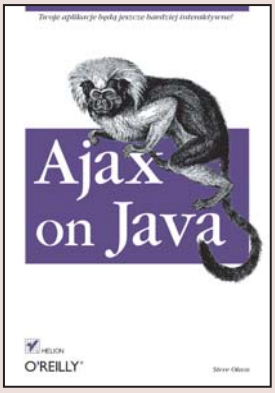
Ajax on Java



Autor: Steven Olson
Tłumaczenie: Sławomir Dzieniszewski
ISBN: 978-83-246-1110-2
Tytuł oryginału: Ajax on Java
Format: B5, stron: 232

Twoje aplikacje jeszcze bardziej interaktywne!

- Jak integrować funkcje Ajaksa z aplikacjami JSP?
- Jak korzystać z dostępnych bibliotek znaczników i tworzyć własne?
- Jak łączyć techniki Ajax i Struts?

Technologia Ajax oparta na kodzie Java gwarantuje uzyskanie prawdziwej interaktywnoś ci witryny internetowej, wysoce komfortowej zarówno dla użytkownika, jak i jej administratora. Wieloś ć zestawów narzędziowych i technik umożliwia maksymalne uproszczenie i przyspieszenie pracy webmastera. Zaimplementowanie Ajaksa w aplikacjach pisanych w języku Java pozwala na uzyskanie niemal wszystkich efektów potrzebnych do sprawnego funkcjonowania dynamicznej strony WWW bez koniecznoś ci wykorzystywania innych technologii.

Książka „Ajax on Java" to wprowadzenie do technologii Ajax, które pokazuje, jak wzbogacać o funkcje ajaksowe aplikacje oparte na serwletach, aplikacje JSP, JSF i inne. Dzięki temu podręcznikowi nauczysz się tworzyć bardziej interaktywne, dynamiczne i efektowne strony internetowe poprzez wyeliminowanie pracochłonnego wpisywania danych przez użytkownika i irytującego oczekiwania na odś wieżenie strony. Poznasz również kilka sposobów organizowania komunikacji pomiędzy klientem a serwerem, w tym wykorzystanie formatów JSON, umożliwiających przesyłanie danych o bardziej złożonej strukturze.

Krótko mówiąc, książka „Ajax on Java" podniesie Twoje umiejętnoś ci programowania na wyższy poziom.

- Budowanie i instalowanie aplikacji Ajax
- Integrowanie funkcji Ajax z aplikacjami JSP
- Metody tworzenia dokumentów XML
- Tworzenie biblioteki znaczników
- Pobieranie i instalowanie biblioteki Ajax
- Pisanie kodu JSP z wykorzystaniem Struts-Layout
- Konfigurowanie serwletów
- Wykorzystywanie zestawu narzędziowego GWT
- Wyszukiwanie błędów w kodzie aplikacji

Ajax on Java – komfort webmasterów i użytkowników!

Wydawnictwo Helion
ul. Kościuszki 1c
44-100 Gliwice
tel. 032 230 98 63
e-mail: helion@helion.pl

Spis treści

ROZDZIAŁ 2.

JavaScript i Ajax

Tajemnica technologii Ajax polega na sprytnym wykorzystaniu języka JavaScript. Ajax nie jest szkieletem programowania dla stron WWW tak jak Struts czy Tapestry i pod tym akronimem tak naprawdę nie ukrywa się żadna nowa cudowna technologia. Sekret Ajax polega na bezpośrednim komunikowaniu się za pomocą języka JavaScript z serwerem stron WWW, dzięki czemu unika się cyklu zatwierdzenie danych – odpowiedź, tak dobrze znanego wszystkim użytkownikom stron WWW.

Programiści języka Java zazwyczaj unikają języka JavaScript. Z różnych powodów, lepszych i gorszych. Oczywiście dodanie kolejnej warstwy skryptowej do strony JSP zwiększa tylko zamieszanie. Niemniej kod JavaScript wykonywany jest bezpośrednio przez przeglądarkę internetową i dlatego jest bardzo szybki. Nie ma potrzeby oczekiwania, aż serwer wygeneruje odpowiedź: kod JavaScript jest w stanie prawie natychmiast wygenerować wynik i odpowiednio aktualizować stronę.

