Dostosowywanie wyglądu aplikacji ADF Faces

Marek Wojciechowski
Politechnika Poznańska
e–mail: Marek.Wojciechowski@cs.put.poznan.pl

Abstrakt. ADF Faces to opracowana przez firmę Oracle w ramach frameworka Oracle Application Development Framework (ADF) biblioteka komponentów graficznych do budowy interfejsu użytkownika w aplikacjach opartych o technologię JavaServer Faces (JSF). Charakterystyczne dla JSF, a w związku z tym również dla ADF Faces, komponentowe podejście do budowy stron WWW stanowiących interfejs aplikacji odcina programistę od wynikowego kodu HTML, a przez to ogranicza możliwości wpływania na wygląd aplikacji poprzez arkusze stylów CSS. Aby zaoferować twórcom aplikacji większą niż przewidziana w ramach technologii JSF kontrolę nad szatą graficzną aplikacji, ADF Faces udostępnia mechanizm tzw. skórek, umożliwiających precyzyjne określenie formatowania poszczególnych komponentów strony. Celem artykułu jest przedstawienie zasady działania skórek, sposobu ich tworzenia i wykorzystywania w aplikacjach, a także wsparcia narzędzi Oracle dla tworzenia i edycji skórek.

1. Wprowadzenie

Oracle Application Development Framework (ADF) to framework do tworzenia złożonych aplikacji w języku Java, stanowiący kluczowy element platformy Oracle Fusion Middleware. Oferuje on wiele innowacyjnych rozwiązań takich jak deklaratywne wiązanie warstwy usług biznesowych z warstwą prezentacji abstrahujące od technologii wykorzystanych do implementacji tych warstw aplikacji (ADF Model) czy też modularne podejście do sterowania przepływem w aplikacji webowej (ADF Task Flows). Jednakże można z całą pewnością zaryzykować stwierdzenie, że wizytówką frameworka ADF i jego najważniejszą technologią składową jest obecnie ADF Faces.

ADF Faces to biblioteka komponentów graficznych do budowy interfejsu użytkownika w aplikacjach opartych o technologię JavaServer Faces (JSF). Bogactwo i interaktywność komponentów ADF Faces, uzyskana w dużym stopniu dzięki technice Ajax, spowodowały marginalizację technologii ADF Swing, umożliwiającej tworzenie w ADF aplikacji desktopowych, wcześniej stosowanych m.in. gdy wymagana była większa responsywność interfejsu użytkownika niż charakterystyczna dla pierwszych aplikacji webowych. Aktualna wersja biblioteki ADF Faces (11g) ze względu na predyspozycje do budowy tzw. bogatych aplikacji internetowych (ang. Rich Internet Applications – RIA) określana jest również mianem ADF Faces Rich Client.

Biblioteka ADF Faces nie tylko jest obecnie podstawową technologią tworzenia warstwy prezentacji aplikacji opartych o stos technologii frameworka ADF, ale może być użyta również w standardowych aplikacjach Java EE wykorzystujących JSF. Aby rozpropagować swoją bibliotekę komponentów wśród programistów JSF, Oracle przekazał poprzednią wersję ADF Faces (10g) Apache Software Foundation, gdzie jest ona niezależnie rozwijana jako Apache MyFaces Trinidad. Co ciekawe, niektóre mechanizmy, które pojawiły się w Apache MyFaces Trinidad, zostały później wykorzystane przez Oracle w ADF Faces Rich Client.

Ważną zaletą ADF Faces jest możliwość precyzyjnego dostosowywania wyglądu poszczególnych komponentów jak i całych stron aplikacji poprzez specyfikację takich aspektów wizualnych jak czcionka, kolorystyka, obramowania, ikony, obrazki stanowiące tło, itd. Ponieważ wynikowa postać stron ADF Faces trafiająca do przeglądarek użytkowników to HTML, formatowanie treści oparte jest oczywiście o arkusze stylów CSS. Jednakże podobnie jak w przypadku standardowych stron JSF, twórcy stron ADF Faces są w pewnym sensie ograniczeni przez komponentowe podejście do interfejsu użytkownika, odnosząc reguły formatowania do złożonych komponentów, a nie ich elementów składowych. Odpowiedzią na problemy z

bezpośrednim stosowaniem CSS w aplikacjach ADF Faces jest dostępny od wersji 11g biblioteki mechanizm skórek (ang. skinning). Skórka (ang. skin) to specjalny, oparty o składnię CSS arkusz stylów, którego selektory odnoszą się do komponentów ADF Faces. Mechanizm skórek ułatwia zapewnienie spójności wyglądu aplikacji, jednocześnie pozwalając na precyzyjne formatowanie złożonych komponentów. Ponadto, skórki mogą być zmieniane w trakcie pracy aplikacji przez użytkowników końcowych. Ze względu na wspomniane wyżej zalety, mechanizm skórek jest obecnie zdecydowanie preferowanym sposobem formatowania stron ADF Faces. O słuszności przyjętego rozwiązania świadczy też fakt, że konkurencyjne dla ADF Faces biblioteki komponentów oparte o JSF (RichFaces, ICEFaces) również oparły się w zakresie formatowania o szeroko pojętą koncepcję skórek.

Celem niniejszego artykułu jest kompleksowe przedstawienie zagadnienia dostosowywania wyglądu aplikacji ADF Faces, począwszy od bezpośredniego stosowania arkuszy stylów CSS, a skończywszy na skórkach, na które oczywiście położony będzie główny nacisk. Opisane będą również narzędzia ułatwiające implementację i testowanie skórek. Proces tworzenia, aplikacji i testowania skórki przedstawiony będzie na przykładzie modyfikacji jednej ze skórek dostarczonych wraz ze środowiskiem Oracle JDeveloper.

