a rozšíření povědomí o platformě NetBeans a pevně věřím, že tato publikace je jen jedna z dlouhé řady knih, které nás ve vztahu k NetBeans ještě čekají.

Michal Petřík, překladatel a člen komunity NetBeans

KAPITOLA 1

Úvod

V této kapitole:

- Co je to Rich Client
- Co je to platforma Rich Client
- Výhody Rich Client platformy
- Charakteristika platformy NetBeans
- Shrnutí

O čem je tato kniha

Tato kapitola vás uvede do tématu „Rich Client“ (pozn. překl.: doporučujeme používat anglický technický termín). Dozvíte se, co to Rich Client je a jak vám může pomoci Rich Client platforma. Nakonec se krátce zmíníme o přednostech a charakteristikách platformy NetBeans.

Co je to Rich Client

V architektuře klient-server se termín „Rich Client“ používá pro klienty, když se data zpracovávají většinou na straně klienta. Klient také poskytuje grafické uživatelské prostředí (GUI). Většinou se jedná o aplikace, které se dají rozšířit pomocí zásuvných modulů. Rich Client může řešit více než jeden problém, často dokonce i problémy, které jsou vzdálené jeho původnímu účelu.

Rich Client je obvykle postaven nad nějakým základem – základním rámcem (frameworkem). Tento základní rámec nabízí určitou infrastrukturu, na které může uživatel postavit aplikaci z logických částí, kterým se říká moduly. Ideální je, když nesouvisející řešení (třeba vytvořená různými výrobci) mohou pracovat společně, takže několik modulů se může tvářit jako jeden obrovský modul. Výrobci a poskytovatelé softwaru mohou skládat distribuce Rich Client aplikace z různých modulů podle přání a potřeb uživatele.

A navíc má Rich Client výhodu ve snadné distribuci a aktualizaci, např. automatickou aktualizací přes Internet nebo možností spustit klienta přímo z Internetu, třeba pomocí Java Web Start.

Přehled vlastností Rich Client aplikace:

- flexibilní a modulární architektura aplikace,
- nezávislost na platformě,
- přizpůsobení uživateli,
- možnost pracovat on-line i off-line,
- jednoduchá distribuce k uživateli,
- jednoduchá aktualizace klienta.

Co je to platforma Rich Client

Rich Client platforma je program, který poskytuje prostředí pro životní cyklus aplikace a je základem desktopové aplikace. Mnoho desktopových aplikací má stejné nebo podobné vlastnosti. Jsou to například nabídky (menu), nástrojové lišty, stavový řádek, zobrazení průběhu zpracování, zobrazení dat, přizpůsobení nastavení, ukládání a načtení konfigurace a uživatelských dat, úvodní obrazovka (splash screen), okno O aplikaci (About box), nastavení národního prostředí (internationalization), systém nápovědy atd. Framework už všechny tyto funkce a komponenty obsahuje a není třeba je programovat.

Možnost konfigurace a rozšiřitelnost aplikace jsou pak hlavním znakem takového frameworku. Výsledkem například je, že položky nabídky zapíšete deklarativně do textového souboru, odkud si je framework automaticky nahraje. To znamená, že se zdrojový kód soustředí do jednoho místa a vývojáři se mohou starat jen o skutečné problémy budoucí aplikace.

Nejdůležitější vlastností Rich Client platformy je její architektura. Aplikace postavené na takové platformě se vytvářejí ve formě modulů, ve kterých jsou izolovány jednotlivé logicky ucelené části aplikace. Modul je popsán deklarativně a platforma ho automaticky načte. To vede k tomu, že nejsou nutné explicitní vazby mezi vaším kódem a aplikací, takže mezi samostatně fungujícími moduly jsou volné vztahy a zjednoduší se dynamické rozšíření aplikace nebo změna chování. Tímto způsobem je velice

jednoduché sestavit z jednotlivých modulů aplikaci specifickou pro daného uživatele nebo pro problémovou doménu.

Rich Client platforma osvobodí vývojáře od úkolů, které nesouvisí s obchodní logikou aplikace. Na konci vývoje získáte dobře vytvořenou a moderní architekturu aplikace.

Výhody Rich Client platformy

Kromě modularity, kterou nabízí Rich Client architektura a která vede k robustní a pro koncové uživatele hodnotné aplikaci, musíme zdůraznit i rozsáhlou podporu pro vlastní vývoj. Uveďme ještě několik dalších výhod Rich Client platformy:

- zkrácení doby vývoje,
- konzistence uživatelského rozhraní,
- aktualizace,
- nezávislost na platformě,
- možnost opakovaného použití a spolehlivost.

Nyní si je představíme blíže.

Zkrácení doby vývoje

Rich Client platforma poskytuje velké množství funkcí (a rozhraní, API, programových balíčků) pro vývoj desktopových aplikací. Funkce platformy mohou být například použity programátorem ke správě oken a menu nebo pro nastavení voleb. Díky znovupoužitelným připraveným komponentám se vývojář může soustředit jen na obchodní logiku aplikace.

Konzistence uživatelského rozhraní

Použitelnost aplikace je klíčovou vlastností, zvláště když je určena pro specialisty v dané oblasti. Rich Client platforma obsahuje framework pro zobrazení uživatelského prostředí a kromě toho za nás řeší i konzistenci, přístupnost a použitelnost.

Aktualizace

Při použití Rich Client platformy máte k dispozici i rychlý a efektivní způsob distribuce nových modulů nebo aktualizace. To znamená, že všichni uživatelé aplikace nemusí být informováni o nové verzi a nutnosti aktualizovat. Aktualizace jsou distribuovány a instalovány ve formě modulů, takže jednotlivé funkce aplikace mohou být vyvíjeny a dodávány nezávisle na ostatních. Modulární architektura aplikace pohlídá i to, že distribuce některé části bude nebo nebude čekat, dokud nebudou hotovy jiné moduly.

Nezávislost na platformě

Rich Client platformy jsou vytvořeny na základě mezinárodních standardů a komponent určených pro opakované použití. Výsledkem je, že na ní vytvořenou javovou aplikaci můžete nainstalovat na různé systémy, třeba Windows nebo Linux, kde je dostupné běhové prostředí jazyka Java (JRE – Java Runtime Environment). Je vemi důležité, že jednou vyvinutá aplikace může být bez úpravy rovnou nasazena v různých systémech. Takže platforma nám šetří čas a peníze. Aplikace

postavená na Rich Client platformě nepotřebuje další knihovny nebo komponenty, jen běhové prostředí Javy.

Opakované použití a spolehlivost

Rich Client platforma obsahuje množství funkcí a modulů, které mohou být použity ve vaší aplikaci. A když modul nesplňuje všechny požadavky aplikace, může posloužit jako základ pro rozšíření nebo změnu, pokud je potřeba. Většinou je k platformě k dispozici i zdrojový kód, takže je možné se podílet na úpravách nebo změně vlastní platformy. To je důležité pro vysoký stupeň spolehlivosti a svobody.

Charakteristika platformy NetBeans

Platforma NetBeans nabízí, kromě obecných předností Rich Client platformy, množství frameworků, API (programových balíčků) a několik specifických vlastností, které vám mohou být užitečné. Některé důležité vlastnosti, které jsou charakteristické pro NetBeans, si dále popíšeme:

- podpora uživatelského prostředí,
- editor,
- přizpůsobení zobrazení,
- podpora pro vytváření průvodců,
- datové systémy,
- internacionalizace,
- systém nápovědy.

A teď podrobněji.

Podpora uživatelského prostředí

K dispozici máte okna, nabídky, nástrojové lišty a další komponenty. Vy se věnujete jen vytvoření vlastních akcí, do kterých soustředíte svůj kód. Bude jednoduchý, lepší a odolný proti chybám. GUI platformy NetBeans je postaveno stoprocentně na AWT/Swing a dá se rozšířit vlastními komponentami.

Editor

Výkonný editor NetBeans, který je v NetBeans IDE, můžete použít ve svojí aplikaci také. Nástroje a funkce editoru můžete rychle a snadno rozšířit a přizpůsobit pro své účely.

Přizpůsobení zobrazení

Zobrazení uživatelských a aplikačních nastavení je důležité v každé aplikaci. NetBeans má vytvořený framework, který vám umožní jednoduše vytvořit a integrovat vaše dialogy pro nastavení voleb a vlastností v elegantní formě.

Podpora pro vytváření průvodců

NetBeans obsahuje jednoduché nástroje pro vytvoření rozšiřitelných a uživatelsky přívětivých průvodců, které provedou uživatele jednotlivými kroky složitějších úkolů aplikace.