Technologia Ajax dodaje tu interakcję z serwerem, jednak bez konieczności zatwierdzania (i wysyłania) danych przyciskiem *Submit*. Kiedy potrzebne są nowe dane od serwera, strona WWW z kodem JavaScript po prostu wysyła żądanie, a serwer odsyła z powrotem odpowiednie dane — tym razem nie jest to jednak nowa strona w kodzie HTML. Serwer zwraca dane, które kod JavaScript będzie mógł wyświetlić na bieżącej, już załadowanej stronie. Efekt jest taki, że nasza aplikacja WWW zaczyna bardziej przypominać zwykłą aplikację instalowaną na komputerze. Mówiąc w skrócie, korzystając z technologii Ajax, możemy osiągnąć na naszych stronach WWW poziom interaktywności zbliżony do tego znanego z profesjonalnych aplikacji instalowanych na komputerze.

Celem tej książki nie jest nauczenie Czytelnika programowania w języku JavaScript ani nawet omawianie jego wad i zalet. Zakładam tutaj, że każdy z Czytelników ma już jakieś doświadczenie z językiem JavaScript. Ci, dla których jest on nowością, powinni zajrzeć do książki *JavaScript. Przewodnik programisty* autorstwa Davida Flanagana (wydawnictwo RM). Jest to najlepszy obecnie dostępny przewodnik po języku JavaScript. Mimo iż język JavaScript różni się od Javy, niemniej programiści języka Java nie powinni mieć większych problemów ze zrozumieniem kodu JavaScript. Jak łatwo się będzie przekonać, kod JavaScript zaprezentowany w tym rozdziale jest dość prosty. Dopóki składnia języka jest dla Czytelnika zrozumiała, nie ma potrzeby dokładnego studiowania języka JavaScript.

Tworzenie aplikacji

Zaczniemy od przygotowania kompletnego kodu HTML i JavaScript naszej pierwszej aplikacji. Będzie to prosta strona WWW wyświetlająca liczbę dziesiętną odpowiadającą każdemu znakowi. Następnie oddzielimy kod JavaScript od kodu HTML i przyjrzymy się mu dokładnie.

Kod HTML ukazany został na listingu 2.1.

Listing 2.1. index.html

```
<html>
<head>
    <link rel="stylesheet" type="text/css" href="style.css">
    <SCRIPT language="JavaScript" src="ajax.js"></SCRIPT>
    <title>Ajax on Java, Rozdział 2 przykład</title>
</head>
<body onload="focusIn( );">
    <h1> AJAXOWY DEKODER ZNAKÓW </h1>
    <h2> Wciśnij klawisz, by poznać jego kod liczbowy. </h2>
    <table>
        <tr>
            <td>
                Tu podaj klawisz ->
                <input type="text" id="key" name="key"
                       onkeyup="convertToDecimal( );">

            </td>
        </tr>
    </table>
    <br />
    <table>
        <tr>
            <td colspan="5" style="border-bottom:solid black 1px;">
                Wciśnięty klawisz:
                <input type="text" readonly id="keypressed">
            </td>
        </tr>
        <tr>
            <td> Kod dziesiętnie </td>
        </tr>
        <tr>
            <td><input type="text" readonly id="decimal"></td>
        </tr>
    </table>
</body>
</html>
```

W większości jest to standardowy kod HTML. Zawiera on tylko dwa odwołania do kodu JavaScript: do funkcji `focusIn()` i `convertToDecimal()`. Funkcja `focusIn()` po prostu umieszcza kursor od razu w odpowiednim polu służącym do wprowadzania danych, dzięki czemu użytkownik nie musi go tam sam przesuwać myszą.

Funkcja `convertToDecimal()` będzie natomiast naszą bramą do świata technologii Ajax. Listing 2.2 prezentuje kod JavaScript obsługujący naszą stronę WWW, przechowywany w pliku *ajax.js*.

