Inżynieria oprogramowania, SUM Informatyka,
rok akademicki 2008/2009

PROJEKT DYDAKT
HARMONIX

Podręcznik użytkownika

AUTORZY
Mateusz Bilski (mateusz.bilski@webmasta.pl)
Michał Furman (michal.furman@webmasta.pl)

PROWADZĄCY:
mgr Witold Rakoczy

AGH 2008

Spis treści

1. Szybki start

1.1. Znajomość wybranych zagadnień Adobe Flex

Do zrozumienia sposobu wykorzystania framework'u HARMONIX wymagana jest znajomość następujących właściwości technologii Adobe Flex:

1. **Składnia języka MXML** (link: MXML Syntax)
2. **Obsługa Event'ów** (link: Using Events)
3. **Zastosowanie Data Provider'a** (link: Data Provider)
   - **Przykład: Użycie kolekcji jako data provider'a** (Using a collection)
4. **Zastosowanie Item Renderer'ów** (link: Using item renderers)
5. **Operacje drag & drop** (link: Using Drag and Drop)
6. **Bindowanie danych** (link: Binding Data)

1.2. Użycie biblioteki HARMONIX w Adobe Flex Builder

Użycie skompilowanej biblioteki HARMONIX realizuje się poprzez załączenie do bibliotek projektu pliku **harmonix.swc** poprzez wykonanie następujących kroków:

1. Klikamy prawym przyciskiem myszy na ikonie projektu i wybieramy ***Properties***
2. Z lewego menu wybieramy ***Flex Build Path***
3. Wybieramy zakładkę ***Library Path***
4. Klikamy ***Add SWC...*** i wskazujemy plik **harmonix.swc**
5. Potwierdzamy ***OK***

1.3. Użycie kodów źródłowych HARMONIX w Adobe Flex Builder

Aby użyć kodów źródłowych biblioteki HARMONIX należy wykonać następujące czynności:

1. Klikamy prawym przyciskiem myszy w zakładce ***Flex Navigator***
2. Wybieramy kolejno ***Import*** → ***General*** → ***Existing Projects into Workspace***
3. Klikamy ***Browse...*** i wskazujemy katalog, w którym znajduje się biblioteka **HARMONIX**
6. Klikamy prawym przyciskiem myszy na ikonie projektu, który ma korzystać z biblioteki i wybieramy ***Properties***

7. Z lewego menu wybieramy ***Flex Build Path***
8. Wybieramy zakładkę ***Library Path***
9. Klikamy ***Add Project...*** i wskazujemy zaimportowany wcześniej projekt **harmonix**
10. Potwierdzamy ***OK***
11. W menu ***Project*** zaznaczamy opcję ***Build Automatically***, aby biblioteka była kompilowana automatycznie po każdej operacji zapisania zmian

1.4. Kompilacja źródeł biblioteki HARMONIX przy pomocy Apache Ant

Do kompilacji źródeł biblioteki HARMONIX przy pomocy kompilatora **mxmlc** i narzędzi **ant** wymagane są następujące instalacje:

1. apache-ant-1.7.1
2. Antennae-1.2.2

Po zainstalowaniu powyższych narzędzi zgodnie z instrukcjami zawartymi na stronach projektów (Ant i Antennae), należy uruchomić z linii komend skrypt **build.bat** znajdujący się w głównym katalogu projektu HARMONIX.

2. Zastosowania

2.1. Rozmieszczenie dużej ilości danych wizualnych

Biblioteka HARMONIX dostarcza kilku gotowych layout'ów pozawalających na czytelne ułożenie dużej ilości danych tego samego typu. Funkcjonalność ta realizowana jest za pomocą komponentu ***LayoutContainer***, oraz następujących layoutów:

1. **HorizontalSeriesLayout** - Renderowanie danych w porządku horyzontalnym, od lewej do prawej, we wierszach.

2. **VerticalSeriesLayout** – Renderowanie danych w porządku wertykalnym, od góry do dołu, w kolumnach.

3. **CoverFlowLayout** – Renderowanie danych w formie dynamicznej horyzontalnej listy z jednym wyszczególnionym i powiększonym elementem.

Istnieje możliwość pokazywania danych czysto graficznych oraz danych należących do określonej kolekcji przekazanej do komponentu jako **dataProvider**. Możliwa jest dynamiczna zmiana layout'u która spowoduje przerenderowanie zawartości kontenera w sposób odpowiedni dla danego layout.