Datové systémy

Načítaná data mohou být lokální nebo přístupná přes FTP, CVS, z databáze nebo ve formě XML souboru. V NetBeans existuje abstrakce, která jedním modulem transparentně zpřístupní data ostatním modulům. Skutečný původ dat už není důležitý, protože se o něj postaralo API platformy NetBeans.

Internacionalizace

NetBeans poskytuje třídy a metody pro internacionalizaci nápovědy a ostatních zdrojů. Jednoduše uložíte textové konstanty do souborů vlastností (.properties) a platforma již sama načte texty a ikony podle aktuálního nastavení země a jazyka.

Systém nápovědy

NetBeans poskytuje centrální systém pro integraci a zobrazení témat nápovědy. Je založen na standardním *JavaHelp*. Každý modul se může podílet svým tématem na nápovědě celé aplikace. Kromě toho platforma poskytuje také svoji kontextovou nápovědu.

Shrnutí

V této kapitole jsme se naučili, co to je Rich Client a jaké má výhody, probrali jsme modulární architekturu a její použití v Rich Client platformách. Zmínili jsme také, že Rich Client platforma poskytuje mnoho dalších výhod a vlastností. Zmínili jsme např. podporu pro konzistentní uživatelské rozhraní a aktualizaci aplikace za běhu. Nakonec jsme popsali několik základních charakteristik platformy NetBeans.

KAPITOLA 2

Struktura platformy NetBeans

V této kapitole:

- Architektura platformy NetBeans
- Distribuce platformy NetBeans
- Běhový kontejner NetBeans
- Systém zavaděčů tříd NetBeans
- Shrnutí

Podívejte se dovnitř!

Abychom získali přehled o tom, jak je Rich Client aplikace strukturována a jaký je vztah mezi vaší aplikací a platformou NetBeans, pohovoříme v této kapitole o architektuře platformy NetBeans. Také vám představíme nezávislé základní stavební bloky platformy NetBeans a úkoly, které pro vás zajistí běhový kontejner platformy. Nakonec vysvětlíme strukturu systému zavádění tříd v NetBeans a její roli při budování aplikace nad platformou.

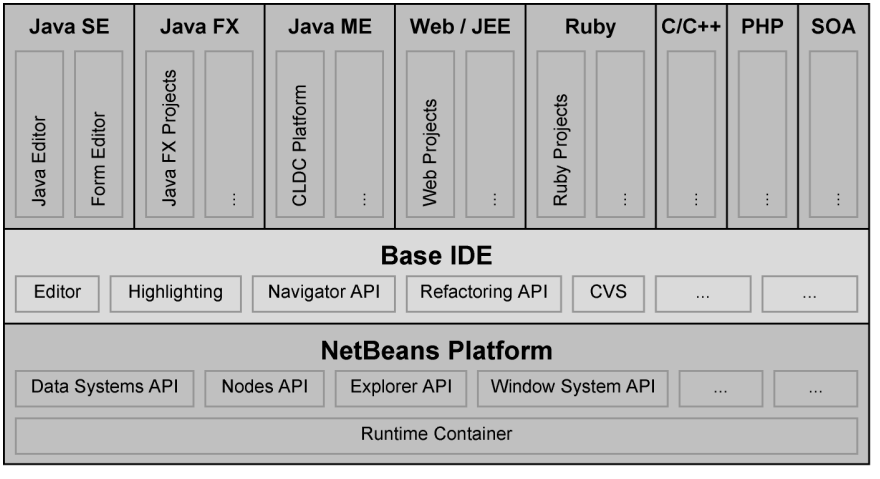
Architektura platformy NetBeans

Velikost a komplexnost moderních aplikací neustále roste. Současně ale musí být profesionální aplikace flexibilní a rychle a snadno rozšiřitelná. Proto je vhodné aplikaci rozdělit do samostatných částí a každou část sestavit jako blok s modulární architekturou. Tyto samostatné části musí být nezávislé a musí mít dobře definované rozhraní, které mohou využívat ostatní části aplikace a rozšiřovat jejich funkce.

Rozdělení aplikace do modulů, což jsou logicky nezávislé části, enormně zvýší kvalitu návrhu aplikace. Oproti monolitické aplikaci, v níž může každá třída používat kód jiné třídy, je tato architektura daleko flexibilnější a daleko snáze udržovatelná. Pravda, můžete třídu skrýt a chránit před vnějším světem, ale tato ochrana na úrovni tříd je příliš jemná pro to, aby byla užitečná v rámci stavby aplikace. Platforma NetBeans řeší právě tento základní aspekt moderní klientské aplikace. Její principy a struktura přímo podporuje vývoj a koncepci flexibilní a modulární aplikace.

Základním stavebním kamenem platformy NetBeans je *modul* (module). Modul je sada funkčně svázaných tříd a jiných zdrojů, popisu rozhraní, které modul vystavuje, a také popis ostatních modulů, které modul potřebuje pro svoji funkci. Celá platforma NetBeans (a také každá aplikace nad ní postavená) je rozdělená do modulů. Ty jsou načteny jádrem platformy NetBeans, které je známo jako běhový kontejner NetBeans (*NetBeans Runtime Container*). Běhový kontejner načítá moduly aplikace dynamicky a automaticky, přičemž je zodpovědný za běh celé aplikace.

NetBeans IDE je velice dobrým příkladem modulární Rich Client aplikace. Funkčnost a vlastnosti IDE, jako je třeba podpora jazyka Java a editor zdrojového kódu, jsou vytvořeny ve formě modulů platformy NetBeans. To má i tu obrovskou výhodu, že se aplikace dá rozšířit dalšími moduly, a tak přizpůsobit potřebám uživatele. Také umožní nepoužívané moduly deaktivovat nebo odinstalovat.

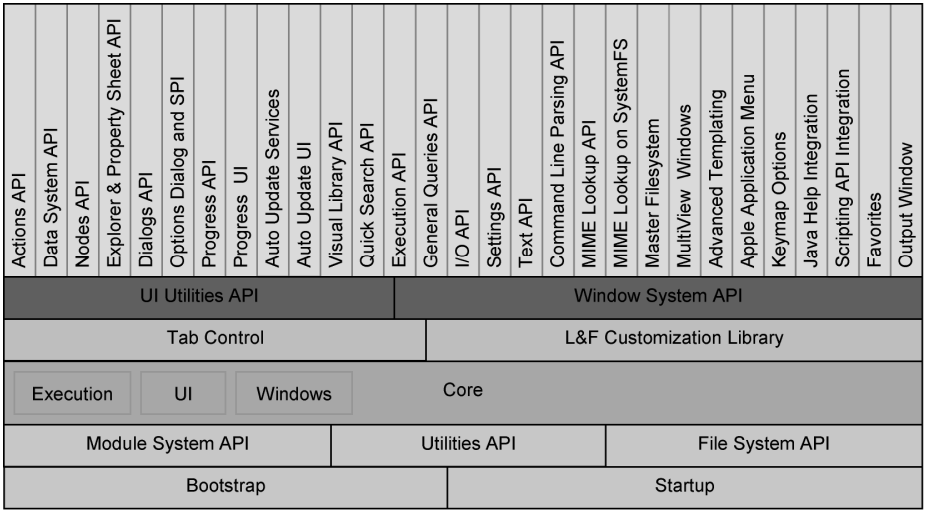


Obrázek 2.1: Základní struktura NetBeans IDE

Aby platforma NetBeans poskytla tak vysoký stupeň modularity vašim aplikacím, obsahuje mechanismy a principy, které pomohou vytvořit moduly rozšiřitelné dalšími moduly a umožní jim spolu komunikovat bez velké závislosti mezi sebou. Jinými slovy: platforma NetBeans podporuje slabou vazbu (*loose coupling*) modulů v aplikaci.

Aby se optimalizovalo zapouzdření kódu v modulech, které je nezbytné, poskytuje platforma NetBeans svůj systém zavaděčů tříd (classloaders). Každý modul je načten svým zavaděčem, a vytvoří tak oddělenou nezávislou jednotku kódu. Modul může explicitně zpřístupnit své rozhraní (a funkčnost) ostatním modulům. Aby mohl modul používat funkce jiných modulů, musí deklarovat závislost na těchto modulech. Tyto závislosti se zapisují do souboru manifestu modulu, běhové prostředí NetBeans si je odtud načte a postará se o to, aby aplikace nastartovala v konzistentním stavu. Nic víc. Je to základní deklarativní koncept slabé vazby v platformě NetBeans. Snahou je, aby se co možná nejvíce informací přesunulo do popisných a konfiguračních souborů a nevázat tyto akce do zdrojového kódu Javy. Modul je popsán ve svém manifestu a v dohodnutých souborech jazyka XML, takže není třeba přidávat nic dalšího. Z těchto informací platforma NetBeans zjistí informace o modulech, jejich umístění a o jejich požadavcích.