2. Zmiana wyglądu stron ADF Faces poprzez arkusze stylów CSS

2.1. CSS w JavaServer Faces i ADF Faces

Formatowanie stron ADF Faces poprzez arkusze stylów CSS nie odbiega znacząco od ogólnego podejścia stosowanego w JavaServer Faces. Standardowe komponenty JSF generalnie posiadają dwa atrybuty przeznaczone do aplikowania do nich reguł formatujących CSS: style i styleClass[1]. Pierwszy z nich umożliwia specyfikację za pomocą składni CSS formatowania wewnątrz znacznika komponentu, drugi umożliwia przypisanie komponentowi klasy (lub klas), do której (których) odnoszą się selektory reguł zawartych w zewnętrznym arkuszu stylów CSS. W ADF Faces atrybut style został przemianowany na inlineStyle, co lepiej oddaje jego przeznaczenie. Dodatkowo, komponenty ADF Faces mogą posiadać jeszcze atrybuty contentStyle i labelStyle, które umożliwiają specyfikację formatowania dla odpowiednio: zawartości i etykiety komponentu, a nie komponentu jako całości.

2.2. Domyślne formatowanie stron ADF Faces

Do zilustrowania sposobu określania wyglądu stron ADF Faces, początkowo w oparciu o standard CSS, a w dalszej części artykułu poprzez mechanizm skórek, wykorzystamy prosty formularz do wyszukiwania według podanego słowa lub frazy. Formularz będzie zawierał dwie kontrolki z biblioteki ADF Faces: pole tekstowe z etykietą (kontrolka Input Text z zestawu kontrolek tekstowych i kontrolek wyboru – Text and Selection) oraz przycisk (kontrolka Button z zestawu kontrolek ogólnych – General Controls). W przykładzie pominiemy obsługę logiki wyszukiwania jako nieistotną z punktu widzenia zagadnień wizualnej prezentacji strony. Kod źródłowy strony oraz jej wygląd w przeglądarce przy domyślnym formatowaniu zostały przedstawione poniżej.

```
<?xml version='1.0' encoding='UTF-8'?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
```

[1] Niektóre komponenty posiadają dodatkowe atrybuty do współpracy z CSS, umożliwiające przypisanie klasy do regionu bądź stanu komponentu. Przykładem jest komponent h:dataTable, który posiada dodatkowe atrybuty umożliwiające przypisanie klas do nagłówka tabeli, jej stopki oraz kolejnych wierszy i kolumn.

```
<f:view xmlns:f="http://java.sun.com/jsf/core"
xmlns:af="http://xmlns.oracle.com/adf/faces/rich">
    <af:document title="Wyszukiwarka" id="d1">
        <af:form id="f1">
            <af:inputText label="Słowo lub fraza" id="it1" />
            <af:commandButton text="Szukaj" id="cb1"/>
        </af:form>
    </af:document>
</f:view>
```

Rys. 1. Przykładowy formularz (formatowanie domyślne)

Mimo że nie dokonaliśmy żadnych ustawień dotyczących formatowania (ani w kodzie samej strony ani na szczeblu projektu), łatwo zauważyć, że wygląd strony w przeglądarce odbiega od domyślnego sposobu prezentacji zawartości HTML. Po podejrzeniu źródła strony wysłanej do przeglądarki w sekcji <head> można będzie zauważyć treść podobną do przedstawionej poniżej:

```
...
<head><link rel="stylesheet" charset="UTF-8" type="text/css" href="/Ploug11-
ViewController-context-root/adf/styles/cache/fusionFx-v2-desktop-ycin8s-ltr-
safari-cmp.css"><title>Wyszukiwarka</title></head>
...
```

Jak widać, za zaobserwowane formatowanie strony odpowiada arkusz stylów CSS, który został automatycznie dołączony przez framework ADF. Arkusz ten jest efektem zastosowania skórki, w tym wypadku fusionFx w wersji v2, co można odczytać z nazwy pliku. Mechanizmem skórek i jego związkami z CSS zajmiemy się w dalszej części artykułu, na razie podkreślmy tylko, że jest on domyślnie włączony i przy braku jawnej konfiguracji ustawień związanych z formatowaniem stron, użyta będzie skórka domyślna – tak jak było w naszym przypadku.

2.3. Formatowanie stron ADF Faces zewnętrznym arkuszem stylów

Załóżmy, że chcemy zmodyfikować wygląd formularza tak aby etykiety pola tekstowego i przycisku były wyświetlone czcionką times, a tło pola tekstowego było jasnoszare (kolor #DDDDDD). Zgodnie z duchem CSS nasze pierwsze podejście do realizacji tego zadania będzie oparte o zewnętrzny arkusz stylów definiujący dwie klasy: klasyczny – odpowiadającą za określenie czcionki i szary – specyfikującą kolor tła:

```
.szary
{
  background-color: #DDDDDD;
}
.klasyczny
{
  font-family: times;
}
```

Zdefiniowane klasy należy teraz przyporządkować kontrolkom ADF Faces na stronie:

```
...
  <af:inputText label="Słowo lub fraza" id="it1" styleClass="klasyczny szary"/>
  <af:commandButton text="Szukaj" id="cb1" styleClass="klasyczny"/>
...
```

Preferowanym sposobem dołączenia zewnętrznego arkusza stylów w CSS w ADF Faces jest użycie znacznika <af:resource>, choć dostępny jest również standardowy znacznik JSF 2.0 <h:outputStylesheet>. Nie należy używać do tego celu znacznika HTML <link>, gdyż element ten musi znajdować się w sekcji <head> dokumentu HTML, a ADF Faces nie udostępnia bezpośrednio sekcji <head> i <body> zastępując je pewnego rodzaju abstrakcją dokumentu w formie elementu <af:document>. Poniższy fragment kodu strony ADF Faces ilustruje sposób podpięcia arkusza stylów znacznikiem <af:resource>:

```
<?xml version='1.0' encoding='UTF-8'?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<f:view xmlns:h="http://java.sun.com/jsf/html"
xmlns:f="http://java.sun.com/jsf/core"
xmlns:af="http://xmlns.oracle.com/adf/faces/rich">
    <af:document title="emps.jsf" id="d1">
        <af:resource type="css" source="resources/css/ploug.css"/>
...
```

Efekt dołączenia naszego arkusza stylów pokazuje Rys. 2. Zauważmy, że po pierwsze – tam gdzie nie podaliśmy własnych zasad formatowania, w dalszym ciągu obowiązują ustawienia domyślnej skórki (np. kolorystyka przycisku), a po drugie – efekty działania naszego arkusza stylów nie w pełni są zgodne z oczekiwaniami: o ile krój czcionki zmienił się dla przycisku, to pozostał niezmieniony dla etykiety pola tekstowego, a do tego szare tło zostało zaaplikowane nie do obszaru wprowadzania tekstu pola tekstowego, a do jego etykiety. Przyjrzyjmy się kolejno obu powyższym kwestiom.