2.1.1. Graficzne komponenty

Komponent **LayoutContainer** jest odpowiedzialny za rozmieszczanie danych wizualnych (komponentów **UIComponent**) na podstawie dostarczonego opisu Layout'u (obiekty pochodne do **AbstractLayout**). Za rozmieszczenie dzieci jest odpowiedzialny w całości dostarczony Layout, który definiuje odpowiednie reguły. Komponent ten może być wykorzystany do :

1. Reprezentacja komponentów graficznych, których cykl życia nie jest zależny od danych

2. Kontener dużej ilości komponentów graficznych, których istnienie kontroluje się poprzez typowe dla flex'owych kontenerów operacje **addChild()** i **removeChild()**

2.1.2. Graficzne komponenty typu Data Renderer

Komponent **DataContainer** dziedziczy bezpośrednio z komponentu **LayoutContainer**, czyli oferuje wszystkie jego możliwości takie jak różne formy

layout'ów oraz ich dynamiczną zmianę. Przewagą komponentu DataContainer jest renderowanie danych biznesowych składowanych w kolekcji, za pomocą dostarczonego za pomocą własności **itemRenderer** komponentu odpowiedzialnego za wizualizacje złożonych lub generycznych obiektów kolekcji, w zależności od ich wartości. Dzięki temu jesteśmy w stanie oddzielić widok (prezentacje danych) od struktury w której są one przechowywane. Kontener DataContainer renderuje wszelkie zmiany dokonane na dostarczonej kolekcji oraz obiektach w niej przechowywanych bez konieczności ingerencji developera. Rozwiązanie to jest analogiczne do wykorzystanego w komponentach Flex'owych takich jak **DataGrid** czy **List**. Komponent ten może być wykorzystany do :

1. Reprezentacja graficzna danych przechowywanych w modelu, których zmiany muszą być widoczne dla użytkownika

2. Pokazywanie danych dynamicznie ładowanych z zewnętrznych źródeł (**HTTPService**, **RemoteObject**, **WebService**)

2.1.3. Dynamiczna zmiana layout'u

Komponent typu **LayoutContainer** i jego pochodna **DataContainer** oferują możliwość dynamicznej zmiany reguł wyświetlania komponentów „dzieci". Za zmianę reguł pozycjonowania dzieci odpowiedzialna jest własność **layoutClass**, która przyjmuje obiekty pochodne do **AbstractLayout**. Możliwe jest zbindowanie wartości własności layoutClass do komponentu ComboBox lub innego dającego możliwość zaznaczenia wyboru. Komponent ten może być wykorzystany do :

1. Zaproponowanie użytkownikowi zmiany sposobu wyświetlania informacji w zależności od jego potrzeb i formy pokazywanych danych

2. Dynamiczna zmiana sposobu wyświetlania danych w zależności od ich ilości i zawartości (bez interakcji użytkownika)

2.2. Operacje schowka (clipboard)

Biblioteka HARMONIX oferuje ujednoliconą obsługę operacji na schowku, czyli kopiowanie, wycinanie oraz wklejanie elementów. Komponenty LayoutContainer oraz DataContainer mają wbudowaną obsługę schowka, która można włączyć za pomocą właściwości **clipboardEnabled**. Operacje schowka w komponencie LayoutContainer są wykonanywane na graficznych dzieciach (w schowku przechowywane są referencje do obiektów), natomiast w przypadku komponentu DataContainer w schowku przechowywane są dane powiązane z kopiowanym/wycinanym itemRenderer'em (czyli wizualną reprezentacją danych).

Możliwe jest wykorzystanie schowka w innych komponentach poprzez kompozycje obiektu **ClipboardClient** oraz implementację interfejsu **IclipboardClient**.

Komponenty **ClipboardClient** oraz **ClipboardManager** mogą być zastosowane do :

1. Ujednolicenie operacji schowka w całej aplikacji

2. Operacje na graficznych reprezentacjach danych za pomocą szybkich skrótów klawiszowych

3. Modyfikowanie kolekcji danych przez użytkownika w naturalny sposób poprzez kopiowanie/wycinanie pomiędzy graficznymi kontenerami.