Listing 2.2. ajax.js

```
var req;

function convertToDecimal( ) {
    var key = document.getElementById("key");
```

```
    var keypressed = document.getElementById("keypressed");
    keypressed.value = key.value;
    var url = "/ajaxdecimalcodeconverter/response?key=" + escape(key.value);
    if (window.XMLHttpRequest) {
        req = new XMLHttpRequest( );
    }
    else if (window.ActiveXObject) {
        req = new ActiveXObject("Microsoft.XMLHTTP");
    }
    req.open("Get",url,true);
    req.onreadystatechange = callback;
    req.send(null);
}

function callback( ) {
    if (req.readyState==4) {
        if (req.status == 200) {
            var decimal = document.getElementById('decimal');
            decimal.value = req.responseText;
        }
    }
    clear( );
}
function clear( ) {
    var key = document.getElementById("key");
    key.value="";
}
function focusIn( ) {
    document.getElementById("key").focus( );
}
```

Przyjrzyjmy się funkcji `convertToDecimal()`, która w kodzie HTML jest naszym punktem wejścia do tego pliku z kodem JavaScript. Najważniejszym obiektem JavaScript, którego będziemy używać, jest obiekt `XMLHttpRequest`. Niestety podstawowy problem z językiem JavaScript polega na tym, że kod tego języka skryptowego nie będzie taki sam dla wszystkich przeglądarek. W przeglądarkach Mozilla, Firefox i Safari nowy obiekt `XMLHttpRequest` tworzymy w następujący sposób:

```
new XMLHttpRequest();
```

W przeglądarce Internet Explorer natomiast musimy użyć obiektu Active X:

```
new ActiveXObject("Microsoft.XMLHTTP");
```

Ponieważ nie jesteśmy w stanie z góry przewidzieć, z jakiej przeglądarki internetowej będą korzystać użytkownicy odwiedzający naszą stronę WWW, musimy przygotować kod, który będzie współpracował z wszystkimi najważniejszymi przeglądarkami. Po pierwsze, musimy ustalić, czy użytkownik korzysta z przeglądarki Internet Explorer, czy może jakiejś innej, takiej jak na przykład Firefox czy Mozilla. Zajmuje się tym następujący fragment kodu:

```
if (window.XMLHttpRequest) {
    req = new XMLHttpRequest( );
}
else if (window.ActiveXObject) {
    req = new ActiveXObject("Microsoft.XMLHTTP");
}
```

Kod ten po prostu tworzy (w zależności od przeglądarki) odpowiedni obiekt `req`, który wykorzystamy do zbudowania naszej strony Ajax.

Przyjrzyjmy się teraz części kodu, która wykonuje rzeczywistą pracę. W kolejnym rozdziale będziemy korzystać z kodu prezentowanego tutaj w pliku *ajax.js*, przyjrzyjmy się mu więc uważnie i zbadajmy mechanizm komunikacji z serwerem. Ponieważ jesteśmy programistami Javy, program, z którym kod JavaScript się komunikuje, będzie serwletem, niemniej dla strony WWW nie ma to znaczenia.

Funkcja `convertToDecimal()` najpierw pobiera z formularza łańcuch (`String`), a następnie przypisuje zmiennej `url` wartość `"/ajaxdecimalcodeconverter/response?key=..."`. Na koniec wysyła ten adres URL serwerowi (w naszym przypadku serwletowi) i oczekuje na odpowiedź (którą będzie dziesiętna wartość kodu przypisana klawiszowi). Inaczej niż na zwykłej stronie nie wysyłamy danych serwerowi dopiero po wciśnięciu przycisku zatwierdzającego *Submit*. Tym razem wysyłamy dane w sposób asynchroniczny (to znaczy gdy tylko użytkownik wciśnie klawisz, którego kod chcemy wyświetlić).

Po bloku `if...else`, w którym ustalamy, z jakiej przeglądarki korzysta użytkownik i po przygotowaniu odpowiedniego obiektu `req`, otwieramy połączenie z serwerem za pomocą następującego wywołania:

```
req.open("Get",url,true);
```

Przyjrzyjmy się trzem parametrom użytej tu funkcji `req.open()`:

`"Get"`
: Pierwszy parametr informuje JavaScript, czy wysyłać serwerowi żądanie za pomocą funkcji `HTTPPost()`, czy `HTTPGet()`. Metoda `HTTPPost()` ukrywa parametry w żądaniu, natomiast metoda `HTTPGet()` umieszcza parametry w adresie URL tak, że są widoczne dla każdego. W tym przykładzie wybrałem funkcję `HTTPGet()`, ponieważ łatwiej wtedy zorientować się, jakie parametry zostały przesłane serwerowi, a parametrów jest niezbyt wiele. Gdybyśmy wysyłali długi i złożony zestaw parametrów, skorzystałbym z metody „Post”[1].