Rys. 2. Przykładowy formularz (formatowanie zewnętrznym arkuszem CSS)

Po podejrzeniu źródła strony wysłanej do przeglądarki w sekcji nagłówka dokumentu zauważymy treść podobną do przedstawionej poniżej:

```
...
<head><link rel="stylesheet" charset="UTF-8" type="text/css" href="/Ploug11-
ViewController-context-root/adf/styles/cache/fusionFx-v2-desktop-ycin8s-ltr-
safari-cmp.css"><title>Wyszukiwarka</title><link rel="stylesheet"
type="text/css" afrres="true" href="resources/css/ploug.css"></head>
...
```

Użycie znacznika <af:resource> skutkuje oczywiście wygenerowaniem elementu <link> dołączającego wskazany arkusz stylów. Należy jednak zwrócić uwagę, że oprócz naszego arkusza stylów w dalszym ciągu dołączany jest arkusz CSS wynikający z domyślnej skórki. Dzieje się tak dlatego, że mechanizm skórek jest nie tylko preferowanym i domyślnym mechanizmem formatowania stron ADF Faces, ale również obowiązkowym. Twórca aplikacji może wybrać jedną ze standardowo dostępnych skórek, może też stworzyć własną, a gdy nie dokona w tym zakresie żadnej konfiguracji, użyta będzie skórka domyślna. Stosując własne arkusze stylów CSS można więc skorygować formatowanie stron ADF Faces wynikające z obowiązującej dla projektu skórki, ale nie można oprzeć się tylko i wyłącznie na nich.

Wyjaśnienia problemu niezgodnego z oczekiwaniami formatowania pola tekstowego oczywiście też powinniśmy szukać w źródle HTML wysłanym do przeglądarki. O ile kontrolka przycisku jest renderowana jako element <button> HTML-a, to kontrolka pola tekstowego, jako przykład komponentu złożonego, renderowana jest jako zestaw różnych, odpowiednio zagnieżdżonych

elementów HTML, w tym wypadku – w formie tabelki HTML zawierającej jeden wiersz z dwoma komórkami: jedną dla etykiety, a drugą dla zasadniczego pola tekstowego:

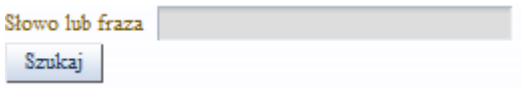
```
<table class="klasyczny szary x1u" id="it1" cellpadding="0" cellspacing="0"
border="0" summary=""><tbody><tr><td class="xu"><label
for="it1::content">S&#322;owo lub fraza</label></td><td valign="top" nowrap
class="xpn"><input id="it1::content" name="it1" class="x25"
type="text"></td></tr></tbody></table>
```

Łatwo zauważyć, że zdefiniowane przez nas klasy zostały zaaplikowane do najbardziej zewnętrznego elementu HTML fragment dokumentu reprezentującego kontrolkę ADF Faces. Podejście to nie stanowi problemu dla prostych kontrolek takich jak przycisk, skutkujących pojedynczym elementem HTML, ale może prowadzić do niezgodnego z intencjami twórcy strony sposobu prezentacji komponentu dla kontrolek złożonych. W naszym przypadku wyspecyfikowany kolor tła został zaaplikowany do całej tabelki reprezentującej pole tekstowe z etykietą, co spowodowało zmianę tła etykiety. Co więcej, samo pole tekstowe do wprowadzenia danych nie otrzymało tła określonego w naszym arkuszu stylów, gdyż jego formatowanie zostało określone klasą CSS wynikającą ze skórki, nadpisując ustawienia z komponentu otaczającego.

2.4. Formatowanie stron ADF Faces poprzez zagnieżdżone style CSS

Problem formatowania komponentów złożonych można oczywiście próbować rozwiązać odnosząc reguły CSS do wynikowego HTML-a. Takie rozwiązanie byłoby jednak zależne od sposobu renderowania komponentów ADF Faces i w przypadku jego zmiany w przyszłych wersjach frameworka mogłoby wymagać modyfikacji. Lepszym, choć nie idealnym, rozwiązaniem jest użycie dodatkowych atrybutów do współpracy z CSS: labelStyle i contentStyle, które posiadają złożone komponenty ADF Faces. Za pomocą pierwszego można przypisać styl CSS do etykiety, a za pomocą drugiej – do zawartości. Kontrolka pola tekstowego ma oba te atrybuty. Poniższy fragment kodu pokazuje sposób ich wykorzystania do uzyskania zamierzonego przez nas formatowania:

```
...
  <af:inputText label="Słowo lub fraza" id="it1"
                contentStyle="background-color: #DDDDDD;"
                labelStyle="font-family: times;"/>
...
```



Rys. 3. Przykładowy formularz (formatowanie stylami CSS w atrybutach labelStyle i contentStyle)

Skutek zastosowania zagnieżdżonych stylów pokazuje Rys. 3. Tym razem uzyskany efekt jest zgodny z zamierzeniami. Mankamentem rozwiązania opartego o style CSS zagnieżdżone w atrybutach labelStyle i contentStyle jest oczywiście to, że trudno w oparciu o nie zrealizować efekt spójnego wyglądu komponentów danego rodzaju i generalnie stron aplikacji. Jeśli jednak przyjąć, że bezpośrednie stosowanie CSS w ADF Faces i tak ma charakter uzupełniania i modyfikacji formatowania wynikającego ze skórki, jest to technika akceptowalna pod warunkiem, że nie będzie nadużywana, a ustawienia wyglądu odnoszące się do więcej niż jednego komponentu będą realizowane poprzez dostosowanie skórki.

Kończąc dyskusję na temat bezpośredniego stosowania CSS względem komponentów ADF Faces należy jeszcze wspomnieć, że style CSS zawarte w atrybutach labelStyle i contentStyle, a także inlineStyle, mogą zawierać konstrukcje języka wyrażeń EL, dzięki czemu można zrealizować np. warunkowe formatowanie komponentów.