4. Możliwość wizualizacji zawartości schowka przy pomocy kolekcji **clipboard** w klasie **ClipboardManager** (singleton)

2.2.1. Zaznaczanie elementów w kontenerach

Aby użytkownik miał możliwość skorzystania z operacji schowka musi mieć możliwość zaznaczania elementów w kontenerze. Komponenty **LayoutContainer** oraz **DataContainer** oferują dwa sposoby oznaczania, które realizowane są poprzez ustawienie odpowiednich właściwości:

1. **selectable** – użytkownik ma możliwość zaznaczenia jednego elementu w danym momencie, kliknięcie poza zaznaczony element powoduje jego odznaczenie

2. **allowMultipleSelection** – użytkownik ma możliwość jednoczesnego zaznaczenia kilku elementów poprzez przytrzymanie klawisza **crtl**.

   - kliknięcie poza zaznaczone elementy powoduje odznaczenie wszystkich zaznaczonych elementów

   - kliknięcie na dany element bez przytrzymania przycisku **crtl** powoduje zaznaczenie danego elementów oraz odznaczenie wszystkich pozostałych

2.2.2. Kopiowanie danych

Użytkownik ma możliwość kopiowania danych za pomocą skrótu klawiszowego **crtl + C**. Kopiowane są odpowiednio dla kontenera dane wizualne lub dane z kolekcji (data providera). Operacja to powoduje wyczyszczenie aktualnej zawartości schowka oraz umieszczenie w schowku skopiowanych elementów.

2.2.3. Wycinanie danych

Użytkownik ma możliwość wycinania danych za pomocą skrótu klawiszowego **crtl + X**. Kopiowane są odpowiednio dla kontenera dane wizualne lub dane z kolekcji (data providera). Operacja to powoduje usunięcie wyciętych danych z widoku oraz ew. kolekcji, wyczyszczenie aktualnej zawartości schowka oraz umieszczenie w schowku skopiowanych elementów.

2.2.4. Wklejanie danych

Użytkownik ma możliwość wklejenia zawartości danych za pomocą skrótu klawiszowego **crtl + Z**. Dane mogą być wklejone tylko do kontenerów danych oraz elementy graficzne mogą być wklejone tylko do kontenerów graficznych. Operacja ta nie powoduje wyczyszczenia schowka, dlatego też należy pamiętać o tym, że użytkownik może próbować wkleić do danego kontenera kilkukrotnie te same dane, co może mieć niebagatelne znaczenie w przypadku kontenera danych (zwielokrotnione referencje do tych samych danych w jednej kolekcji).

2.3. Operacje Drag & Drop (przeciągnij i upuść)

Biblioteka HARMONIX zawiera uproszczoną i ujednoliconą obsługę przenoszenia elementów pomiędzy kontenerami za pomocą operacji Drag & Drop. Podobnie jak w przypadku schowka, komponenty **LayoutContainer** oraz **DataContainer** mają wbudowana obsługę operacji Drag & Drop, którą można włączyć za pomocą właściwości **dragEnabled** (możliwość przenoszenia elementów) oraz **dropEnabled** (możliwość upuszczania elementów). Drag & Drop w komponencie LayoutContainer są wykonanywane na graficznych dzieciach, natomiast w przypadku komponentu DataContainer są to dane powiązane z przenoszonymi elementami itemRenderer (czyli wizualną reprezentacją danych).

HARMONIX obsługuje dwie możliwe operacje Drag & Drop:

1. Wycinanie i wklejanie – domyślne zachowanie. Dane lub elementy graficzne są wycinane z jednego kontenera i wklejane do drugiego, podobnie jak ma to miejsce przy operacjach schowka.

2. Kopiowanie przenoszonych danych – realizowane poprzez naciśnięcie podczas operacji Drag & Drop klawisza **ctrl**. Dane są kopiowane z jednego kontenera do drugiego (docelowego).

