

Zarządzanie obiektami typu **SessionBeans, RequestBean i ApplicationBeans**

Programowanie komponentowe 5

Okres życia obiektów

- Okres aplikacji oznacza czas życia obiektu typu `ApplicationBean1` i kończy się, gdy serwer kończy wykonanie aplikacji. Wartości przechowywane w obiekcie typu `ApplicationBean1` są dostępne przez cały czas życia **każdej sesji** i **każdej fazy request** należących do tej samej aplikacji.
- Okres sesji oznacza czas życia obiektu typu `SessionBean1` i zaczyna się, kiedy użytkownik po raz pierwszy wywołuje stronę aplikacji internetowej i kończy się, kiedy czas sesji kończy się zgodnie z wyznaczonym czasem typu `Timeout` lub gdy aplikacja internetowa przerywa sesję z powodu jej unieważnienia np. za pomocą metody `session.invalidate()`.
- Okres żądań (okres request) oznacza czas życia obiektu typu `RequestBean1` zaczyna się, kiedy użytkownik wysyła dane z formularza strony internetowej i kończy się, kiedy odpowiedź (faza response) jest w pełni zrealizowana.

Ostrzeżenie: Nie można użyć obiektu typu `RequestBean1`, jeżeli strona zawiera element `<redirect>` wewnątrz elementu `<navigation-case>` w regule nawigacji. (Te reguły są widoczne, kiedy kliknie się przycisk XML w edytorze nawigacji strony (`Page Navigation` editor)). Kiedy strona jest zatwierdzona (submit), element `<redirect>` przekierowuje do innej strony i kończy fazę request zanim osiągnięta nowa strona może wykorzystać wartości przechowywane w obiekcie typu `RequestBean1`.

Application

Session

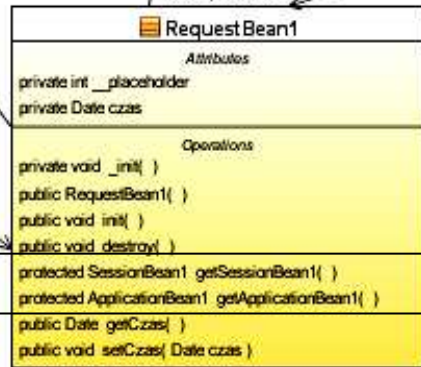
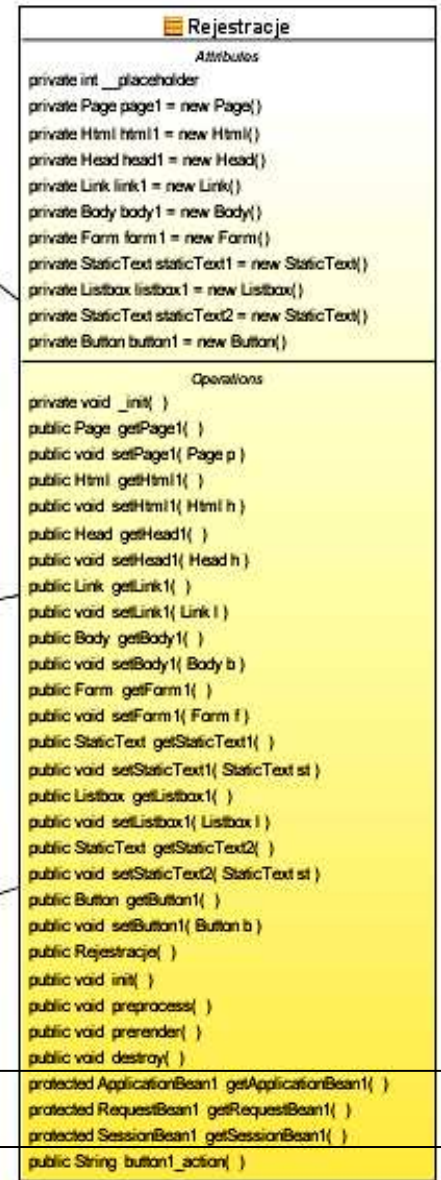


Session



Przykład demonstrujący czas życia obiektów typu **ApplicationBean1**, **SessionBean1** oraz **RequestBean1**