3. Mechanizm skórek w ADF Faces

3.1. Geneza i zasada działania

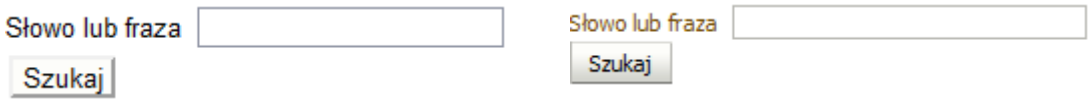
Koncepcja skórek (ang. skin) wykracza poza aplikacje internetowe. Wiele popularnych aplikacji desktopowych (np. odtwarzacz muzyki Winamp) umożliwiało i umożliwia dostosowywanie ich wyglądu przez użytkowników końcowych poprzez wybór jednej z oferowanych skórek. Również w obszarze aplikacji internetowych, a nawet konkretnie bibliotek komponentów JSF, ADF Faces nie jest jedyną biblioteką stosującą skórki, gdyż stosuje je również, choć oczywiście w innej formie, np. biblioteka RichFaces. Główną motywacją dla skórek w ADF Faces, było nie tyle umożliwienie użytkownikom zmieniania wyglądu aplikacji, co zapewnienie twórcom aplikacji mechanizmu specyfikacji wyglądu stron, który będzie operował na poziomie komponentów, a jednocześnie umożliwiał precyzyjne formatowanie. Niemniej, można w aplikacji ADF Faces oprogramować również dynamiczny wybór skórki w trakcie pracy aplikacji.

Korzenie mechanizmu skórek dostępnego w ADF Faces sięgają jeszcze dawnej, firmowej technologii Oracle o nazwie Oracle User Interface XML (UIX). UIX wykorzystywał do specyfikacji stylu prezentacji język XML Style Sheet (XSS). Po pojawieniu się technologii JSF, Oracle „przepisał" komponenty UIX na komponenty zgodne z JSF i tak narodziła się biblioteka ADF Faces przejmując po UIX XSS jako język opisu stylów. Gdy Oracle podarował bibliotekę ADF Faces w wersji 10g fundacji Apache, biblioteka ta rozwinęła się pod nazwą Apache MyFaces Trinidad. W szczególności zmodyfikowany i rozszerzony został mechanizm skórek. Biblioteka ADF Faces Rich Client została w znacznym stopniu oparta na projekcie Trinidad, przejmując m.in. w całości mechanizm skórek, czego przejawem są choćby nazwy plików konfiguracyjnych z nim związanych.

Jak już wspomniano wcześniej, mechanizm skórek silnie bazuje na CSS i wynikową postacią arkusza stylów wysyłana do przeglądarki jest standardowy arkusz CSS 2.x, co zapewnia zgodność z większością popularnych przeglądarek. Źródłowy dokument skórki jest również w formacie CSS (składnia CSS 3), ale jego reguły nie odnoszą się do elementów HTML, ale do komponentów ADF Faces. Skórka definiowana jest na poziomie projektu, a nie poszczególnych stron, co ułatwia zapewnienie spójnego wyglądu aplikacji. W trakcie pracy aplikacji w oparciu o arkusz skórki generowany jest standardowy arkusz stylów CSS i właśnie on trafia do przeglądarki. Składnia reguł skórki umożliwia różnicowanie formatowania zależnie od rodzaju przeglądarki użytkownika, co pozwala zabezpieczyć aplikację przed w dalszym ciągu występującymi różnicami w interpretacji stylów CSS 2.1 między przeglądarkami.

3.2. Skórki dostępne w środowisku JDeveloper

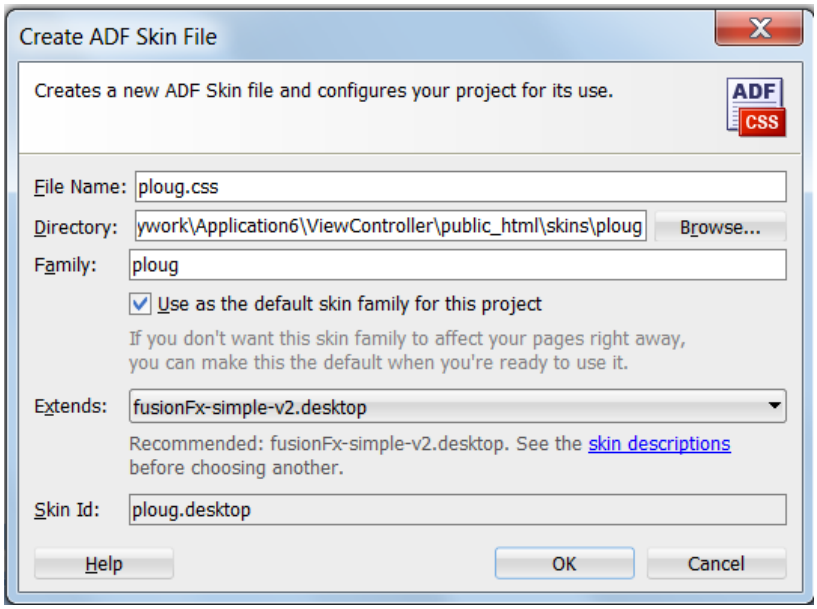
JDeveloper oferuje do wyboru kilka predefiniowanych skórek, które można wykorzystać w swoich aplikacjach lub użyć jako bazy dla własnych skórek. Najprostsza skórka, oferująca minimalne formatowanie, to skórka simple. Ze skórki simple wywodzą się skórki fusion i blafplus-medium. Rozwinięciem blafplus-medium jest skórka blafplus-rich, a z kolei skórka fusion jest korzeniem hierarchii swoich licznych modyfikacji. Obecnie generalnie preferowana jest skórka fusion i pojawiające się wraz z kolejnymi wersjami JDevelopera jej warianty, a skórki blafplus są zdeprecjonowane. Rys. 4 pokazuje wygląd naszego prostego formularza po zastosowaniu skórek simple i blafplus-rich (wygląd dla nowszej wersji skórki fusion – fusionFx został przedstawiony wcześniej na Rys. 1). Już na tym prostym przykładzie widać ascetyczny styl skórki simple w porównaniu z blafplus-rich i skórkami fusion. Różnice między fusion a blafplus-rich sprowadzają się głównie do kolorystyki – skórki fusion bazują na odcieniach niebieskiego, a blafplus-rich beżu i brązu. Ponadto, fusion intensywnie wykorzystuje obrazki jako tło w celu uzyskania efektu cieniowania.



