AFSwinx & AFRest

Uživatelská příručka

Verze: 1.0.0.

Martin Tomášek

martin@toms-cz.com

7.12.2014

# Úvod

AFSwinx a AFRest jsou frameworky umožňující generování uživatelského rozhraní. Generování je prováděno na serverové straně a interpretování na klientovi. K přenosu lze využít například služby typu REST. Referenční implementace je pro Java SE aplikace založené na Swingu. Framework nabízí kromě generování rozhraní také moduly umožňující připojení na server a získání definicí a dat. Níže se seznámíte se základními koncepty použití a funkčností, kterými Framework disponuje.

## Základní koncept

Základním konceptem je generování definicí dat na serverové straně. Lze takto mít pod kontrolou uživatelské rozhraní na klientovi. Definice také zohledňují aktuální datový či visuální model a dokáží pružně reagovat na změny datových typů či polí. Klient pouze specifikuje adresy zdrojů, na kterých jsou definice a adresy zdrojů, na kterých jsou data. Dále lze specifikovat adresy zdrojů pro aktualizace, mazání či vytváření dat. Klient tedy slouží pouze jako interpreter.

Serverová strana umožňuje na základě datových typů generovat různé typy widgetů. K dílčí generaci je využit Framework AspectFaces.

# Použití frameworku

## Závislosti

Framework je koncipován jako Maven projekt. Následující závislosti musí být přidány na straně serveru do projektu, či jejich JAR jako knihovny. Dále je potřeba stáhnout přednastavené konfigurační soubory ke generování dat a vložit je do WEB-INF. Tyto soubory lze později upravit. Stáhnout je lze zde.



Obrázek 1 Maven závislosti serveru

Na klientské straně je potřeba přidat následující závislosti

Obrázek 2 Maven závislosti klienta

## Serverová strana

Serverová strana vytváří definice dat a dodává data klientské aplikaci. Zároveň je potřeba aby serverová strana uměla i požadavek přijmout. K přijmutí požadavku lze využít REST, který lze velmi snadno implementovat za použití již stávajících knihoven. Zdroj, který generuje definice dat, pak vypadá například takto:



Obrázek 3 Rest definice zdroje na serveru

Třída Country, na základě které bude vytvořena definice je zachycena na obrázku 5. Pro jednoduchost nyní můžeme říci, že definice v sobě bude obsahovat pole pro id, jméno (name), zkratku (shortcut), a příznak aktivity (active). Důvodem tohoto zjednodušeného řešení je fakt, že v této příručce prozatím nebyly rozebrány možnosti generujícího frameworku, kterým je AspectFaces. Definice nespecifikuje, zdali se bude jednat o tabulku či o formulář celá tato záležitost je v plné kompetenci klienta.

Dalším důležitým zdrojem je zdroj s daty, či zdroj, který přijme data k vytvoření upravení či smazání. Příkladem je zdroj, který produkuje sadu dat do tabulky a je zachycen na Obrázek 4.



Obrázek 4 zdroj, který vrátí sadu dat

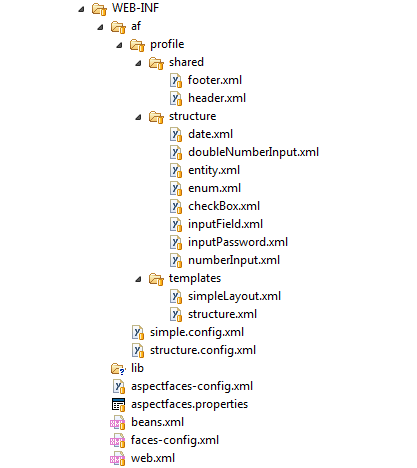


Obrázek 5 Objekt k inspekci na serveru

V případě generování formuláře musí zdroj vracet právě jeden objekt, na základě kterého bude formulář sestaven. Příklad takového zdroje lze najít na Obrázek 6. Formulář může být naplněn daty pouze z konkrétní instance třídy. 

Obrázek 6 zdroj, který vrací instanci země

## Generování definic s využitím AspectFaces

Jak již bylo zmíněno, k hlavní inspekci dat je využit Framework AspectFaces. Při prvním seznámení s projektem bylo potřeba zkopírovat do WEB-INF obsah vzorového nastavení pro generování. Struktura je zobrazená na obrázku Obrázek 7. Důležitým souborem je structure.config.xml. Tento soubor specifikuje mapování datových typů na komponenty. V případě, že datový typ v této specifikaci chybí tak mu nebude přiřazena žádná komponenta a tento prvek se tedy ve formuláři neobjeví. Jednotlivé komponenty jsou ve složce structure. Důvodem je pouze specifikace v structure.config.xml, která na tuto složku odkazuje. Komponenty tak mohou být umístěny v libovolné složce. Další důležitou součástí jsou šablony, které jsou umístěny ve složce templates. Tyto šablony specifikují, jak se budou komponenty vytvářet, dále lze mezi ně při vytváření vkládat tagy. Pro účely AFSwinx postačí, že kořenová třída by měla používat structure.xml a odvozené třídy simpleLayout.xml. Detaili xml se můžete prohlídnout ve vzorovém projektu.