Vlastní platforma NetBeans je sestavena ze skupiny základních modulů, které jsou nutné pro start aplikace a uživatelského prostředí. Navíc má platforma NetBeans mnoho dalších API a služeb, které zjednoduší vývoj. Můžeme vyjmenovat např. Actions API, které zpřístupní třídy akcí, Nodes API a Options SPI, které pomohou jednoduše vytvořit a integrovat vlastní nastavení aplikace. Mimoto existuje množství použitelných komponent, třeba Okno výstupu (Output window) a Okno oblíbené (Favorites window).



Obrázek 2.2: Architektura NetBeans platformy

Distribuce platformy NetBeans

Distribuci platformy NetBeans obvykle nemusíte stahovat, protože je součástí NetBeans IDE, které je samo o sobě Rich Client aplikací postavenou nad platformou. Když vytváříte aplikaci v NetBeans IDE a vytvoříte její distribuci, vytáhne se platforma z NetBeans IDE, které používáte k vývoji. NetBeans IDE máte možnost registrovat více různých platforem. Pro tento účel můžete stáhnout oddělenou distribuci platformy NetBeans z oficiální stránky na adrese *http://platform.netbeans.org*.

Nyní se podíváme blíže na moduly, které tvoří distribuci platformy NetBeans:

- Moduly *org-netbeans-bootstrap*, *org-netbeans-core-startup*, *org-openide-filesystems*, *org-openide-modules* a *org-openide-util* tvoří běhový kontejner (*NetBeans Runtime Container*). Toto je jádro platformy NetBeans a odpovídá za instalaci a načtení modulů aplikace.
- Moduly *org-netbeans-core*, *org-netbeans-core-execution*, *org-netbeans-core-ui*, *org-netbeans-core-windows* poskytují základní funkčnost aplikace postavené nad platformou NetBeans.
- *org-netbeans-core-output2* je už připravený modul, který poskytuje okno pro výpis zpráv aplikace a práci s nimi. Více o tomto modulu najdete v kapitole 9.
- Modul *org-netbeans-core-multiview* je framework pro vícepohledová okna, jaká používá např. vizuální návrhář GUI (*Matisse GUI Builder*), a poskytuje API pro vytvoření různých pohledů na data.
- Modul *org-openide-windows* obsahuje *Window System API*. Je asi nejpoužívanějším API aplikace. Najdete zde základní třídy pro vývoj oken aplikace a (kromě jiných) také třídu WindowManager, která vám zpřístupní informace o všech oknech v aplikaci.
- Funkci aktualizace aplikace poskytuje modul *org-netbeans-modules-autoupdate-services*. Obsahuje vše potřebné pro nalezení, stažení a instalaci modulů do aplikace. Modul *org-netbeans-modules-autoupdate-ui* k tomu přidává Plugin Manager, ve kterém si může uživatel aktualizaci aplikace sám řídit.
- Modul *org-netbeans-modules-favorites* nabízí okno pro zvolení a uložení seznamu složek a souborů (třeba integrovaným výběrem souborů), které lze otvírat. Akce přidané složkám a souborům pomocí Data Systems API fungují i v tomto okně.
- Modul *org-openide-actions* vytváří řadu důležitých systémových akcí, které už máte k dispozici, např. *Copy*, *Paste* a *Print*. Každá může být implementovaná jako kontextově závislá.
- *org-openide-loaders* – velmi užitečný modul, který obsahuje *Data Systems API* pro integraci tříd pro načítání dat (datových loaderů) spojených s konkrétním typem dat.
- *Nodes API* v modulu *org-openide-nodes* obsahuje jeden ze základních konceptů platformy NetBeans, tzv. uzlů (*Nodes*). Uzel Node může být zobrazen v okně průzkumníka, má vlastní akce a podporuje seznam vlastností (property sheets).
- Modul *org-openide-explorer* poskytuje framework pro zobrazení pohledu (okna) průzkumníka, jako je třeba Okno projektů (Projects window) a Okno souborů (Files window) používané v NetBeans IDE.
- Modul *org-netbeans-modules-editor-mimelookup* poskytuje API pro nalezení nastavení, služeb a dalších objektů specifických pro daný typ souboru (MIME-typ) spolu se SPI, které umožní vytvoření poskytovatele dat daného MIME-typu. Modul *org-netbeans-modules-editor-mimelookup-impl* je speciální implementace tohoto SPI, kterou pro svoji práci používá System Filesystem.
- *org-netbeans-modules-javahelp* obsahuje běhovou knihovnu *JavaHelp* a zpřístupňuje její implementaci modulu Module System API, aby JavaHelp integroval a seskupil příspěvky všech modulů.
- Modul Master Filesystem *org-netbeans-modules-masterfs* poskytuje vaší aplikaci základní abstrakci (obal, wrapper) souborového systému.
- Modul *org-netbeans-modules-options-api* poskytuje Okno nastavení (Options window), které můžete přizpůsobit a pomocí SPI rozšířit o svá nastavení.
- Modul *org-netbeans-api-progress* vám umožní řídit dlouhotrvající úlohy. Modul *org-netbeans-modules-progress-ui* nabízí vizualizaci průběhu úlohy a možnost úlohu ručně ukončit.

- *org-netbeans-modules-queries* poskytuje obecné API pro prohledávání, abyste mohli zjistit informace o souborech zpracovávaných aplikací. Nad ním je SPI pro vytvoření vaší vlastní implementace dotazu.
- *org-netbeans-modules-sendopts* obsahuje *Command Line Parsing API a SPI*, u kterého si můžete registrovat vaši vlastní obsluhu (handler) pro zpracování argumentů příkazové řádky.
- Modul *org-netbeans-modules-settings* poskytuje API pro uložení nastavení modulu ve vlastním formátu a také několik připravených formátů pro uložení.
- *org-openide-awt* obsahuje *UI Utilities API*, které obsahuje mnoho užitečných tříd pro zobrazení komponent uživatelského rozhraní.
- Modul *org-openide-dialogs* poskytuje API pro zobrazení standardních i vlastních dialogů. Navíc obsahuje i Wizard Framework (framework pro vytváření průvodců).
- *org-openide-execution* umožní pomocí API asynchronní provádění dlouhotrvajících úloh.
- *org-openide-io* poskytuje API a SPI pro zobrazení vstupu a výstupu dat v aplikaci. Modul nabízí i standardní implementace, takže aplikace může zapisovat do svého okna výstupu (Output Window).
- *Text API* v modulu *org-openide-text* poskytuje rozšíření javax.swing.text API.
- Moduly *org-netbeans-swing-plaf* a *org-netbeans-swing-tabcontrol* odpovídají za Look & Feel a zobrazení záložek. Modul *org-jdesktop-layout* je obal *Swing Layout Extensions Library*.
- *org-netbeans-api-visual* obsahuje *Visual Library API*, knihovnu pro modelování a vizualizaci dat.
- Modul *org-netbeans-spi-quicksearch* obsahuje Quick Search API a SPI, které je v platformě NetBeans od verze 6.5. Poskytuje infrastrukturu pro implementaci poskytovatelů vyhledávání např. položek nabídek, akcí a souborů. Hlavní vyhledávací pole se zobrazí uživateli.

Do své aplikace můžete přidávat i ostatní moduly, např. i moduly se svojí distribuce NetBeans IDE.

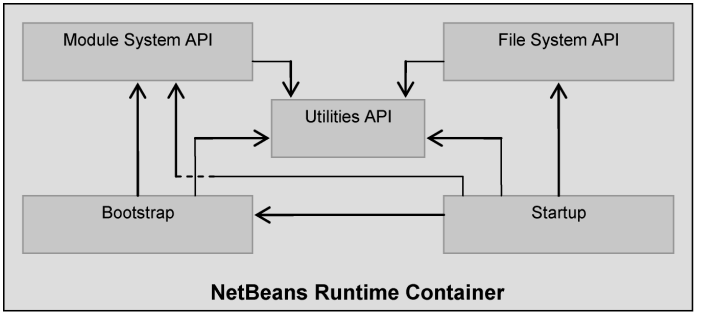
Běhový kontejner NetBeans

Základ platformy NetBeans a její modulární architektury se jmenuje běhový kontejner NetBeans (*NetBeans Runtime Container*). Skládá se z těchto 5 modulů:

- *Bootstrap* – tento modul je spuštěn dříve než všechny ostatní. Zavolá všechny registrované handlery pro příkazovou řádku a připraví počáteční zavaděč tříd (boot classloader), který načte modul Startup.
- *Startup* – nastartuje aplikaci, inicializuje modulový systém a modul souborového systému.
- *Module System* – odpovídá za správu modulů, jejich nastavení a závislosti.
- *File System* – zpřístupní virtuální datový systém s přístupem nezávislým na platformě. Zpočátku se použije k načtení zdrojů modulů do aplikace.
- *Utilities* – poskytuje základní komponenty např. pro komunikaci mezi moduly.