Rys. 4. Wygląd przykładowego formularza dla skórki simple (po lewej) i blafplus-rich (po prawej)

3.3. Tworzenie własnych skórek

Tworzenie skórek ADF Faces w pierwszych wersjach JDevelopera 11g wspierających ADF Faces Rich Client wymagało „ręcznego" utworzenia pliku CSS skórki i pliku konfiguracyjnego skórek `trinidad-skins.xml`. Aktualnie dostępny jest kreator ADF Skin z kategorii Web Tier→JSF/Facelets, w którym informacje niezbędne do utworzenia skórki podaje się w trybie interaktywnym. Okno kreatora pokazuje Rys. 5. Tworząc nową skórkę należy podać przede wszystkim nazwę pliku CSS, nazwę rodziny skórek i istniejącą skórkę, którą nowo tworzona skórka będzie rozszerzać. Kreator nie pozwala na niepodanie skórki, którą nowa skórka będzie rozszerzać – skórki tworzone przez programistę zawsze muszą być oparte o jedną z istniejących już skórek.



Rys. 5. Okno kreatora skórek

Efektem działania kreatora jest plik CSS skórki niezawierający żadnych reguł, a jedynie poniższe deklaracje wiążące przestrzenie nazw komponentów z prefiksami, które będą używane w selektorach komponentów:

```
@namespace af "http://xmlns.oracle.com/adf/faces/rich";
@namespace dvt "http://xmlns.oracle.com/dss/adf/faces";
```

Poza samym plikiem CSS skórki tworzony lub uaktualniany (zależnie od tego czy tworzymy pierwszą czy kolejną skórkę w projekcie) jest XML-owy plik konfiguracyjny skórek `trinidad-skins.xml`. Jego zawartość dla ustawień kreatora z Rys. 5 jest następująca:

```
<?xml version="1.0" encoding="windows-1250"?>
<skins xmlns="http://myfaces.apache.org/trinidad/skin">
  <skin>
    <id>ploug.desktop</id>
    <family>ploug</family>
    <extends>fusionFx-simple-v2.desktop</extends>
    <render-kit-id>org.apache.myfaces.trinidad.desktop</render-kit-id>
```

```
    <style-sheet-name>skins/ploug/ploug.css</style-sheet-name>
    <bundle-name>ploug/view.skinBundle</bundle-name>
  </skin>
</skins>
```

W pliku konfiguracyjnym skórek wątpliwości nie powinny budzić elementy <extends> i <style-sheet-name>. To oczywiście nazwa rozszerzanej skórki i nazwa pliku CSS skórki. Oprócz pliku CSS z regułami formatującymi, skórka obejmuje też plik komunikatów – jest on wskazany elementem <bundle-name>. Szerszego komentarza wymagają pozostałe elementy potomne elementu <skin>. Tworząc skórkę tworzymy ją jako członka rodziny skórek (<skin-family>) obejmującej zestaw skórek dla różnych urządzeń. Urządzenie dla którego skórka jest przeznaczona wskazane jest elementem <render-kit-id>. Kreator tworzy skórkę dla przeglądarek desktopowych, świadczy o tym ostatni człon nazwy render-kit, który w połączeniu z nazwą rodziny skórek zgodnie z przyjętą konwencją tworzy identyfikator skórki (<id>). Oprócz proponowanego domyślnie przez kreator, aktualnie w ADF Faces dostępny jest jeszcze tylko jeden render-kit – dla urządzeń PDA.

W przypadku gdy skórka jest rozwijana i powstają kolejne jej wersje, zamiast nadawać im inne nazwy rodziny skórek można opatrzyć je kodem wersji i dodatkowo jedną z wersji wskazać jako domyślną, co ilustruje poniższy fragment pliku konfiguracyjnego skórek:

```
<skin>
  <id>ploug.desktop</id>
  <family>ploug</family>
  <extends>fusionFx-simple-v2.desktop</extends>
  <render-kit-id>org.apache.myfaces.trinidad.desktop</render-kit-id>
  <style-sheet-name>skins/ploug/ploug.css</style-sheet-name>
  <bundle-name>ploug/view.skinBundle</bundle-name>
  <version>
    <default>true</default>
    <name>v3</name>
  </version>
</skin>
```

Wskazując jako obowiązującą dla projektu skórkę występującą w kilku wersjach, można wskazać jawnie konkretną wersję lub pominąć określenie wersji, co spowoduje wybór wersji oznaczonej w konfiguracji skórki jako wersja domyślna.

3.4. Selektory i reguły

Plik CSS skórki stosuje składnię CSS, a więc zawiera reguły zbudowane z selektorów i deklaracji. Selektory oczywiście wskazują komponenty i ich różne aspekty, a deklaracje opisują sposób prezentacji. Selektory typowo odwołują się do nazw komponentów poprzedzonych prefiksem przestrzeni nazw, np. af|inputText, mogą też odwoływać się do klas i pseudoklas. Wśród dostępnych pseudoklas są standardowe pseudoklasy przewidziane przez standard CSS – :hover, :active, :visited i :selected oraz szereg innych np. :inactive i :highlighted opisujące stany wiersza tabelki z danymi (<af:table>). Wśród wykorzystywanych w skórkach ADF Faces pseudoklas szczególne znaczenie posiada pseudoklasa :alias, służąca do definiowania globalnych selektorów ustawiających właściwości (np. krój czcionki) dla całej skórki, upraszczając definicję skórki.

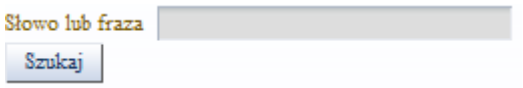
Charakterystyczną cechą selektorów skórki jest możliwość odwołania się to pseudoelementów identyfikujących obszary poszczególnych komponentów, które mają być poddane określonemu formatowaniu. Pseudoelementy w składni selektorów poprzedzane są podwójnym dwukropkiem, np. af|inputText::content identyfikuje zawartość komponentu pola tekstowego. Selektory mogą też odwoływać się do tzw. motywów (ang. themes), które umożliwiają różnicowanie formatowania (głównie kolorystyki) komponentów w zależności od kolorystyki kontenera, który je otacza.