Obrázek 7 struktura projektu

## Klient

Klientem se v našem frameworku rozumí jakákoliv Swingová aplikace. Framework generuje panely, které lze přidat do dalších panelů. Tímto způsobem můžeme docílit toho, že pouze část aplikace bude využívat generované uživatelské rozhraní. Zbývající část aplikace může být napsána k libovůli uživatele.

### Základní použití

Základním použitím jsou formuláře nebo tabulky. Budeme vycházet ze zdrojů uvedených výše. Důležité jsou tyto faktory

* Endpoint pro definice dat
* Endpoint pro data
* Endpoint pro akce (vytvoření, aktualizace, mazání)
* Komponenta – zvolení zdali půjde o formulář či tabulku
* Skin – základní nebo rozšířený
* Lokalizace – Framework podporuje lokalizace dat i validačních zpráv
* Identifikátor formuláře.

Hlavní správu zajišťuje třída AFSwinx. Jedná se o singleton třídu pomocí, které by měli být vytvářeny komponenty. Skrze tuto třídu lze získat komponentu kdekoliv v rámci aplikace a nad komponentou provádět akce. Příklad použití AFSwinx a vytvoření formuláře je na Obrázek 7. Nejprve je potřeba získat instanci AFSwinx. Poté builder, který postaví tabulku. Tento builder lze vyměnit za formulářový builder. Nejprve builder inicializujeme a to tak, že nastavíme identifikátor tabulky, předáme mu soubor se specifikací zdrojů a identifikátor zdroje, který chceme použít. První dvě řádky načítají soubor connection.xml. Soubor může být načten mnoha způsoby toto je pouze jeden z nich.



Obrázek 8 vytvoření tabulky v AFSwinx

K vytvoření tabulky či formuláře je potřeba specifikovat zdroje. V předchozím příkladu jsme provedli specifikaci na základě xml. Na Obrázek 8 je definice zdrojů. V kořenovém elementu connectionRoot může být 0 až N elementů typu connection s identifikátorem, na základě kterého je rozhodnuto jaký zdroj se použije. Element metaModel v elementu connection je povinný neboť specifikuje zdroj s definicemi z Obrázek 3 . Element data specifikuje konkrétní data z Obrázek 4 a element post specifikuje zdroj, na který budou data odeslána. Kromě textového zápisu lze využít i expression language a to v její standardní formě například: #{id}. Zdroje zobrazené na obrázku 9 získávají data o metamodelu z nezabezpečeného zdroje na adrese <http://localhost:8080/AFServer/rest/country> ve formátu Json. Data pro metamodel jsou na adrese <http://localhost:8080/AFServer/rest/country/list> a data budou zpětně zaslána na adresu <http://localhost:8080/AFServer/rest/country> metodou post. Podporované metody jsou get, post, put a delete.



Obrázek 9 definice zdrojů

# Možnosti

## Mapování

Jak již bylo předesláno. Framework využívá k inspekci data framework AspectFaces. Součástí tohoto frameworku je mapování datových typů na komponenty. Komponenty jsou definovány pomocí xml stejně jako mapování datových typů na komponenty. Ukázka mapování je na obrázku 10. Objekt typu String se bude mapovat na komponentu, jejíž definice je popsána v souboru inputField.xml a do této komponenty budou propagovány proměnné minLength a maxLength. V případě že se jedná o heslo tak se bude objekt mapovat na komponentu inputPassword.xml. Objekty typu int a long se budou mapovat na komponentu numberInput.xml. Objekt typu Address se bude mapovat na entity.xml, neboť se jedná o složitý datový objekt. Cesty ke komponentám odpovídají z obrázku 7 struktura projektu.



Obrázek 10 Mapování datových typů na komponenty

Na základě mapování se použije definice komponenty. Komponenta strinputField.xml je zobrazena na obrázku 11 definice komponenty structure/inputField.xml. Komponenta musí začínat kořenovým elementem widget.