Na obrázku jsou zobrazeny závislosti mezi těmito pěti moduly.

Běhový kontejner je minimální skupina modulů, kterou aplikace platformy NetBeans vyžaduje pro svůj běh. Kromě ostatních modulů nebo nutných nastavení lze distribuovat aplikaci s těmito pěti moduly. Okamžitě se ovšem ukončí, protože nemá nadefinovány jiné další úkoly. Když běhový kontejner startuje, najde všechny dostupné moduly a vytvoří si jejich interní registr. Modul je načten jen jednou a jen tehdy, pokud je třeba. Modul má možnost spustit úlohu v okamžiku, kdy je načten běhovým kontejnerem. To je práce třídy instalátor modulu, o které si řekneme více v kapitole 3. Běhový kontejner také umožňuje dynamické načtení, odstranění, instalaci a odinstalování modulů za běhu. Tato funkčnost existuje proto, aby bylo možné aktualizovat aplikaci nebo deaktivovat nepotřebné moduly.



Obrázek 2.3: Běhový kontejner NetBeans

Pro úplné porozumění Rich Client aplikaci je důležité říci, že modul Bootstrap (první startovaný modul) je spuštěn na platformě závislým spouštěčem. Spouštěč je také zodpovědný za nalezení běhového prostředí Javy požadovaného pro běh aplikace a je částí platformy NetBeans a je specifický pro platformu, např. ve Windows je to soubor `.exe`.

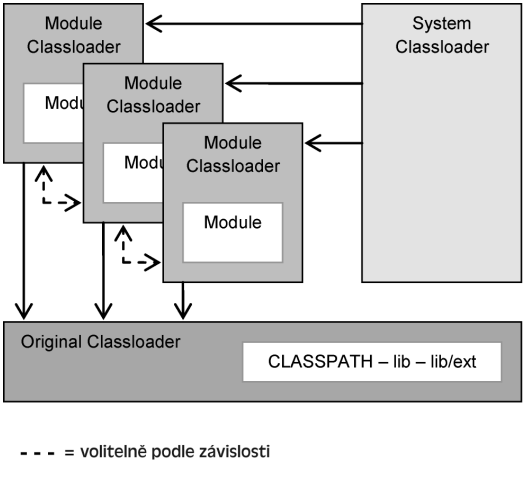
Systém zavaděčů tříd NetBeans

Classloader System platformy NetBeans je základní částí běhového kontejneru NetBeans a základním prvkem zapouzdření modulů a strukturování modulární architektury. Tento systém se skládá ze tří různých druhů classloaderu. Jsou to *Module Classloader*, *System Classloader* a *Original Classloader*. Většina tříd je načtena zavaděčem tříd modulu (Module Classloader). Jen v případech, kdy je třeba načíst zdroje mimo modul, se používá systémový zavaděč tříd (System Classloader). Prvotní zavaděč tříd (Original Classloader) načítá zdroje z classpath aplikačního spouštěče. Modulový a systémový zavaděč tříd mohou mít nekonečný počet zavaděčů tříd jako své rodiče, jsou to tzv. zavaděče s více rodiči (multi-parent). Vztahy mezi těmito typy zavaděčů tříd jsou na obrázku 2.4.

Modulový zavaděč tříd

Pro každý modul registrovaný v modulovém systému je vytvořen modulový zavaděč tříd, takže každý modul obdrží svůj vlastní jmenný prostor. Tento zavaděč v první řadě načítá třídy z archivu JAR modulu, ale také může načítat z více archivů, jak se to stává u obalových modulů knihoven. O nich se budete učit v kapitole 3.

Rodičem každého zavaděče tříd modulu je implicitně prvotní zavaděč tříd (Original Classloader) a je první v seznamu rodičů. Ostatní rodiče jsou zavaděče tříd modulů, na které je nastavena závislost. V kapitole 3 si popíšeme, jak se definuje závislost mezi moduly.



Obrázek 2.4: Systém zavaděčů tříd NetBeans

Zavaděč tříd modulu s více rodiči (multi-parent classloader) umožní třídám, aby byly načteny jiným modulem bez konfliktu jmenných prostorů. Načtení tříd je delegováno rodičovskému zavaděči. Kromě archivu JAR modulu je tento zavaděč odpovědný i za načtení lokálních

rozšíření (Local Extension Archives – viz kapitola 10) z podsložky locale a také opravných archivů (Patch Archives) v podsložce patches, pokud existuje.

Systémový zavaděč tříd

Systémový zavaděč tříd je také zavaděč tříd s více rodiči. Má odkaz na všechny instance zavaděčů tříd modulů, které má jako rodiče. Výsledkem je, že může teoreticky načíst všechny soubory poskytnuté modulem. V zájmu zapouzdření je však doporučeno tento přístup použít jen tehdy, pokud je to absolutně nezbytné.

Přístup k systémovému zavaděči tříd můžete získat dvěma způsoby: pomocí Lookup, kterým se budeme mnohem více zabývat později, nebo kontextovým zavaděčem tříd aktuálního vlákna. Tím je (kromě případů, kdy jste nastavili kontextový zavaděč tříd explicitně) systémový zavaděč tříd.

```
ClassLoader cl = (ClassLoader) Lookup.getDefault().lookup(ClassLoader.class);
```

nebo

```
ClassLoader cl = Thread.currentThread().getContextClassLoader();
```

Prvotní zavaděč tříd

Prvotní (aplikační) *zavaděč tříd* používá spouštěč aplikace. Načte třídy a ostatní zdroje v původní *CLASSPATH* a potom z adresářů lib a jejich podadresářů ext. Když archiv JAR není rozpoznán jako modul (např. pokud jsou chybné položky manifestu), není přidán do modulového systému. Tyto zdroje se vždy prohledávají nejdříve. Pokud se ty stejné zdroje naleznou zde a v JARu modulu, jsou ty z modulu ignorovány. Toto uspořádání je důležité pro přizpůsobení (branding) modulů a přípravu vícejazyčných distribucí modulu.

Také tento zavaděč tříd není určen pro načítání všech souvisejících zdrojů. Jeho úkolem je načíst zdroje nutné v rané fázi startu aplikace, např. třídy potřebné pro nastavení Look & Feels.

Shrnutí

V této kapitole jsme prozkoumali strukturu platformy NetBeans. Začali jsme nahlédnutím na její architekturu a jádro, které poskytuje běhový kontejner. Běhový kontejner poskytuje výkonné prostředí pro aplikace postavené nad platformou NetBeans a také infrastrukturu pro modulární aplikaci. Ukázali jsme si, jak systém zavaděčů tříd NetBeans zabezpečí zapouzdření modulu. Krátce jsme popsali části platformy a nejpoužívanější moduly. Nakonec jsme ukázali, že samo NetBeans IDE je vlastně Rich Client aplikace sestávající z modulů. I jeho moduly můžete použít ve své aplikaci.

KAPITOLA 3

Systém pro správu modulů

V této kapitole:

- Úvod
- Struktura modulu
- Typy modulů
- Manifest modulu
- Konfigurační soubor modulu – layer.xml
- Vytváření modulů
- Verzování a závislosti
- Životní cyklus modulu
- Registr modulů
- Používání knihoven
- Shrnutí

Pochopte základní stavební kameny!

Tato kapitola se zaměřuje na modul Systém pro správu modulů v NetBeans (zkráceně modulový systém), který je základní komponentou běhového kontejneru a zodpovídá za načtení a správu všech modulů aplikace. Naučíte se, jak je modul strukturován, jak je integrován do platformy NetBeans a jaký je jeho životní cyklus.

Úvod

NetBeans Module System zodpovídá za správu všech modulů aplikace a také za úkoly, jako je například vytvoření zavaděče tříd, načtení modulů a jejich aktivace a deaktivace. Jeho koncepce je, jak jen to bylo možné, založena na stadardních technologiích Javy. Základní formát modulu pochází ze standardního mechanizmu rozšíření platformy Java (Java extension mechanism). K popisu a správě závislostí mezi moduly byly využity hlavní myšlenky verzování balíčků jazyka Java (Package Versioning Specification).