Jeśli chodzi o właściwości używane w deklaracjach reguł skórki, to są to oczywiście standardowe właściwości CSS określające czcionkę, kolory, krawędzie, itd., ale oprócz nich dostępne są również właściwości specyficzne dla skórek ADF Faces, które nie trafiają do wynikowego arkusza stylów wysyłanego do przeglądarki, a są interpretowane przez framework po stronie serwera. Nazwy właściwości specyficznych dla skórek zaczynają się prefiksem -tr-, np. -tr-graphic-antialiasing.

Aby uzyskać założone formatowanie naszego formularza za pomocą skórki, należy w jej pliku CSS (po deklaracjach przestrzeni nazw) umieścić poniższe reguły:

```
af|inputText::content { background-color: #DDDDDD }
af|inputText::label { font-family: times; }
af|commandButton { font-family: times; }
```

W przypadku pola tekstowego, które jest komponentem złożonym, aby określić formatowanie specyficzne dla jego regionów w selektorach reguł zostały użyte wspomniane wcześniej pseudoelementy. Efekt formatowania formularza za pomocą skórki przedstawia Rys. 6. Oczywiście jest on identyczny z uzyskanym poprzez zagnieżdżone style. Jednakże dzięki zastosowaniu skórki zdefiniowane formatowanie obowiązywać będzie dla wszystkich pól tekstowych i przycisków na wszystkich stronach aplikacji, a ponadto specyfikacja formatowania jest odseparowana od definicji strony.



Rys. 6. Przykładowy formularz (formatowanie za pomocą skórki)

3.5. Przypisywanie skórki stronom aplikacji

Jak już zostało to wspomniane wcześniej, skórki w przeciwieństwie do arkuszy stylów CSS nie są przypisywane poszczególnym stronom, ale wskazywane jako obowiązujące dla całego projektu. Odbywa się to w pliku `trinidad-config.xml`. W naszym przypadku konfiguracja projektu była następująca:

```
<?xml version="1.0" encoding="windows-1250"?>
<trinidad-config xmlns="http://myfaces.apache.org/trinidad/config">
  <skin-family>ploug</skin-family>
</trinidad-config>
```

Należy podkreślić, że w pliku `trinidad-config.xml` wskazuje się rodzinę skórek, a nie konkretną skórkę poprzez jej identyfikator. Wynika to z faktu, że odpowiednia skórka z wskazanej rodziny jest wybierana automatycznie w oparciu o to jaki render-kit jest wykorzystywany, co z kolei zależy od typu urządzenia użytkownika końcowego.

Gdybyśmy przygotowali kilka wersji naszej skórki i chcieli jako obowiązującą dla projektu wskazać inną niż domyślna, należałoby określić ją w pliku `trinidad-config.xml` jawnie w sposób zaprezentowany poniżej:

```
<?xml version="1.0" encoding="windows-1250"?>
<trinidad-config xmlns="http://myfaces.apache.org/trinidad/config">
  <skin-family>ploug</skin-family>
  <skin-version>v2</skin-version>
</trinidad-config>
```

Nazwa rodziny skórek w pliku konfiguracyjnym projektu nie musi być statyczna. Możliwe jest użycie do wskazania nazwy rodziny skórek języka wyrażeń EL np. powołując się na wyznaczaną programowo właściwość komponentu zarządzanego. W ten sposób można zróżnicować skórki dla poszczególnych stron, jeśli z jakiegoś względu byłoby to pożądane. Użycie wyrażenia EL jako nazwy rodziny skórek w pliku konfiguracyjnym może też być wykorzystane do umożliwienia użytkownikom zmiany skórki w trakcie korzystania z aplikacji.

3.6. Testowanie i debugowanie skórek

W przypadku rozbudowanych skórek może się okazać w trakcie testowania aplikacji, że formatowanie w niektórych miejscach odbiega od zamierzonego, a znalezienie przyczyny takiego stanu rzeczy nie jest łatwe. Aby zidentyfikować reguły wpływające na formatowanie konkretnego elementu na stronie i w ten sposób odszukać przyczynę błędnego formatowania, można skorzystać z narzędzi takich jak np. dodatek Firebug dla przeglądarki Firefox.

Na potrzeby debugowania skórki zalecane jest wyłączenie kompresji arkuszy stylów generowanych przez mechanizm skórek. Domyślnie arkusze te są kompresowane ze względów wydajnościowych, ale przez to trudne jest powiązanie występujących w nich nazw klas z selektorami skórki. Wyłączenie kompresji arkuszy stylów zleca się odpowiednim parametrem kontekstowym w pliku konfiguracyjnym modułu webowego `web.xml`. Stosowny fragment kodu przedstawiony jest poniżej:

```
<context-param>
  <param-name>org.apache.myfaces.trinidad.DISABLE_CONTENT_COMPRESSION</param-name>
  <param-value>true</param-value>
</context-param>
```

Dzięki wyłączeniu kompresji, nazwy klas takie jak np. `x25` zastąpione będą nazwy nawiązujące do selektorów użytych w skórce takie jak np. `af_inputText_content`, co powinno znacząco ułatwić identyfikację reguł skórki odpowiadających za formatowanie poszczególnych elementów wynikowego dokumentu HTML.

3.7. Tworzenie archiwum JAR ze skórką

Skórka może być utworzona w ramach projektu zawierającego strony z niej korzystające. Jednakże gdy planujemy wykorzystanie raz zaprojektowanej skórki w wielu różnych projektach/aplikacjach, sensownym i zalecanym rozwiązaniem jest przygotowanie archiwum JAR ze skórką. Takie archiwum dołącza się później do projektu analogicznie do innych bibliotek. Po dołączeniu archiwum ze skórką do projektu, można ją wskazać jako obowiązującą w pliku `trinidad-config.xml`.

Tworząc archiwum JAR należy plik CSS skórki jak i plik konfiguracyjny `trinidad-skins.xml` z jej definicją umieścić w katalogu `META-INF`. Jeśli skórka zawiera obrazy, wykorzystywane jako tło lub ikony, należy je umieścić w podkatalogu `adf` katalogu `META-INF` (ewentualnie w podkatalogach tego podkatalogu).