Obrázek 11 definice komponenty structure/inputField.xml

|  |  |
| --- | --- |
| Element | Popis |
| widgetType | Typ komponenty. Specifikuje, zdali se jedná o textové pole, číselné pole, drodown pole a podobně. Výčet možností bude specifikován v další kapitole |
| fieldName | Název pole. Například v objektu country z druhé kapitoly by se mohlo jednat například o name, shortCut a active. |
| Label | Popis, který bude zobrazen klientovi u políčka s touto hodnotou. Lze použít i lokalizační texty například country.name a framework si z resource bundle již hodnotu pro country.name dohledá sám. |
| Validations | Validace, které bude vstupní pole podporovat. Výčet validací bude rozebrán dále |
| fieldLayout | Popis layoutu. Layout vstupního pole na základě této definice budou prvky skládány v rámci této komponenty. |
| Readonly | Specifikuje, zdali je objekt pouze pro čtení. |
| Visible | Specifikuje, zdali je objekt viditelný. |

V případě, že se jedná o složitý datový typ je využitá komponenta entity.xml. Tuto komponentu nelze měnit a v případě změny, může být způsobena nefunkční inspekce složitých datových typů. Této komponentě se předá název třídy nad kterou bude provedena inspekce a název proměnné v aktuální entitě.



Obrázek 12 entity.xml – komponenta popisující neprimitivní datový typ

## Komponenty

Níže je seznam podporovaných komponent včetně jejich názvu a identifikace v XML.

|  |  |  |
| --- | --- | --- |
| Komponenty | Identifikátor | Popis |
| Vstupní textové pole | textField | Standardní vstupní pole pro text |
| Pole textu | Label | Standardní výstupní pole. Needitovatelné. Zobrazování jako text. |
| Vstupní číselné pole integer | numberField | Vstupní pole pro čísla typu int. |
| Vstupní číselné pole double | numberDoubleField | Vstupní pole pro čísla typu double. |
| Vstupní číselné pole long | numberLongField | Vstupní pole pro čísla typu long. |
| Výběrové menu | dropDownMenu | Výběrové menu. Lze vybrat jednu z několika možností. |
| Zaškrtávací pole | checkBox | Zaškrtávací políčko, Lze zaškrtnout jednu z možností. |
| Vstupní textová area | textArea | Vstupní textové pole pro velké texty. |
| Výběrové menu | option | Výběrové menu lze vybrat jednu z možností. |
| Datové pole s kalendářem | calendar | Vstupní pole, do kterého se vloží data pomocí date pickeru. |
| Vstupní pole pro heslo | Password | Vstupní pole pro heslo. |

### Validace

Komponenty mohou mít různé validace. Validace nespecifikují vizuální stav komponenty, ale ověřují její vnitřní stav, předtím než je komponenta využita ke zpětnému sestavení objektu. Rozeznáváme následující validace.

|  |  |  |
| --- | --- | --- |
| Název | Lokalizační klíč | Popis |
| MinValue | validation.number.toSmall | Validuje minimální číselnou hodnotu pole. |
| MaxValue | validation.number.toBig | Validuje maximální číselnou hodnotu pole. |
| MinLength | validation.length.toSmall | Validuje minimální počet znaků v poli. |
| MaxLength | validation.length.toBig | Validuje maximální počet znaků v poli. |
| Required | validation.required | Validuje, zdali je pole vyplněno. |
| Number | validation.number | Validuje, zdali je v poli číselná hodnota. Typ číselné hodnoty je závislé na číselné. |
| Retype | validation.retype | Validuje, zdali bylo pole správně opsáno. |

### Layouty

Uvnitř komponenty lze určovat jakým způsobem bude komponenta postavena a kde bude umístěn její popis (label). Layouty jsou určeny podle os. Rozeznáváme osu X a Y a jednosloupcový layout a dvousloupcový layout. Zároveň lze určit pozice labelu. Zdali bude jako první či jako poslední. Následující tabulka specifikuje prvky layoutu.

|  |  |
| --- | --- |
| Název | Popis |
| LabelPossition (before, after, none) | Pozice labelu. Před komponentou, po komponentě, label nebude uveden. |
| LayoutDefinition (TwoColumnsLayout, OneColumnLayout) | Definice layoutu dvousloupcový, jednosloupcový. |
| LayoutOrientation(AxisX, AxisY) | Orientace layoutu ( podle osy X, Y) |

V případě dvousloupcového layoutu podle osy Y budou komponenty vykreslovány nejprve dvě vedle sebe a další pod ně. V případě dvousloupcového layoutu podle osy X. Budou komponenty vykreslovány nejprve jako dvě pod sebe a vedle nich další. Počet sloupců určují počet komponent na jednom patře a orientace určuje orientaci všech pater.

# Závěr