Základní vlastnosti, např. popis modulu a závislostí na jiném modulu, jsou obsaženy v souboru `manifest`. Do něho se přidávají další atributy specifické pro NetBeans. Pro návrh specifikace modulu se využívá Java Activation Framework a také vnitřní funkce JDK (např. podpora spustitelných souborů JAR). Kromě atributů v manifestu většina modulů nepotřebuje další instalační kód, protože se do platformy NetBeans přidávají deklarativně. XML soubor `layer.xml` obsahuje informace pro aplikaci a definuje integraci modulu do platformy NetBeans. V tomto souboru je vše, co modul přidává do platformy, a to od akcí v nabídce až po služby.

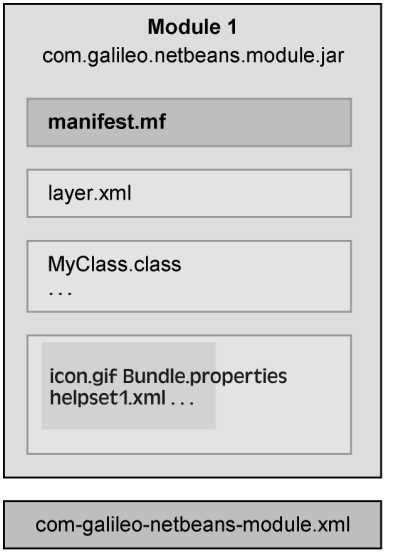
Struktura modulu

Modul je obyčejný JAR soubor, který obsahuje tyto části:

- soubor manifest (`manifest.mf`);
- soubor layer.xml (`layer.xml`);
- soubory class;
- zdroje jako ikony, properties, soubory lokalizace, nápovědu atd.

Povinný je jen soubor manifestu, jelikož je v něm identifikace modulu a deklaruje JAR jako modul NetBeans. Ostatní závisí na účelu modulu. Pokud je modul použitý jako knihovna, je zbytečný i soubor `layer.xml`.

Ke každému modulu náleží konfigurační XML soubor (pro modul na obrázku 3.1 `com-galileo-netbeans-module.xml`), který je umístěný vně JARu. Je to prvně načtený soubor informací o modulu, tj. oznámí platformě existenci modulu.



Obrázek 3.1: Modul NetBeans

Typy modulů

Všechny moduly jsou deklarovány modulovému systému v konfiguračních XML souborech umístěných v klastru „platforma“ ve složce `config/Modules`, tedy vně vlastního modulu. Tuto složku načte systém pro správu modulů a podle těchto informací nahraje moduly při startu aplikace. Konfigurační soubor popisuje jméno modulu, verzi a umístění (`name`, `version` a `location`) a také zda a jak se má modul načíst. Podívejte se na následující obsah souboru:

Výpis 3.1: Konfigurační soubor modulu – com-galileo-netbeans-module.xml

```
<module name="com.galileo.netbeans.module">
   <param name="autoload">false</param>
   <param name="eager">false</param>
   <param name="enabled">true</param>
   <param name="jar">modules/com-galileo-netbeans-module.jar</param>
   <param name="reloadable">false</param>
   <param name="specversion">1.0</param>
</module>
```

Parametr `enabled` říká, zda se má modul načíst. Jsou tři možné způsoby, jak má být modul načten. Když jsou oba parametry `autoload` a `eager` nastaveny na false, je typ modulu `Regular`. Pokud má jeden z parametrů hodnotu true, je modul typu `Autoload` nebo `Eager`. Typ modulu se zadává ve vlastnostech modulu (Module properties z místní nabídky) ve skupině API-Versioning (podívejte se na obrázek 3.7). Standardně je modul typu `Regular`.

Regular

Je to nejběžnější typ modulu aplikace. Načte se při startu aplikace. Doba startu aplikace se prodlouží o dobu inicializace modulu. Proto je třeba, abychom inicializaci modulu zajistili co nejkratší. Obvykle není potřeba provést vše při načtení modulu. Mnoho úkolů může být také definováno deklarativně.

Autoload

Tento typ modulu je načten jen tehdy, když ho jiný modul vyžaduje. Využívá se zde principu odloženého (lazy) načtení. Tento typ modulu obvykle používají jiné moduly jako knihovnu.

Eager

Moduly Eager jsou načteny jen tehdy, když jsou splněny všechny závislosti. Je to další způsob, jak zkrátit čas spuštění. Například: když modul X závisí na modulech A a B, které nejsou dostupné, není důvod načítat modul X.

Manifest modulu

Každý modul v platformě NetBeans má soubor manifest. Je to textový popis modulu a jeho prostředí. Při načtení modulu je to první soubor čtený z modulu systémem pro správu modulů. Modul NetBeans je rozpoznán podle toho, zda manifest obsahuje atribut `OpenIDE-Module`. Je to jediný povinný atribut. Jeho hodnota je libovolý identifikátor, ale obvykle se používá jméno bázového balíčku (**Code Name Base** v průvodci pro vytvoření modulu), např. `com.galileo.netbeans.module`, aby se předešlo konfliktu jmen (mohl by nastat, pokud by byl vytvořen různými vývojáři). Tento identifikátor se používá k jednoznačnému rozlišení modulu při aktualizaci nebo při deklaraci závislostí.

Atributy

Dále jsme popsali stručně všechny atributy v manifestu i s krátkým příkladem.

Popis modulu

Tyto atributy souboru manifestu textovou formou popisují integraci modulu do platformy.

- OpenIDE-Module: Definuje jednoznačné jméno modulu, pod kterým ho systém pro správu modulů zná. Atribut je povinný.

```
OpenIDE-Module: com.galileo.netbeans.module
```

- OpenIDE-Module-Name: Název modulu určený pro zobrazení uživateli. Zobrazuje ho i Plugin Manager.

```
OpenIDE-Module-Name: Můj první modul
```

- OpenIDE-Module-Short-Description: Krátký popis funkce modulu.

```
OpenIDE-Module-Short-Description: Krátký popis mého prvního modulu
```

- OpenIDE-Module-Long-Description: Dlouhý popis funkce modulu. Text je zobrazen v Plugin Manageru. Vyplnění tohoto atributu se doporučuje, protože informuje uživatele a vlastnostech a funkcích modulu.

```
OpenIDE-Module-Long-Description:
  Zde můžete vypsat delší popis modulu v několika větách.
  Můžete vysvětlit, co modul umí.
```

- OpenIDE-Module-Display-Category: Kategorie modulu. Moduly jsou seskupeny do virtuálních skupin, a ty se prezentují uživateli jako funkční jednotka.

```
OpenIDE-Module-Display-Category: Moje moduly
```

- OpenIDE-Module-Install: Určuje třídu, která vykoná inicializační akce v průběhu životního cyklu modulu, tzv. instalátor modulu (Module Installer) (viz část „Životní cyklus modulu“).

```
OpenIDE-Module-Install: com/galileo/netbeans/module/ModuleLifecycle.class
```

- OpenIDE-Module-Layer: Jeden z nejdůležitějších atributů. Určuje cestu k souboru layer.xml (viz část „Soubor layer.xml“), ve kterém je popis integrace modulu do platformy.

```
OpenIDE-Module-Layer: com/galileo/netbeans/module/resources/layer.xml
```

- OpenIDE-Module-Public-Packages: Aby se zajistilo zapouzdření, přístup ke třídám modulu je standardně zakázán. Tento atribut deklaruje, které balíčky (package) modulu jsou veřejné, a dovolí ostatním modulům používat jejich třídy. Nezbytný je u knihoven.

```
OpenIDE-Module-Public-Packages:
   com.galileo.netbeans.module.actions.*,
   com.galileo.netbeans.module.util.*
```

- OpenIDE-Module-Friends: Pokud balíčky tohoto modulu mohou využívat jen některé balíčky (které musí být veřejné-public), vypište je zde.

```
OpenIDE-Module-Friends:
  com.galileo.netbeans.module2,
  com.galileo.netbeans.module3
```

- OpenIDE-Module-Localizing-Bundle: Soubor properties, který se používá k lokalizaci (viz kapitola 8).

```
OpenIDE-Module-Localizing-Bundle:
  com/galileo/netbeans/module/resource/Bundle.properties
```

Verzování a závislosti

Následující atributy definují verze a závislosti. Jejich popis a použití je popsáno dále v sekci „Verzování a závislosti".