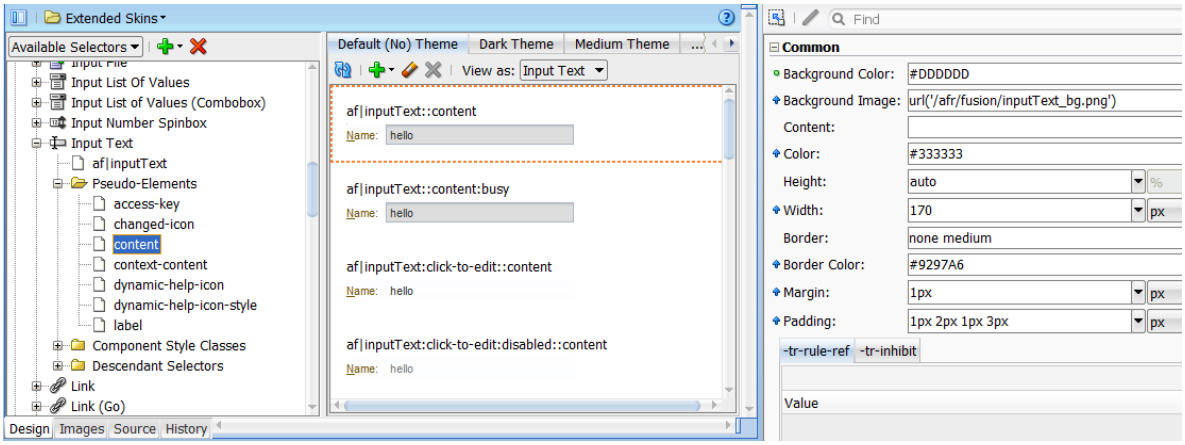
4. Edytor skórek

Projektanci skórek korzystający z wersji JDevelopera wcześniejszych niż 10.1.2 (Release 2) napotykali dwa istotne problemy. Pierwszy z nich to ogromna liczba różnych komponentów ADF Faces i ich regionów czy aspektów, które poddają się formatowaniu, połączona ze złożonością skórek dostarczonych z JDeveloperem stanowiących bazę do tworzenia własnych skórek. Problem ten jednak można było rozwiązać z pomocą dokumentacji. Znacznie trudniejszy do rozwiązania był problem, który napotykali twórcy skórek pragnący rozszerzyć jedną ze skórek `fusion`, w taki sposób

aby zmienić jej kolorystykę. Skórki fusion cechuje duża liczba różnych wykorzystywanych kolorów (nawet ponad 100) oraz bazowanie na obrazkach w celu uzyskania efektów cieniowanego tła. Ewentualna modyfikacja kolorystyki obrazków skórki fusion w jednym z narzędzi graficznych z jednoczesnym dopasowaniem obrazków do kolorów poszczególnych kontrolek określanych w skórce, choć z pewnością możliwa, była zadaniem bardzo skomplikowanym.

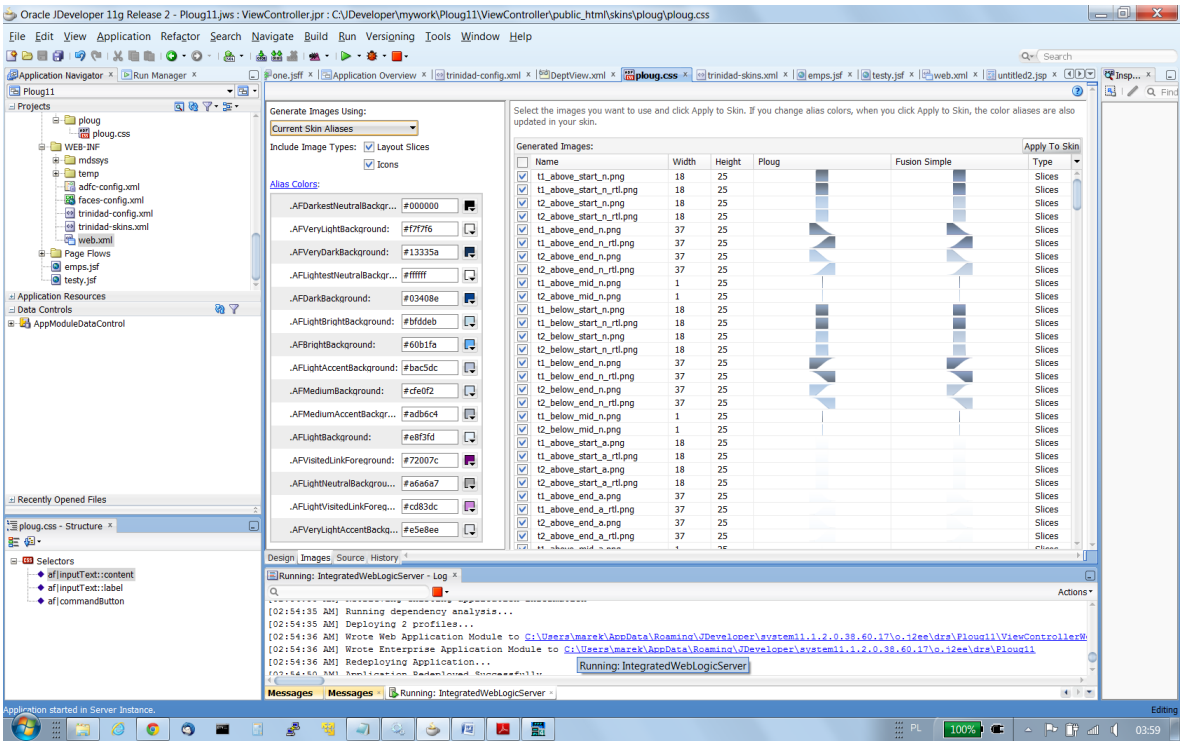
Odpowiedzią na oba wyżej wymienione problemy jest graficzny edytor skórek, który pojawił się w środowisku JDeveloper 11g Release 2. Wraz z nim dostarczone zostały dwie nowe odmiany skórki fusion: fusion-simple i fusionFx-simple. Cechuje je znacznie mniejsza liczba użytych kolorów i czcionek niż w wypadku skórek fusion i fusionFx, co ma ułatwić modyfikację ich kolorystyki. Programistów chcących tworzyć własne skórki na bazie skórki fusion zachęca się do wyboru jako bazy dla nowej skórki właśnie jednej z uproszczonych skórek fusion.