`url`
: Drugi parametr to adres URL, który przesyłamy serwerowi. Adres ten przygotowaliśmy wcześniej w naszej metodzie.

`true`
: Ostatni parametr określa, czy mamy do czynienia z wywołaniem asynchronicznym, czy nie. Kiedy parametrowi temu zostanie przypisana wartość `true`, żądanie wysyłane jest w sposób asynchroniczny. Podczas tworzenia aplikacji Ajax zawsze będziemy przypisywać temu znacznikowi wartość `true`. W uproszczeniu mówiąc, oznacza on „niczego nie zatrzymuj, po prostu poinformuj mnie, kiedy dane powrócą”.

Alternatywą jest przypisanie trzeciemu parametrowi funkcji `req.open()` wartości `false` (fałsz). Spowodowałoby to zatrzymanie przeglądarki do momentu, aż serwer zwróci odpowiednie dane — o ile oczywiście je odeśle (nigdy nie ma takiej gwarancji). Trudno w takim przypadku oczekiwać pełnej satysfakcji klienta, dlatego też będziemy zawsze przypisywać trzeciemu parametrowi wartość `true` (prawda).

[1] Wybiegam tu trochę naprzód, niemniej warto wiedzieć, że metody `Get` należy używać tylko wtedy, gdy żądanie nie zmienia w żaden sposób danych przechowywanych na serwerze. W tym przypadku sytuacja jest oczywista. Używanie metody `Get`, gdy zmieniamy dane na serwerze, byłoby sporym błędem (na przykład jeśli wysyłamy nowe dane lub usuwamy dane już istniejące). W tym przypadku należy użyć metody `Post`.

Teraz zwróćmy uwagę na następującą instrukcję:

```
req.onreadystatechange=callback;
```

Ten wiersz umożliwia nam używanie wywołania funkcji w sposób asynchroniczny. Informujemy obiekt `req`, by przywoływał funkcję zwrotną `callback()` za każdym razem, gdy nastąpi zmiana stanu. Dzięki temu będziemy przetwarzać dane nadchodzące z serwera od razu, gdy tylko powrócą do przeglądarki. Zostaniemy poinformowani, gdy tylko coś się wydarzy.

Co to takiego funkcja zwrotna?

Funkcja zwrotna (ang. *callback*) to wykonywalny kod przesyłany jako parametr innej funkcji. W naszym przypadku przesyłamy do obiektu `XMLHttpRequest` kod informujący, jaką funkcję należy przywołać, przy zmianie stanu na `ready` (gotowy).

Kod JavaScript generuje żądanie, które wysyłane jest do serwletu. Kiedy serwlet odeśle odpowiednie informacje, przywołana zostanie funkcja zwrotna. Dzięki temu funkcja zwrotna będzie mogła wyświetlić te nowe informacje użytkownikowi. Jaką funkcję należy przywołać, określiliśmy za pomocą następującego kodu:

```
req.onreadystatechange = callback;
```

Jest to naprawdę użyteczne narzędzie programistyczne. Od tej pory użytkownik nie musi już czekać na załadowanie nowej strony WWW (lub przeładowanie starej), ponieważ gdy tylko nadejdą nowe dane, zostaną wyświetlone na bieżącej stronie.

Ostatnia instrukcja funkcji `convertToDecimal()` wysyła żądanie:

```
req.send(null);
```

Teraz przyjrzyjmy się funkcji zwrotnej `callback()`:

```
function callback( ) {
    if (req.readyState==4) {
        if (req.status == 200) {
            if (window.XMLHttpRequest) {
                nonMSPopulate( );
            }
            else if (window.ActiveXObject) {
                msPopulate( );
            }
        }
    }
    clear( );
}
```

Ta funkcja sprawdza stan gotowości `readyState` i kod stanu zwrócony przez serwer. Stan gotowości `readyState` może przyjmować jedną z pięciu wartości podanych w tabeli 2.1.

Tabela 2.1. Dopuszczalne wartości readyState

Wartość	Stan
0	Uninitialized (nieinicjowane)
1	Loading (w trakcie ładowania)
2	Loaded (załadowane)
3	Interactive (interaktywnie)
4	Complete (zakończone)

Funkcja zwrotna `callback()` przywoływana jest przy każdej zmianie stanu, co nie zawsze może nam odpowiadać. Nie chcemy przecież nic robić, dopóki żądanie nie zostanie zakończone, dlatego zdecydowaliśmy, że będziemy czekać, dopóki stan nie zmieni się na `Complete` (`req.readyState == 4`).