- OpenIDE-Module-Module-Dependencies: Definuje závislosti mezi moduly. Také je možné určit nutnou minimální verzi modulu.

```
OpenIDE-Module-Module-Dependencies:
  org.openide.util > 6.8.1,
  org.openide.windows > 6.5.1
```

- OpenIDE-Module-Package-Dependencies: Modul může být také závislý na určitém balíčku. To říká právě tento atribut.

```
OpenIDE-Module-Package-Dependencies:
  com.galileo.netbeans.module2.gui > 1.2
```

- OpenIDE-Module-Java-Dependencies: Když modul vyžaduje určitou verzi Javy, nastavte ji zde.

```
OpenIDE-Module-Java-Dependencies: Java > 1.5
```

- OpenIDE-Module-Specification-Version: Atribut definuje verzi specifikace modulu. Obvykle se zapisuje v Dewey desítkovém formátu.

```
OpenIDE-Module-Specification-Version: 1.2.1
```

- OpenIDE-Module-Implementation-Version: Nastavuje verzi implementace modulu, obvykle jako časovou značku. Toto číslo se mění při každé změně modulu.

```
OpenIDE-Module-Implementation-Version: 200701190920
```

- OpenIDE-Module-Build-Version: Tento atribut je nepovinný a systém pro správu modulů ho ignoruje. Je dobrý pro popis, obvykle se používá datová značka.

```
OpenIDE-Module-Build-Version: 20070305
```

- OpenIDE-Module-Module-Dependency-Message: Text, který se zobrazuje, když nemohou být splněny závislosti modulu. Někdy postačí ponechat závislosti nedořešené. Pak je vhodné uživatele informovat zprávou, které moduly jsou potřeba a kde je nalezne.

```
OpenIDE-Module-Module-Dependency-Message:
  Je porušena závislost modulu. Potřebný modul xxx stáhněte z následující URL.
```

- OpenIDE-Module-Package-Dependency-Message: Zpráva vypsaná, pokud není splněna závislost na potřebném balíčku.

```
OpenIDE-Module-Package-Dependency-Message:
  Je narušena závislost na balíčku. Řešení je...
```

- OpenIDE-Module-Deprecated: Tento parametr označuje, že je tento modul zastaralý a bude se v budoucnu rušit. Při načtení modulu se v logu objeví varování.

```
Open(lazy)IDE-Module-Deprecated: true
```

- OpenIDE-Module-Deprecation-Message: Do tohoto nepovinného atributu zadejte zprávu, která se přidá do logu mezi varování o zastaralých vazbách. Informuje uživatele o náhradě zastaralého modulu. Zobrazuje se jen tehdy, když je parametr `OpenIDE-Module-Deprecated` nastaven na true.

```
OpenIDE-Module-Deprecation-Message:
"Modul 1" je zastaralý a nebude už podporován. Použijte "Modul 3".
```

Služby a rozhraní

Další atributy se používají pro definování závislosti na implementaci služby (service) a konkerétním rozhraní poskytovatele (provider). Podrobnější informace o tomto tématu najdete v kapitole 6.

- OpenIDE-Module-Provides: Deklaruje rozhraní služby (service interface), jehož implementaci modul poskytuje.

```
OpenIDE-Module-Provides: com.galileo.netbeans.spi.ServiceInterface
```

- OpenIDE-Module-Requires: Deklarace rozhraní služby, kterou modul vyžaduje. Nezáleží na tom, který modul implementaci služby poskytne.

```
OpenIDE-Module-Requires: org.openide.windows.IOProvider
```

- OpenIDE-Module-Needs: Tento atribut je slabší variantou atributu Requires a nevyžaduje pořadí modulů. To může být užitečné pro moduly API, které vyžadují specifickou implementaci.

```
OpenIDE-Module-Needs: org.openide.windows.IOProvider
```

- OpenIDE-Module-Recommends: Tímto atributem můžete zavést volitelnou závislost. Pokud např. není přítomen poskytovatel java.sql.Driver, může se modul přesto načíst.

```
OpenIDE-Module-Recommends: java.sql.Driver
```

- OpenIDE-Module-Requires-Message: Jako podobné atributy, tento definuje zprávu, pokud není vazba splněna.

```
OpenIDE-Module-Requires-Message:
   Není dostupný poskytovatel služby. Pro více informací jděte na ...
```

Moduly závislé na platformě

Atribut OpenIDE-Module-Requires umožní i definici závislosti na modulech podle operačního systému. Tento atribut testuje přítomnost daného textu (tokenu). Jsou připraveny tyto tokeny:

```
org.openide.modules.os.Windows
org.openide.modules.os.Linux
org.openide.modules.os.Unix
org.openide.modules.os.PlainUnix
org.openide.modules.os.MacOSX
org.openide.modules.os.OS2
org.openide.modules.os.Solaris
```

Systém pro správu modulů se přesvědčí, zda je přítomen token běžícího operačního systému. Například: Když máte modul, který se automaticky načte ve Windows a automaticky deaktivuje na ostatních operačních systémech, nastavte typ modulu jako Eager a přidejte do souboru manifestu tento řádek:

```
OpenIDE-Module-Requires: org.openide.modules.os.Windows
```

Viditelnost

Další parametry nastavují viditelnost modulu v Plugin manažeru. Tak jsou moduly v manažeru zobrazeny jasně a jednoduše.

- AutoUpdate-Show-In-Client: Nastavením atributu na „true“ nebo „false“ určíte, zda se má modul zobrazit v manažeru modulů.

  ```
  AutoUpdate-Show-In-Client: true
  ```

- AutoUpdate-Essential-Module: Hodnoty jsou „true“ nebo „false“. True říká, že tento modul je nepostradatelný pro aplikaci a nemůže být deaktivován nebo odinstalován.

  ```
  AutoUpdate-Essential-Module: true
  ```

Ve spojení s těmito parametry byl ještě od platformy NetBeans verze 6.5 zaveden princip „kit modulů“ – modul soupravy (funkční skupiny). Každý modul viditelný ve správci modulů (AutoUpdate-Show-In-Client: true) je považován za kit-modul. Všechny moduly, na které kit-modul definuje závislost, jsou spravovány stejným způsobem, s výjimkou nezobrazených modulů, které mají závislost na jiném kit-modulu. Např. když je kit-modul deaktivován, všechny závislé moduly jsou také deaktivovány.

To umožní vytvořit obalovací modul (wraper module) k seskupení několika souvisejících modulů, aby se uživateli zobrazily jako jediná jednotka. Můžete vytvořit prázdný modul s atributem AutoUpdate-Show-In-Client nastaveným na true a definujete závislost na modulech, které se mají seskupit. U těchto modulů nastavte atribut na false.

Příklad

Soubor manifestu s několika typickými atributy:

Výpis 3.2: Příklad manifestu

```
OpenIDE-Module: com.galileo.netbeans.module
OpenIDE-Module-Public-Packages: -
OpenIDE-Module-Module-Dependencies:
   com.galileo.netbeans.module2 > 1.0,
   org.jdesktop.layout/1 > 1.4,
   org.netbeans.core/2 = 200610171010,
   org.openide.actions > 6.5.1,
   org.openide.awt > 6.9.0,
OpenIDE-Module-Java-Dependencies: Java > 1.5
OpenIDE-Module-Implementation-Version: 200701100122
OpenIDE-Module-Specification-Version: 1.3
OpenIDE-Module-Install: com/galileo/netbeans/module/Install.class
OpenIDE-Module-Layer: com/galileo/netbeans/module/layer.xml
OpenIDE-Module-Localizing-Bundle: com/galileo/netbeans/module/Bundle.properties
OpenIDE-Module-Requires:
   org.openide.windows.IOProvider,
   org.openide.modules.ModuleFormat1
```

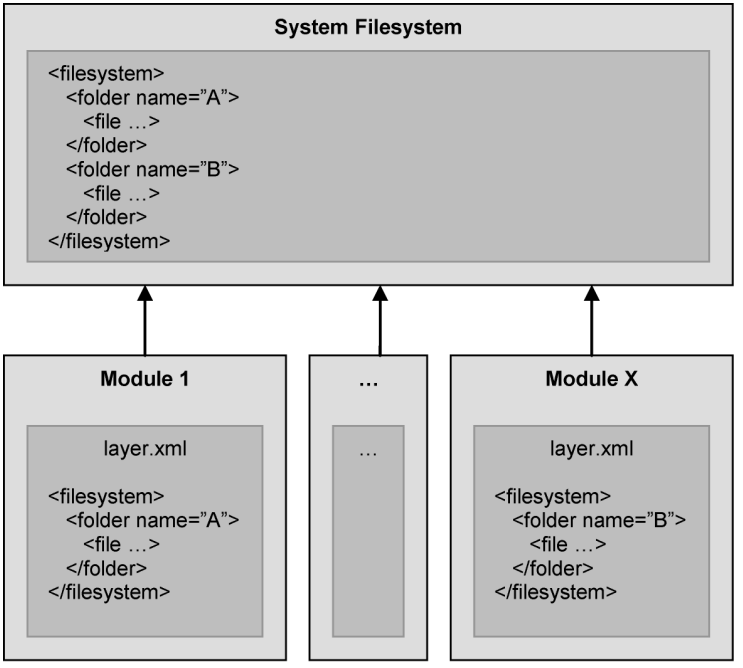
Konfigurační soubor modulu – layer.xml

V manifestu jsou popsána rozhraní, závislosti a prostředí modulu. Kromě něho modul obsahuje i soubor `layer.xml`. Je to základní konfigurační soubor, ve kterém je virtuálně definováno vše, co modul přidává do platformy NetBeans. Částečně na něj lze nahlížet jako na rozhraní mezi modulem a platformou NetBeans, které deklarativně popisuje integraci modulu do platformy.