Edytor skórek ułatwia odszukanie odpowiednich selektorów dotyczących poszczególnych kontrolek ADF Faces, a następnie umożliwia specyfikację ich formatowania poprzez paletę właściwości. Mechanizm ten ilustruje Rys. 7.



Rys. 7. Edytor skórek: interaktywna konstrukcja reguł

Kluczową funkcją edytora skórek z punktu widzenia modyfikacji kolorystyki wariantów skórki fusion (dostosowaną konkretnie do wspomnianych wcześniej skórek w wersji uproszczonej) jest generator obrazków. Docelowa kolorystyka jest określana poprzez kilkanaście kolorów wiodących. Następnie na ich podstawie można zlecić zmianę kolorów wszystkich bądź tylko jawnie wskazanych obrazków modyfikowanej skórki. Okno generatora obrazków edytora skórek pokazuje Rys. 8.



Rys. 8. Edytor skórek: generacja obrazków dla zadanej kolorystyki

Ponieważ wersje frameworka ADF generalnie odpowiadają wersjom środowiska JDeveloper, użycie edytora skórek wbudowanego w środowisko JDeveloper w projektach opartych o starsze wersje frameworka ADF jest problematyczne, jeśli nie jest planowana migracja projektu do najnowszej wersji ADF. Aby umożliwić interaktywne tworzenie i edycję skórek dla wcześniejszych wersji frameworka ADF, Oracle udostępnił ADF Skin Editor również jako samodzielne narzędzie. Narzędzie to przypomina JDeveloper, ale jest okrojone do funkcjonalności obejmującej tworzenie i edycję skórek. Jego zaletą jest możliwość wskazania, z którą wersją ADF tworzona skórka ma współpracować. Przygotowaną skórkę można oczywiście umieścić w archiwum JAR i dołączyć do projektu rozwijanego w starszej wersji JDevelopera.

5. Podsumowanie

ADF Faces to bogata biblioteka komponentów graficznych do budowy interfejsu użytkownika, oparta o technologię JavaServer Faces (JSF). Stanowi ona wizytówkę frameworka ADF zapewniając profesjonalny wygląd aplikacji, jednocześnie oferując przyjazność i responsywność interfejsu użytkownika na poziomie aplikacji desktopowych. Twórcy aplikacji ADF mają pełną kontrolę nad wyglądem stron aplikacji, głównie dzięki mechanizmowi skórek. W przeciwieństwie do tradycyjnych arkuszy CSS, skórki odnoszą się nie do wynikowego HTML-a, a do komponentowego modelu stron, umożliwiając przez to bardziej precyzyjne i spójne określanie formatowania.

Tworzenie własnych skórek, a nawet modyfikacja tych, które dostarczone są wraz z frameworkiem nie jest zadaniem prostym i wymaga nie tylko znajomości technologii, ale i predyspozycji artystycznych. Oczywiście żadne narzędzie nie uczyni z każdego programisty grafika komputerowego, ale z pewnością może ułatwić tworzenie i edycję skórek od strony technicznej. O ile wcześniejsze wersje środowiska JDeveloper 11g pozostawiały wiele do życzenia w tym zakresie, to najnowsza wersja – JDeveloper 11g Release 2 stanowi istotny przełom dzięki edytorowi skórek. Edytor ten ułatwia odnalezienie właściwości odnoszących się do poszczególnych

aspektów prezentacji komponentów, a także umożliwia generację obrazków dla zadanych kolorów wiodących, co jest niezbędne do pełnej modyfikacji kolorystyki skórki fusion lub jej wariantów.

Bibliografia

[Apa11] Apache MyFaces Trinidad, http://myfaces.apache.org/trinidad/, pobrano: 30.08.2011

[Bor11] Bors L.: ADF 11g R2 : Skin Editor First Impressions, http://technology.amis.nl/blog/12213/adf-11g-r2-skin-editor-first-impressions, pobrano: 30.08.2011

[CSS2] Cascading Style Sheets Level 2 Revision 1 (CSS 2.1) Specification, W3C Recommendation, 2011

[CSS3] CSS current work & how to participate, http://www.w3.org/Style/CSS/current-work, pobrano: 30.08.2011

[Jel08] Jellema L.: ADF 11g – Why use skinning? Why not use simple external CSS stylesheets?, http://technology.amis.nl/blog/4169/adf-11g-why-use-skinning-why-not-use-simple-external-css-stylesheets, pobrano: 30.08.2011

[Jel09] Jellema L.: Using ADF Faces 11g Skinning for setting the styles of specific component instances or groups of instances, http://technology.amis.nl/blog/5722/using-adf-faces-11g-skinning-for-setting-the-styles-of-specific-component-instances-or-groups-of-instances, pobrano: 30.08.2011

[Nim11] Nimphius F.: Suggested skin editor workflow, http://blogs.oracle.com/jdevotnharvest/entry/suggested_skin_editor_workflow, pobrano: 30.08.2011

[NiMu09] Nimphius F., Munsinger L.: Oracle Fusion Developer Guide: Building Rich Internet Applications with Oracle ADF Business Components and Oracle ADF Faces, McGraw-Hill Osborne Media, 2009

[Ora11a] Oracle Fusion Middleware Fusion Developer's Guide for Oracle Application Development Framework 11g Release 2 (11.1.2.0.0), 2011

[Ora11b] Oracle Fusion Middleware Web User Interface Developer's Guide for Oracle Application Development Framework 11g Release 2 (11.1.2.0.0), 2011

[Ora11c] Oracle Fusion Middleware Skin Editor User's Guide for Oracle Application Development Framework 11g Release 2 (11.1.2.0.0), 2011

[Ora11d] Changing an Application's Look and Feel by Using Skins, http://download.oracle.com/docs/cd/E18941_01/tutorials/jdtut_11r2_83/jdtut_11r2_83.html, pobrano: 30.08.2011

[Shm11] Shmeltzer S.: ADF Faces Skin Editor - How to Work with It, http://blogs.oracle.com/shay/entry/adf_faces_skin_editor_how, pobrano: 30.08.2011

[Zak07] Zakrzewicz M.: Charakterystyka techniczna ADF Faces 11g Rich Client, Materiały XIII konf. PLOUG, Zakopane, 2007