- **rejestracje**. Jest to pojemnik typu **HashMap** umieszczony w obiekcie typu **ApplicationBean1**, który przechowuje dane wszystkich rejestracji klientów tworzonych podczas kolejnego stanu sesji danego klienta
- **danerejestracji**. Jest to tablica elementów typu **Option**, umieszczona w obiekcie typu **ApplicationBean1**, do przechowania danych rejestracji w formie umożliwiającej wyświetlenie danych w komponencie typu **List Box**
- **rejestracja**. Jest to wartość typu logicznego tworzona podczas stanu sesji aplikacji, przechowywana w bieżącym obiekcie typu **SessionBean1**. Przyjmuje on wartość **false**, kiedy tworzy się nowy obiekt typu **SessionBean1**. Wartość **true** tego atrybutu jest ustawiana przy pierwszej rejestracji klienta podczas bieżącej sesji. Wartość **true** tego atrybutu wykorzystano do zablokowania ponownego rejestrowania się klienta w czasie jednej sesji
- **czas**. Czas ten jest odczytywany z czasu systemowego i przechowywany w bieżącym obiekcie typu **RequestBean1**
 - podczas rejestracji nowego klienta w obiekcie **rejestracje** i
 - żądania połączenia ze stroną do prezentowania danych zarejestrowanych klientów z obiektu **danerejestracji** oraz czasu bieżącej rejestracji z atrybutu **czas**



1. Informacje dotyczące realizacji programu w środowisku NetBeans 6.5

- 1) Z głównego menu wybierz opcję File > New Project.
- 2) W New Project, wybierz Java Web listy kategorii (Categories) i wybierz Web Application z listy projektów (Projects). Kliknij Next.
- 3) Nazwij projekt AplikacjaInternetowa7 (Project Name) i wybierz katalog (Browse dla Project Location). Kliknij Next.
- 4) Wybierz serwer aplikacji GlassFish V2 z listy Server oraz wersję Java EE z listy Java EE Version – domyślne wartości. Kliknij na Next
- 5) Wybierz Visual Web JavaServer Faces i naciśnij Finish.
- 6) Wybierz okno projektu (zakładka Projects) - zawiera ono układ plików typu BluePrints. Plik Page1.jsp jest stroną startową napisaną w języku JSP (zaznaczenie zielonym poziomym trójkątem) i znajduje się w podkatalogu „Web Pages”.

Ad. 1)

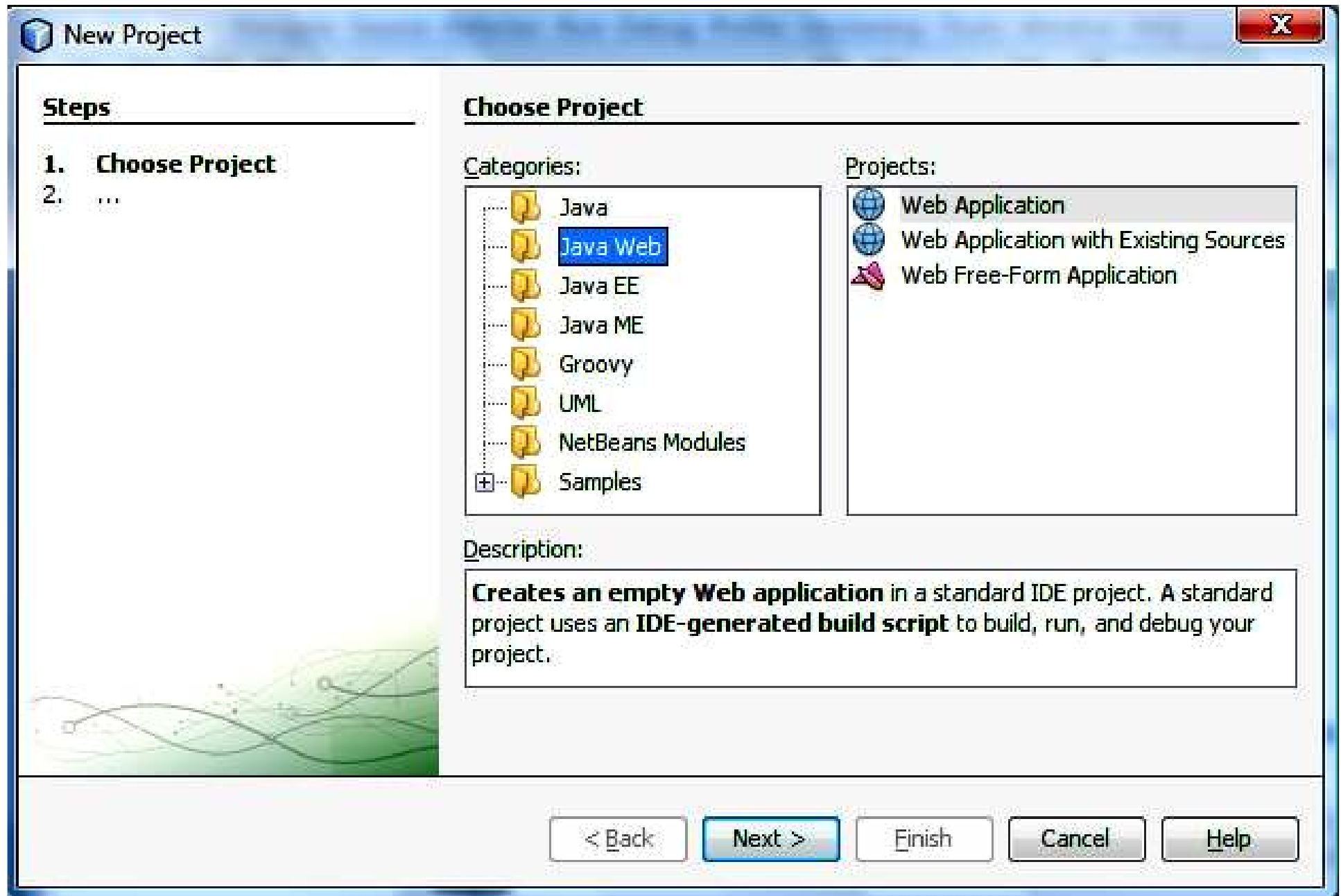
The screenshot displays the NetBeans IDE 6.5 interface for a project named "AplikacjaInternetowa7-Model". The main window shows a UML class diagram with the following classes and their attributes/operations:

- ApplicationBean1**:
 - Attributes: `private int __placeholder`, `private HashMap rejestracje`, `private Option danerejestracji[0..*]`
 - Operations: `private void __init()`, `public ApplicationBean1()`, `public void init()`, `public void destroy()`, `public String getLocaleCharacterEncoding()`, `public HashMap getRejestracje()`, `public void setRejestracje(HashMap rejestracje)`, `public void zarejestruj(String id_klient)`, `public Option[0..*] getDanerejestracji()`, `public void setDanerejestracji(Option danerejestracji[0..*])`, `public void zewarosc_talony_danerejestracji()`
- RequestBean1**:
 - Attributes: `private int __placeholder`, `private Date czas`
 - Operations: `private void __init()`, `public RequestBean1()`, `public void init()`, `public void preprocess()`, `public void prerender()`, `public void destroy()`
- SessionBean1**:
 - Operations: `private void __init()`, `public SessionBean1()`, `public void init()`, `public void passivate()`, `public void activate()`, `public void destroy()`, `protected ApplicationBean1 getApplicationBean1()`, `public boolean isRejestracja()`, `public void setRejestracja(boolean rejestracja)`

The diagram shows associations between these classes: `mSessionBean1` (association with `ApplicationBean1`), `mApplicationBean1` (association with `RequestBean1`), and `mRequestBean1` (association with `RequestBean1`). A "Page" class is also visible in the background diagram.

The bottom status bar shows the output of a report: "processed 344 elements 1 diagrams" and "Report Successful (total time: 11 seconds)".

Ad. 2)



Ad. 3)

New Web Application

Steps

1. Choose Project
- 2. Name and Location**
3. Server and Settings
4. Frameworks

Name and Location

Project Name:

Project Location:

Project Folder:

Use Dedicated Folder for Storing Libraries

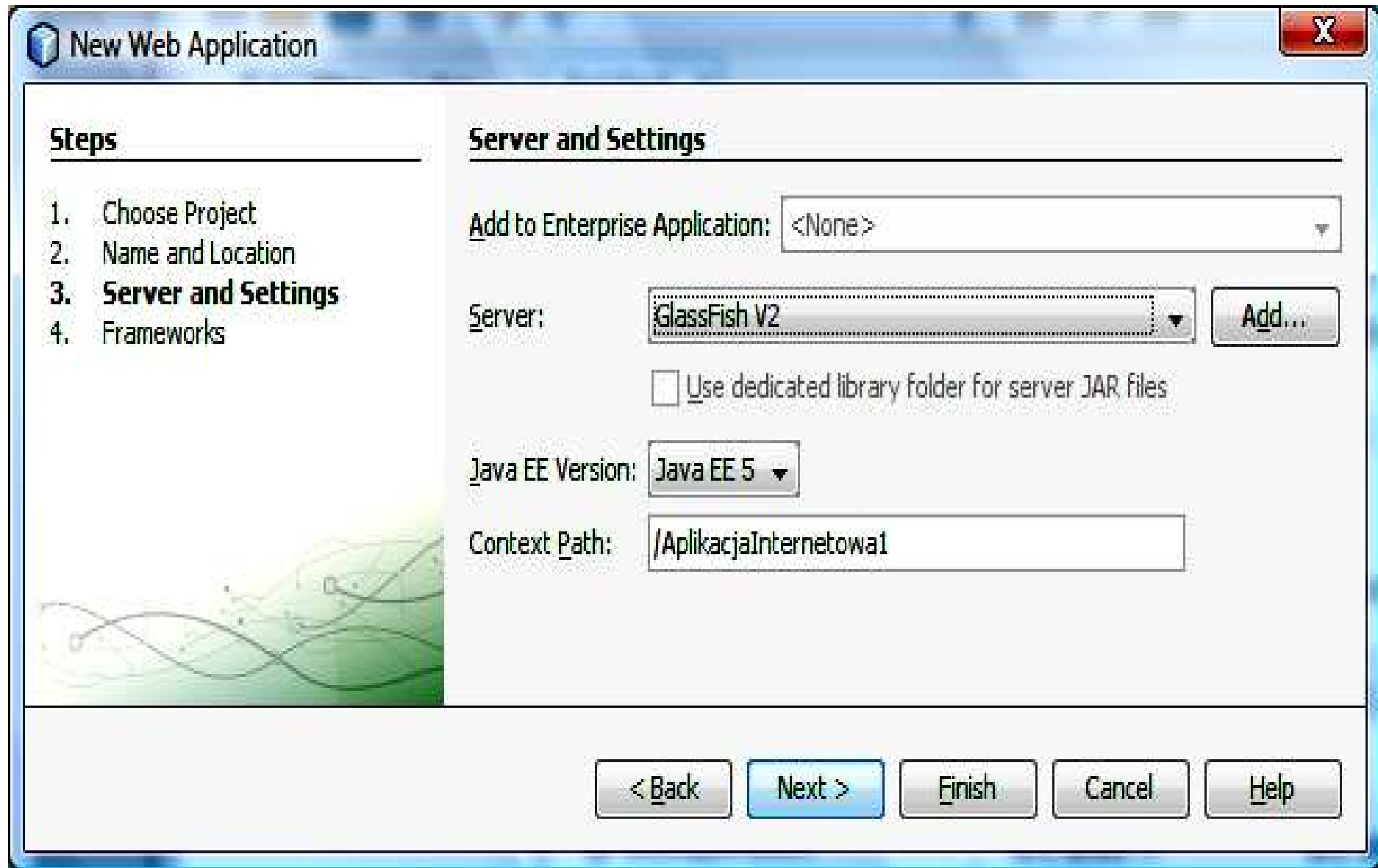
Libraries Folder:

Different users and projects can share the same compilation libraries (see Help for details).

Set as Main Project

< Back Next > Finish Cancel Help

Ad. 4)



The screenshot shows a 'New Web Application' wizard window. The title bar reads 'New Web Application' with a close button (X) on the right. The window is divided into two main sections: 'Steps' on the left and 'Server and Settings' on the right. The 'Steps' section lists four steps: 1. Choose Project, 2. Name and Location, 3. **Server and Settings** (highlighted), and 4. Frameworks. The 'Server and Settings' section contains several configuration options: 'Add to Enterprise Application:' with a dropdown menu set to '<None>'; 'Server:' with a dropdown menu set to 'GlassFish V2' and an 'Add...' button; a checkbox for 'Use dedicated library folder for server JAR files' which is unchecked; 'Java EE Version:' with a dropdown menu set to 'Java EE 5'; and 'Context Path:' with a text input field containing '/AplikacjaInternetowa1'. At the bottom of the window, there are five buttons: '< Back', 'Next >', 'Finish', 'Cancel', and 'Help'. The 'Next >' button is highlighted in blue.

New Web Application

Steps

1. Choose Project
2. Name and Location
3. **Server and Settings**
4. Frameworks

Server and Settings

Add to Enterprise Application: <None>

Server: GlassFish V2 Add...