Kolejny test `req.status == 2000` pozwala nam upewnić się, że obiekt żądania `HTTPRequest` zwrócił stan OK (kod `200`). Jeśli strona nie zostanie odnaleziona, kod stanu (status) będzie równy 404. W tym przykładzie kod powinien być aktywowany tylko wtedy, gdy żądanie zostanie zakończone (stan `Complete`). Warto zauważyć, że wartość stanu `readyState` równa `4` nie gwarantuje nam, że żądanie zostało zakończone (zrealizowane) prawidłowo. Aby sprawdzić, jaki naprawdę był jego rezultat, musimy sprawdzić kod `req.status`.

Kompletną listę kodów stanu protokołu HTTP można znaleźć pod adresem *http://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html*.

W jaki sposób przywoływana jest nasza funkcja JavaScript?

Napisaliśmy użyteczną funkcję JavaScript `convertToDecimal()`, która robi kilka interesujących rzeczy: wysyła żądanie do serwera bez kłopotania użytkownika i sprawia, że odpowiedź serwera zostaje dodana do bieżącej strony WWW. W jaki jednak sposób przywołuje się funkcję `convertToDecimal()`? Odpowiedź jest prosta: przeglądarka przywołuje ją, kiedy wykryje zdarzenie `keyup` w polu „Tu podaj klawisz ->". Oto kompletny kod HTML dla tego pola:

```
<input type="text" id="key" name="key" onkeyup="convertToDecimal( );">
```

Kod `onkeyup="convertToDecimal( );"` informuje przeglądarkę, żeby przywoływała funkcję JavaScript `convertToDecimal()` zawsze, gdy tylko użytkownik wciśnie i zwolni klawisz w tym polu.

Dlaczego korzystamy ze zdarzenia `onkeyup` (zwolnienie klawisza), zamiast ze zdarzenia `onkeypress` (wciśnięcie klawisza)? Jest to istotny niuans programistyczny, nad którym warto się przez chwilę zastanowić. Z pozoru mogłoby się wydawać, że w naszej aplikacji równie dobrze sprawdzałoby się zdarzenie `onkeypress`, tak jednak nie jest. Zarówno zdarzenie `onkeypress`, jak i `onkeydown` uruchamiane są, zanim efekt akcji użytkownika zmieni zawartość pola, wysyłając to, co znajdowało się w polu przed zajściem zdarzenia. Ponieważ chcemy odczytać wprowadzony przez użytkownika znak, musimy użyć zdarzenia `onkeyup` (zwolnienie klawisza), które zachodzi już po umieszczeniu znaku w polu.

W jaki sposób pobieramy wartość wciśniętego klawisza?

Gdy już kontrola zostanie przekazana funkcji `convertToDecimal()`, wykonujemy następujące wywołanie:

```
var key = document.getElementById("key");
```

W tym momencie obiekt o identyfikatorze id równym `key` zawierać będzie kod wciśniętego klawisza w postaci liczby dziesiętnej. Wszystko co nam pozostało, to pobrać wartość, którą zawiera obiekt o nazwie key. Wartość ta przechowywana jest w parametrze `value` elementu `key`, tak więc zmienna `key.value` zawierać będzie wartość (dziesiętny kod) wciśniętego właśnie klawisza.

Gdy już pobierzemy tę wartość, należy umieścić ją w odpowiednim polu, w którym ma zostać wyświetlona. Pozwoli to nam jednocześnie oczyścić pole, w którym użytkownik wciskał klawisz. Pole służące do wyświetlania kodu klawisza nazwaliśmy `keypressed`. Pobiera się je w następujący sposób:

```
var keypressed = document.getElementById("keypressed");
```

Kolejnym krokiem jest przypisanie wartości zmiennej `key` do pola `keypressed`:

```
keypressed.value = key.value;
```

Formatowanie strony

Ostatnim krokiem niezbędnym do przygotowania naszej aplikacji jest utworzenie pliku CSS, który sformatuje wyświetlaną stronę. Plik ten prezentujemy na listingu 2.3.