Existence souboru layer je nastavena v manifestu atributem `OpenIDE-Module-Layer`, který definuje cestu k souboru layer, obvykle pojmenovaném `layer.xml`.

```
OpenIDE-Module-Layer: com/galileo/netbeans/module/layer.xml
```

Formátem souboru je hierarchický souborový systém se složkami, soubory a atributy. Během startu aplikace jsou informace ze všech souborů layer sloučeny do virtuálního souborového systému. Tím je systémový souborový systém (**System Filesystem**) – konfigurace platformy NetBeans za běhu.



Obrázek 3.2: System Filesystem

Tento soubor layer obsahuje několik standardních složek. Ty jsou definovány různými moduly, jsou to body rozšíření (**Extension Points**). Například základní složka `Menu` vypadá takto:

Výpis 3.3: Standardní složka souboru layer

```
<folder name="Menu">
   <folder name="Edit">
      <file name="MyAction.shadow">
         <attr name="originalFile"
            stringvalue=
              "Actions/Edit/com-galileo-netbeans-module-MyAction.instance"/>
      </file>
   </folder>
</folder>
```

Vidíte, že akce `MyAction` je přidána do nabídky **Edit**. Nyní se nestarejte o přesnou syntaxi, o té se hovoří v dalších kapitolách. Nejdříve rozebereme základní strukturu souboru layer. Navíc NetBeans nabízí prostředky pro práci se souborem layer, jak uvidíme v následujících kapitolách, kde vytvoříme svůj první modul. Seznam důležitých bodů rozšíření je uveden v Příloze.

Tímto způsobem může každý modul přidat novou položku do nabídky nebo vytvořit nový panel nástrojů (toolbar). Nabídka se vytvoří podle toho, jak se smíchají informace souborů layer všech modulů do systémového souborového systému. Systém oken (Window System), zodpovědný za

generaci nabídek, jen přečte složku navigační nabídky ze systémového souborového systému, aby získal obsah nabídek.

Uložení informací do systémového souborového systému umožňuje také to, že se moduly dají přidávat a odebírat za běhu. V souborovém systému můžete registrovat posluchače. To také systém oken dělá a upraví si obsah nabídky, jakmile dojde po načtení modulu ke změně.

Pořadí položek

Pořadí, ve kterém se položky souboru layer načítají (a také zobrazují v nabídce), se definuje atributem `position`:

Výpis 3.4: Určení pořadí položek v souboru layer

```
<filesystem>
  <folder name="Menu">
    <folder name="Edit">
      <file name="CopyAction.shadow">
        <attr name="originalFile"
            stringvalue=
                "Actions/Edit/org-openide-actions-CopyAction.instance"/>
        <attr name="position" intvalue="10"/>
      </file>
      <file name="CutAction.shadow">
        <attr name="originalFile"
            stringvalue=
                "Actions/Edit/org-openide-actions-CutAction.instance"/>
        <attr name="position" intvalue="20"/>
      </file>
    </folder>
  </folder>
</filesystem>
```

Takže akce `Copy` se zobrazí před akcí `Cut`. Pokud potřebujete, můžete tento atribut použít i u složek. V praxi se používají pozice s velkými rozestupy, aby se umožnilo pozdější vkládání položek. Když je nějaká pozice použita dvakrát, vypíše se v logu varování.

Aby se dalo pořadí v souboru layer nastavovat pohodlně, nabízí NetBeans IDE v okně projektů (Projects window) tzv. strom souboru layer (*layer tree*), ve kterém jsou zobrazeny položky souboru layer. V něm můžete určovat pořadí pomocí metody táhni a pusť. IDE pak atribut nastaví.

Po vytvoření modulu v další části „Vytváření modulů“ je ukázka, kde najdete strom layer tree. Pořadí akcí nastavíte v průvodci při vytváření akce (viz kapitola 4). Odpovídající atributy vytvoří průvodce.

Pozice položek ve stromu layer tree můžete také přepsat a můžete vložit další položky. Pak se odpovídajícím způsobem přepíší standardní pozice položek. Pozice existujících položek (třeba i položek pocházejících od modulů NetBeans) se změní (přepíše) takto:

```
<attr name="Menu/Edit/CopyAction.shadow/position" intvalue="15"/>
```

Použijte kompletní cestu k souboru před jménem atributu.

Soubory .instance

Soubory typu `.instance` v systémovém souborovém systému (System Filesystem) popisují objekty, jejichž instanci chceme vytvořit. Jméno souboru je obvykle plně kvalifikované jméno třídy java objek-

tu (např. `com-galileo-netbeans-module-MyAction.instance`), jejíž výchozí konstruktor nebo statická metoda vytvoří instanci. Instance je vytvořena moduly `File Systems` a `Data Systems API` takto:

```
public static Object getInstance(String name) {
   FileSystem f = Repository.getDefault().getDefaultFileSystem();
   FileObject o = f.getRoot().getFileObject(name);
   DataObject d = DataObject.find(o);
   InstanceCookie c = d.getCookie(InstanceCookie.class);
   return (c.instanceCreate());
}
```

Pro použití vhodnějšího jména pro instanci můžete plné jméno třídy definovat atributem `instanceClass`. Pak můžete používat mnohem kratší jména:

```
<file name="MyWindow.instance">
   <attr name="instanceClass"
       stringvalue="com.galileo.netbeans.module.MyWindow"/>
</file>
```

U tříd, které nemají bezparametrický výchozí konstruktor, vytvořte instanci statickou metodou a jméno této metody zapište do atributu `instanceCreate`:

```
<file name="MyWindow.instance">
   <attr name="instanceCreate"
           methodvalue="com.galileo.netbeans.module.MyWindow.getDefault"/>
</file>
```

Pokud má metoda `getDefault()` jako parametr `FileObject`, je metodě předána instance `FileObject`. Z objektu `FileObject` můžete načíst své atributy. Atributem také můžete zadat cestu k ikoně nebo jinému zdroji v souboru layer:

```
<file name="MyWindow.instance">
   <attr name="instanceCreate"
           methodvalue="com.galileo.netbeans.module.MyWindow.getDefault"/>
   <attr name="icon" urlvalue="nbres:/com/galileo/icon.gif"/>
</file>
```

Metoda `getDefault()`, která vytvoří instanci třídy `MyWindow`, vypadá takto:

```
public static MyWindow getDefault(FileObject obj) {
   URL url = (URL) obj.getAttribute("icon");
   ...
   return(new MyWindow(...));
}
```

Všimněte si, že jsme specifikovali cestu jako typ `urlvalue`, takže se přímo vrátí instance `URL`. Kromě už známých typů atributu `stringvalue`, `methodvalue` a `urlvalue` je ještě několik dalších. Najdete je v popisu Filesystem DTD (*http://netbeans.org/dtds/filesystem-1_2.dtd*).

Jednu nebo i více instancí určitého typu můžete vytvořit pomocí `Lookup`. Je to vhodnější než pomocí `InstanceCookie` jako v předešlém případě. Proto vytvoříme `Lookup` pro danou složku v systémovém souborovém systému (NetBeans) a voláním metody `lookup()` nebo `lookupAll()` získáme jednu nebo více instancí (pokud je jich definováno více).

```
Lookup lkp = Lookups.forPath("MyInstanceFolder");
Collection<? extends MyClass> c = lkp.lookupAll(MyClass.class);
```

Takto použijeme lookup v kapitole 5, abychom rozšířili místní nabídku top komponenty o své akce ze souboru layer.