2.3.1. Przeciąganie komponentów graficznych

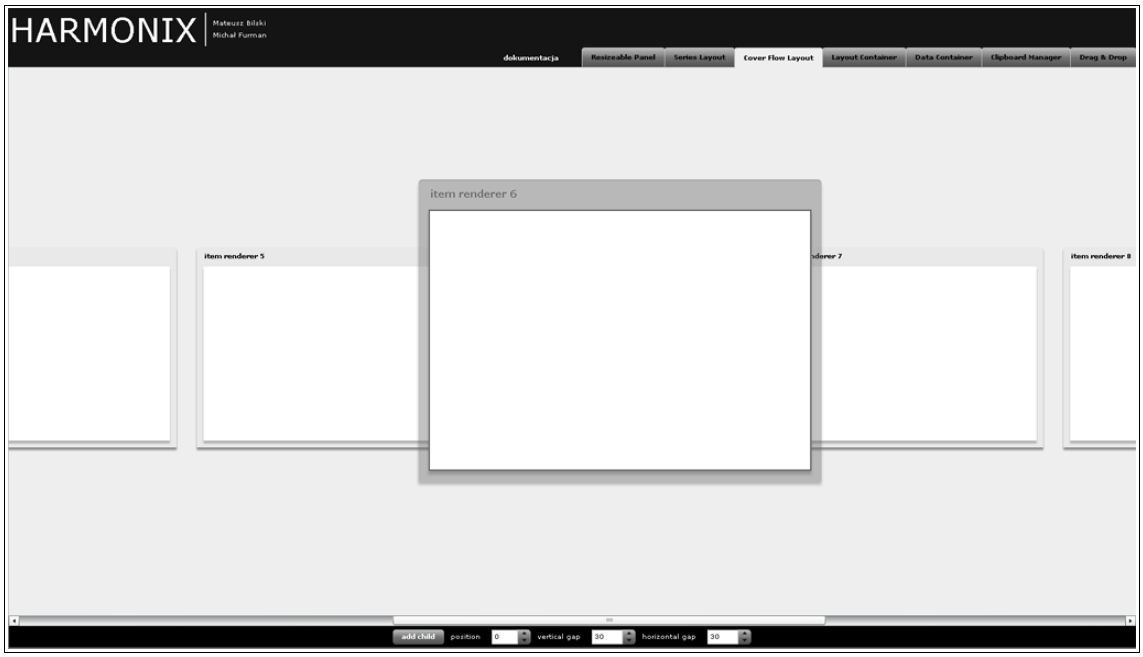
Przeciągane elementy graficzne są odpowiednio kopiowane i przenoszone do docelowego kontenera. Obiekty graficzne mogą być przeciągane tylko pomiędzy kontenerami graficznymi.

2.3.2. Przeciąganie danych

Przeciągane dane są odpowiednio kopiowane i przenoszone z kolekcji (dataProvider'a źródłowego komponentu) do docelowego kontenera. Obiekty reprezentujące dane mogą być przeciągane tylko pomiędzy kontenerami danych (DataContainer i pochodne).