Listing 2.3. style.css

```
body {
    font-family: Arial, Helvetica, sans-serif;
    font-size: small;
    text-align:center;
    background:#cbdada;
}
#keypressed{
    width:3O;
    border:none;
}
#key {
    width:2Opx;
    padding:O;
    margin:O;
    border:none;
    text-align:left
}
h1, h2 {
    font-size:12O%;
    text-align:center;
}
h2 {
    font-size:11O%
}
table, input {
    margin-left:auto;
    margin-right:auto;
    padding:Opx 1Opx;
    text-align:center;
    color:black;
    text-align:center;
    background: #aOf6f5;
    border:solid black 1px;
}
td {
    margin:1Opx 1Opx;
    padding: Opx 5px;
    border: none;
}
input {
    width: 8O;
    border: none;
    border-top:solid #999999 1px;
```

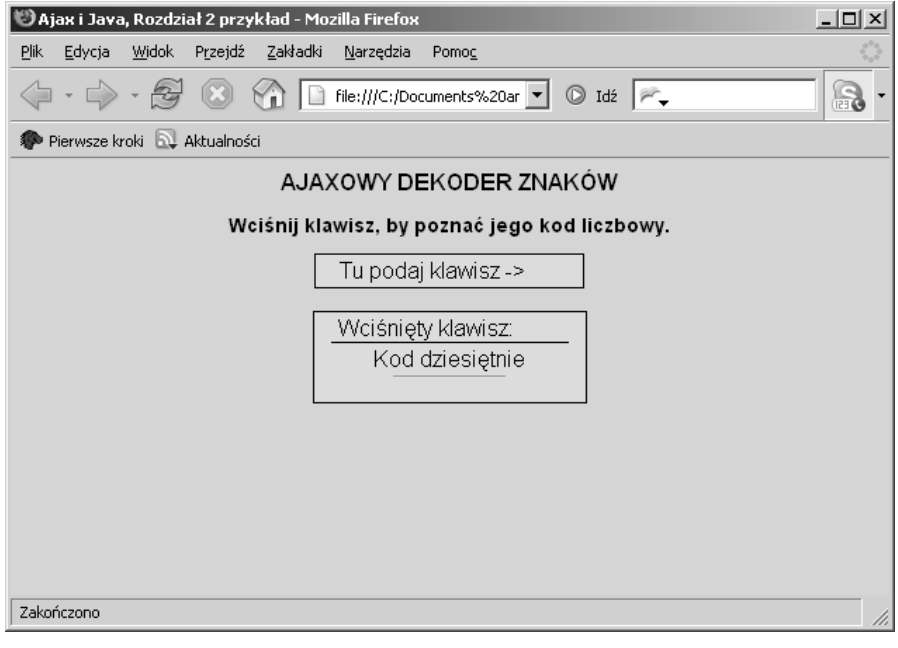
```
    font-size: 80%;
    color: #555555;
}
```

Uruchamianie przykładu

Po pobraniu kodu prezentowanego przykładu z witryny tej książki (*http://www.helion.pl/ksiazki/ajaxja.htm*) będzie można po prostu skopiować pliki z katalogu *ch02*. Niektórzy programiści wolą jednak samodzielnie wpisać kod przykładu, ponieważ pomaga im to zrozumieć działanie kodu.

Aby uruchomić ten program:

1. Zachowaj kod HTML z listingu 2.1 w pliku *index.html*.
2. Zachowaj kod JavaScript z listingu 2.2 w pliku *ajax.js* w tym samym katalogu.
3. Zachowaj kod arkusza stylów CSS z listingu 2.3 w pliku o nazwie *style.css* w tym samym katalogu.
4. Otwórz plik *index.html* za pomocą przeglądarki internetowej. W jej oknie powinna pojawić się strona podobna do tej przedstawionej na rysunku 2.1.



Rysunek 2.1. Program dekodujący klawisze oparty na mechanizmie Ajax w oknie przeglądarki

Kiedy wciśniemy klawisz, odpowiadająca mu litera lub znak pojawią się w polu *Wciśnięty klawisz*, a pole służące do wprowadzania danych zostanie wyczyszczone. Oczywiście ponieważ nie zaimplementowaliśmy jeszcze serwera, nie zobaczymy dziesiętnego kodu klawisza. Dlatego w następnym rozdziale zajmiemy się przygotowaniem serwletu, który zapełniać będzie pole *Kod dziesiętnie*.