Bázová třída nebo rozhraní mohou být definované v souboru layer atributem `instanceOf`. To umožní lookupu efektivnější práci a umožní mu určit, ze které třídy dědí nebo které rozhraní implementuje. Lookup tak může vytvořit jen instance žádaného typu.

Pokud třída `MyAction` z předchozího příkladu implementuje rozhraní `Action`, doplníme položku takto:

```
<file name="com-galileo-netbeans-module-MyAction.instance">
   <attr name="instanceOf" stringvalue="javax.swing.Action"/>
</file>
```

Soubory .shadow

Soubor `.shadow` je druh odkazu (linku nebo reference) na soubor `.instance`. Užívá se hlavně u instancí singletonů (jedináčků), třeba u akcí. Akce je definována ve složce Actions pomocí souboru `.instance` a položka ve složce `Menu` nebo `Toolbars` se pak odvolává na akci pomocí souboru `.shadow`. Soubor `.shadow` se může odkazovat na soubory v systémovém souborovém systému a také na soubory na disku. Tímto způsobem ukládá své odkazy i modul Favorites. Cesta k souboru `.instance` se zadává v atributu `originalFile`.

Výpis 3.5: Propojení souboru .shadow se souborem .instance

```
<folder name="Actions">
  <folder name="Window">
    <file name="com-galileo-netbeans-module-MyAction.instance"/>
  </folder>
</folder>
<folder name="Menu">
  <folder name="Window">
    <file name="MyAction.shadow">
      <attr name="originalFile"
          stringvalue=
              "Actions/Window/com-galileo-netbeans-module-MyAction.instance"/>
    </file>
  </folder>
</folder>
```

Soubory .settings

Soubory `.settings` jsou rozšířenou verzí souboru `.instance` v souboru layer. Informace o typu a o způsobu vytvoření instance se definují v odděleném XML souboru. Základní rozdíl oproti souboru `.instance` je, že ve zvláštním XML souboru může být vypsána kompletní třídní hierarchie a všechna implementovaná rozhraní.

Tyto soubory `.settings` se používají např. pro `TopComponenty` (viz kapitola 5). XML soubor vypadá takto:

Výpis 3.6: Informace v souboru .settings

```
<!DOCTYPE settings PUBLIC
   "-//NetBeans//DTD Session settings 1.0//EN"
   "http://www.netbeans.org/dtds/sessionsettings-1_0.dtd">
```

```
<settings version="1.0">
 <module name="com.galileo.netbeans.module" spec="1.0"/>
 <instanceof class="javax.swing.JComponent"/>
 <instanceof class="org.openide.windows.TopComponent"/>
 <instanceof class="com.galileo.netbeans.module.MyTopComponent"/>
 <instance class="com.galileo.netbeans.module.MyTopComponent"
    method="getDefault"/>
</settings>
```

V souboru layer se odkážeme na tento soubor atributem `url`, kde zadáme relativní cestu k souboru:

```
<folder name="Windows2">
   <folder name="Components">
      <file name="MyTopComponent.settings" url="MyTopComponentSettings.xml"/>
   </folder>
</folder>
```

Vytvoření a užití vlastního obsahu

Váš modul může využívat složky, soubory a atributy souboru layer, aby poskytl body rozšíření (extension points) ostatním modulům. Načtení položek můžete provést požadavkem na systémový souborový systém:

```
FileUtil.getConfigRoot();
```

Tato metoda vrátí objekt `FileSystem`. Jeho vlastnosti a funkce více popisujeme v kapitole 7. Sekce „Okno – TopComponent“ v kapitole 5 ukazuje, jak definovat vlastní položky v souboru layer, jak se mohou načíst a jak mohou poskytnout bod rozšíření ostatním modulům tak, že je zpřístupní ostatním modulům.

Vytváření modulů

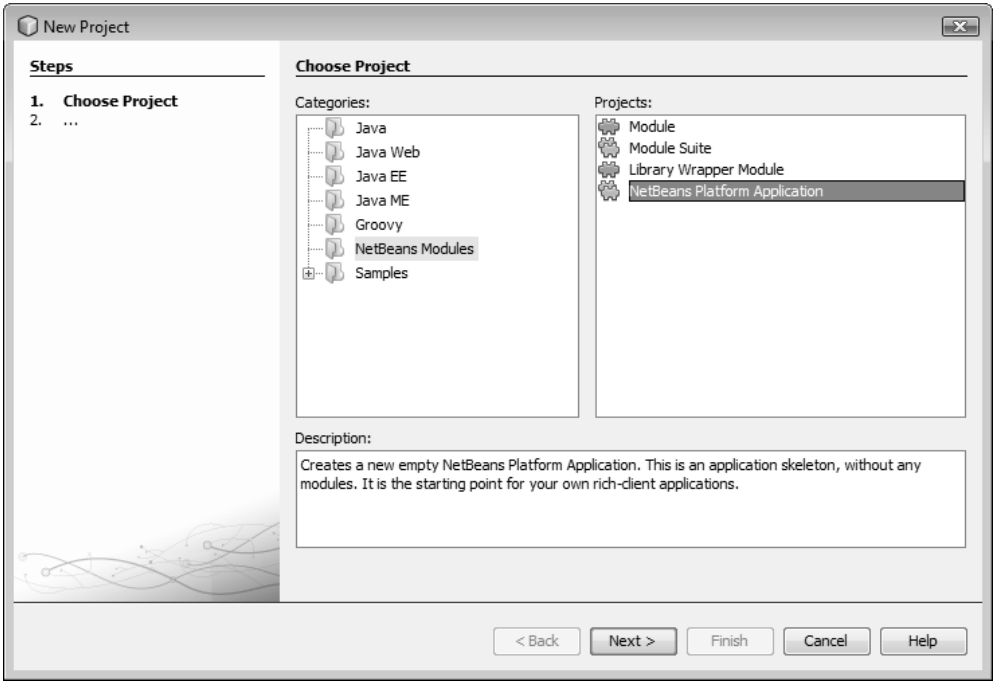
Po úvodním seznámení se strukturou a obsahem modulu vytvoříme svůj první modul. Pěkným příkladem vytvoření modulu je ukázková aplikace integrovaná v NetBeans IDE. Zpočátku navrhneme jen jednoduchý modul.

Nejprve vytvoříme projekt NetBeans aplikace nebo sady modulů (Module suite). Oba typy modulu jsou kontejnery na moduly. Projekt NetBeans Platform Application vytvoří samostatnou Rich Client aplikaci, která startuje jen moduly platformy NetBeans (a požadované moduly). Projekt Module Suite vytvoří skupinu souvisejících modulů, kterou spustíte v kompletní kopii NetBeans IDE. Z Module Suite můžete také vytvořit samostatnou aplikaci. Musíte jen ve vlastnostech projektu **Project Properties**, přístupných z místní nabídky projektu, na záložce **Build** vybrat volbu **Create Standalone Application**. Pokud mají být vynechány moduly typické pro IDE, zvolte **Exclude**. Pak zbydou jen moduly z platformy NetBeans. Tyto moduly vidíte v záložce **Libraries**. Když budete pokračovat ve vytváření aplikace, budou přístupné jen moduly, které budou nastaveny jako nutné pro běh aplikace. Ovšemže můžete přidat i moduly, které nejsou součástí platformy NetBeans.

NetBeans IDE poskytuje průvodce pro vytvoření projektů. Spusťte NetBeans IDE a vyberte **File > New Project**. Dialog zobrazuje několik kategorií projektu. Vyberte **NetBeans Modules**. Pak zvolte v pravé polovině okna typ projektu **NetBeans Platform Application**.

Na další stránce pojmenujte projekt, třeba `My Application`, zvolte umístění, kam se má projekt uložit. Poslední pole může zůstat prázdné. Po stisknutí tlačítka **Finish** se vytvoří projekt aplikace.

A teď můžete vytvořit první modul. To je úkol pro dalšího průvodce. Následujte příkaz **File > New Project**, zvolte kategorii **NetBeans Modules** a typ projektu `Module`. Tlačítkem **Next** přejděte na další stránku. Zapište jméno projektu, třeba `My Module`, vyberte volbu **Add to Module Suite** a vyberte právě vytvořenou aplikaci **NetBeans Platform Application** nebo **Module Suite** v seznamu. Na poslední stránce definujte bázový balíček (code name base) a zobrazované jméno modulu. Připravené defaultní hodnoty pro `Localizing Bundle` a `XML layer` soubor ponechte. Tlačítkem **Finish** ukončíte průvodce, a ten vygeneruje modul.



Obrázek 3.3: Vytvoření projektu Aplikace nad platformou NetBeans