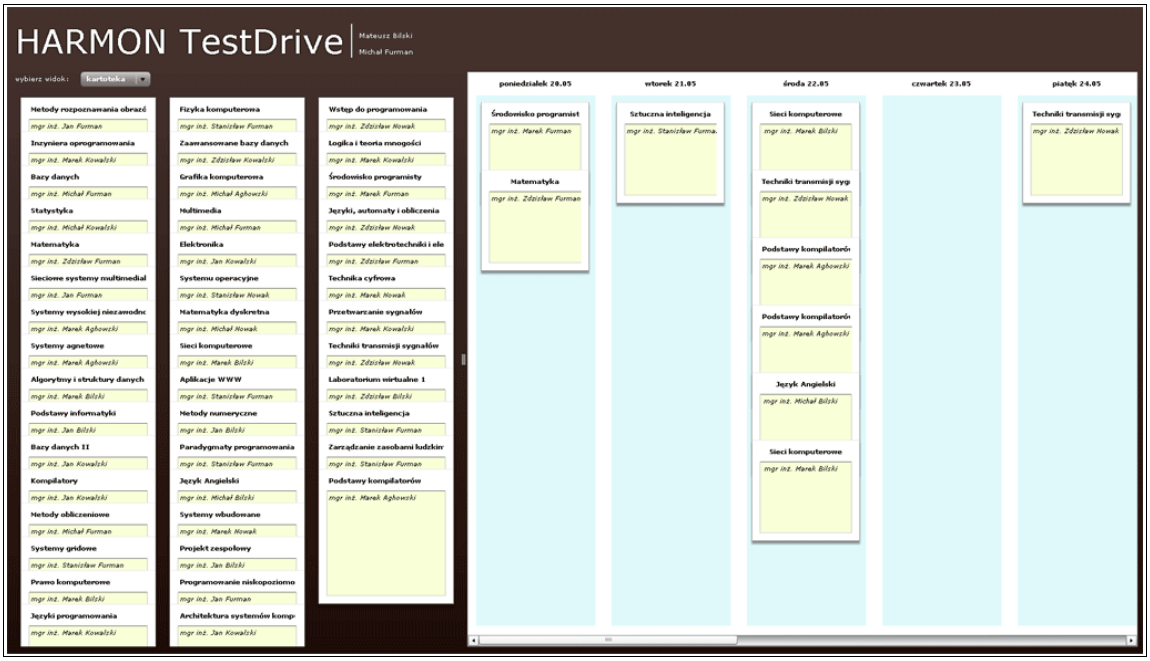
3. Przykłady

3.1. Aplikacja HarmonixOverview



Aplikacja prezentująca możliwości biblioteki Harmonix. Dostępna jest gotowa aplikacja oraz kod źródłowy.

3.2. Aplikacja HarmonTestDrive



Aplikacja prezentująca przykładowe zastosowanie biblioteki HARMONIX do projektu opierającego się na planowaniu harmonogramu. Dostępna jest gotowa aplikacja oraz kod źródłowy.

3.2.1. Funkcjonalności

W aplikacji **HARMON TestDrive** zaimplementowano następujące funkcjonalności:

1. przeglądanie dostępnych kursów na dwa różne sposoby
   - kartoteka o układzie wertykalnym
   - układ cover flow
2. uproszczony widok dni tygodnia
3. przeciąganie z kartoteki na wybrany dzień kursów z wykorzystaniem mechanizmów Drag & Drop
4. kopiowanie i wklejanie wybranych kursów z kartoteki do wybranych dni

3.2.2. Sposób implementacji

Powyższe funkcjonalności zostały zaimplementowane w następujący sposób:

1. Przeglądanie dostępnych kursów:
   - *ArrayCollection* jako kolekcja danych typu *Curse*
   - *DataContainer* jako kontener dla danych z powyższego providera, oraz komponent *CurseRenderer* jako itemRenderer, kontener z ustawieniami:
     - *layoutClass* – zbindowany do *ComboBox'a* z 2 klasami layoutów
     - *selectable* = *true* - daje możliwość zaznaczania elementów
     - *selectionColor* – kolor oznaczenia elementu
     - *clipboardEnable* = *true* – funkcjonalność schowka (kopiuj, wklej)
     - *dropEnabled* = *false* - zablokowana możliwość upuszczania elementów
2. Uproszczony widok dni tygodnia
   - *ArrayCollection* jako kolekcja danych typu Day
   - *TileList* jako lista dni zbindowana do powyższej kolekcji oraz komponentem *DayRenderer*, jako itemRenderer
   - *DayRanderer* zawiera *DataContainer*, który renderuje dane z listy kursów każdego z dni (lista kursów w postaci *ArrayCollection*), kontener z ustawieniami:
     - *layoutClass* - jako *VerticalSeriesLayout*
     - *clipboardEnabled* = *true* – funkcjonalność schowka (kopiuj, wklej)
     - *selectable* = *true* – daje możliwość zaznaczania elementów
     - *allowMultipleSelection* = *true* – możliwość zaznaczania wielu elementów jednocześnie
     - *selectionColor* – ustawienie koloru zaznaczenia
     - *deleteEnable* – możliwość usuwania elementów za pomocą klawisza „delete”
3. Przeciąganie z kartoteki na wybrany dzień kursów z wykorzystaniem mechanizmów Drag & Drop

- *dragEnable* i *dropEnable* są ustawione domyślnie na wartość *true* dla kontenerów, dlatego też, dla widoku kursów upuszczanie zostało wyłączone (false), aby użytkownik nie mógł wklejać do kursów tych samych obiektów

4. Kopiowanie i wklejanie wybranych kursów z kartoteki do wybranych dni
   - Operacje schowka zostały zrealizowane poprzez ustawienie parametrów *selectable* i *allowMultupleSelection* na true, ponieważ domyślnie są one wyłączone.

4. Załączniki

4.1. Biblioteka HARMONIX

Skompilowana biblioteka **SWC**, wygenerowana dokumentacja, kod źródłowy.

4.2. Aplikacja HarmonixOverview

Aplikacja WWW oraz kod źródłowy.

4.3. Aplikacja HarmonTestDrive

Aplikacja WWW oraz kod źródłowy.