Use dedicated library folder for server JAR files

Java EE Version: Java EE 5

Context Path: /AplikacjaInternetowa1

< Back Next > Finish Cancel Help

Ad. 5)

New Web Application

Steps

1. Choose Project
2. Name and Location
3. Server and Settings
4. **Frameworks**

Frameworks

Select the frameworks you want to use in your web application.

- Visual Web JavaServer Faces
- Spring Web MVC 2.5
- JavaServer Faces
- Struts 1.2.9
- Hibernate 3.2.5

Visual Web JavaServer Faces Configuration

Default Java Package:

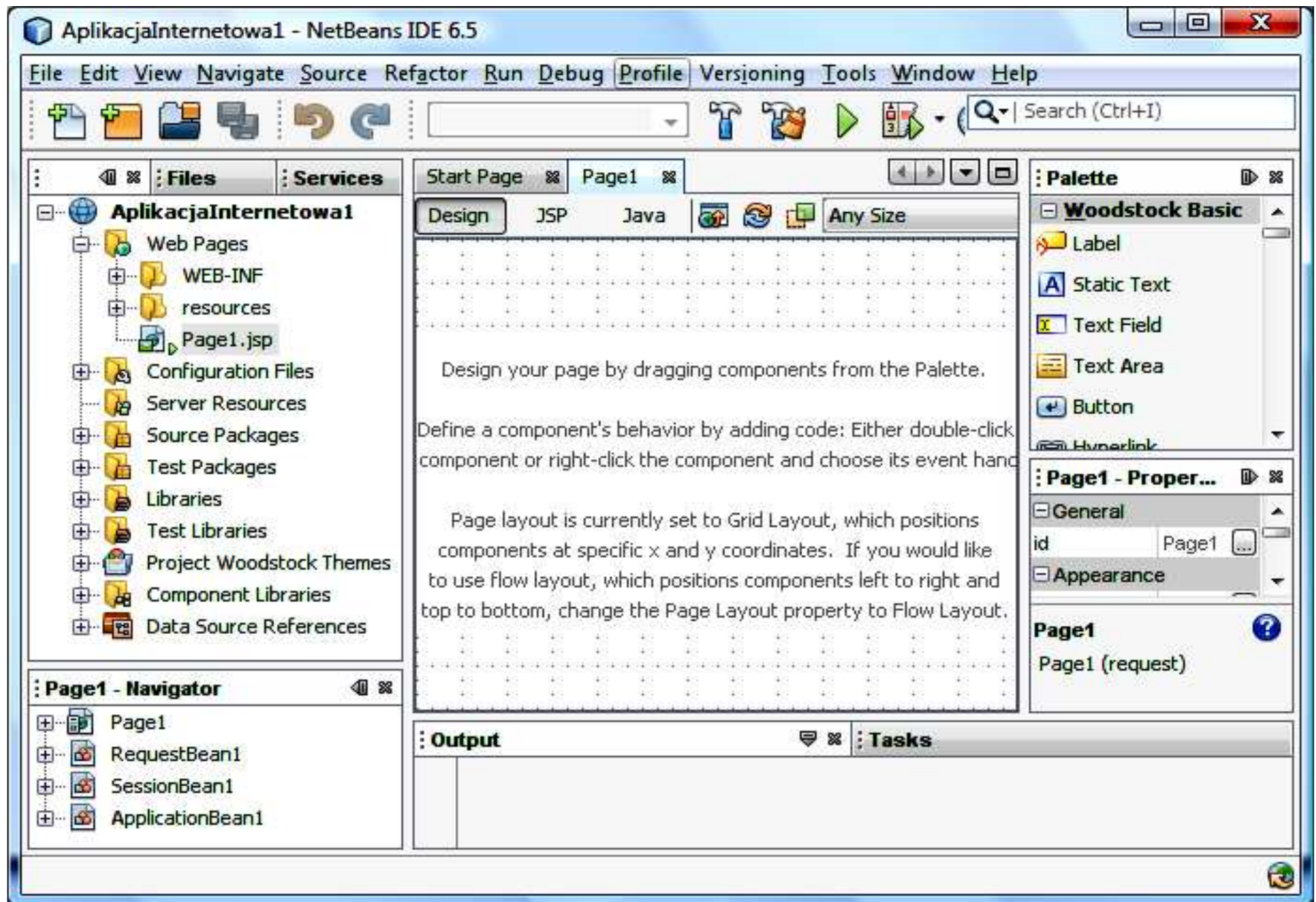
JSF Servlet Name:

Servlet URL Mapping:

Validate XML Verify Objects

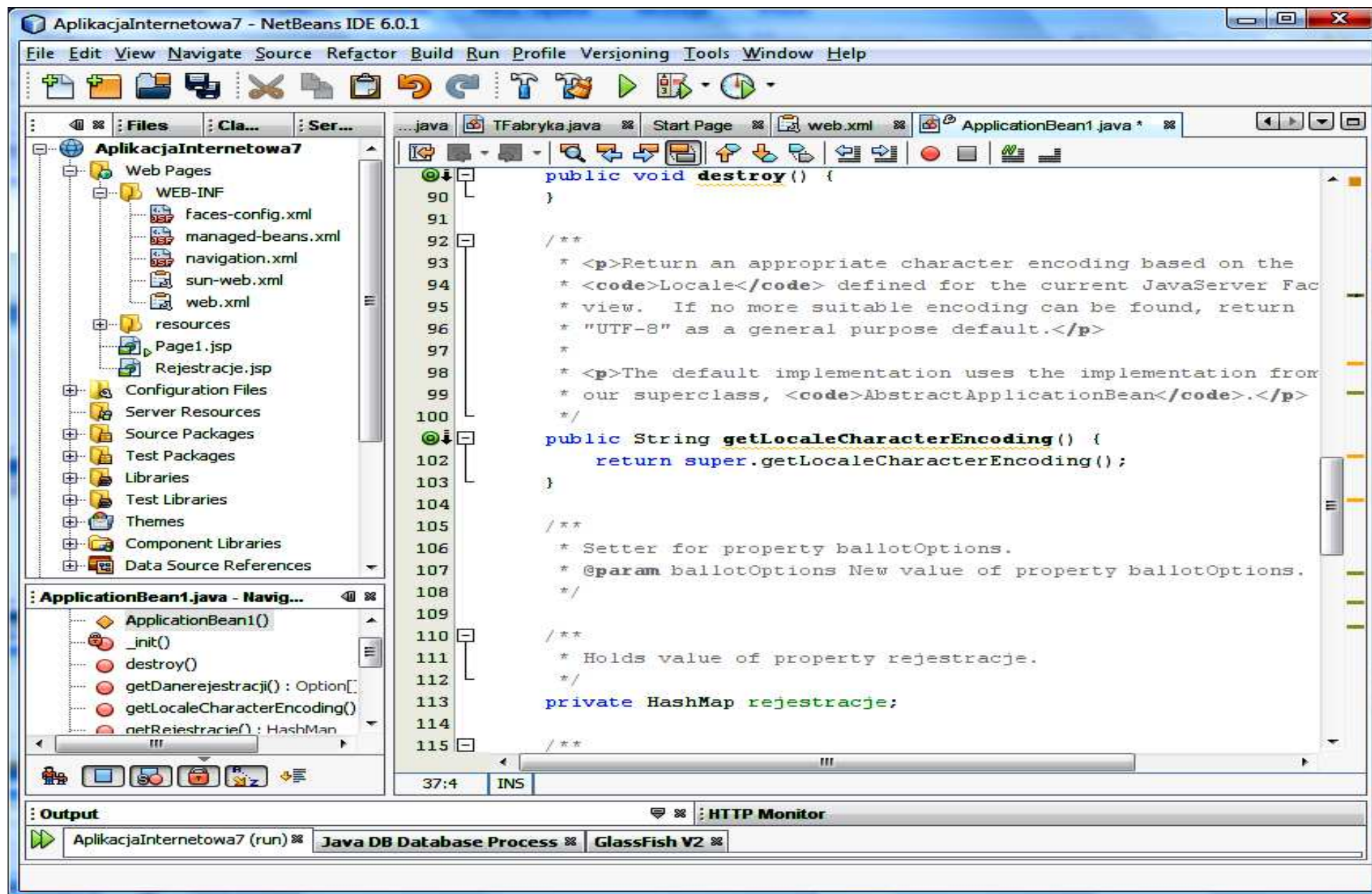
< Back Next > **Finish** Cancel Help

Ad. 6) Utworzenie aplikacji w Visual Web Pack – AplikacjaInternetowa7



2. Dodawanie atrybutu rejestracje typu HashMap do klasy typu ApplicationBean1 – ręczne wpisanie linii kodu.

Import brakujących pakietów - klawisze **CTRL+Shift+I**

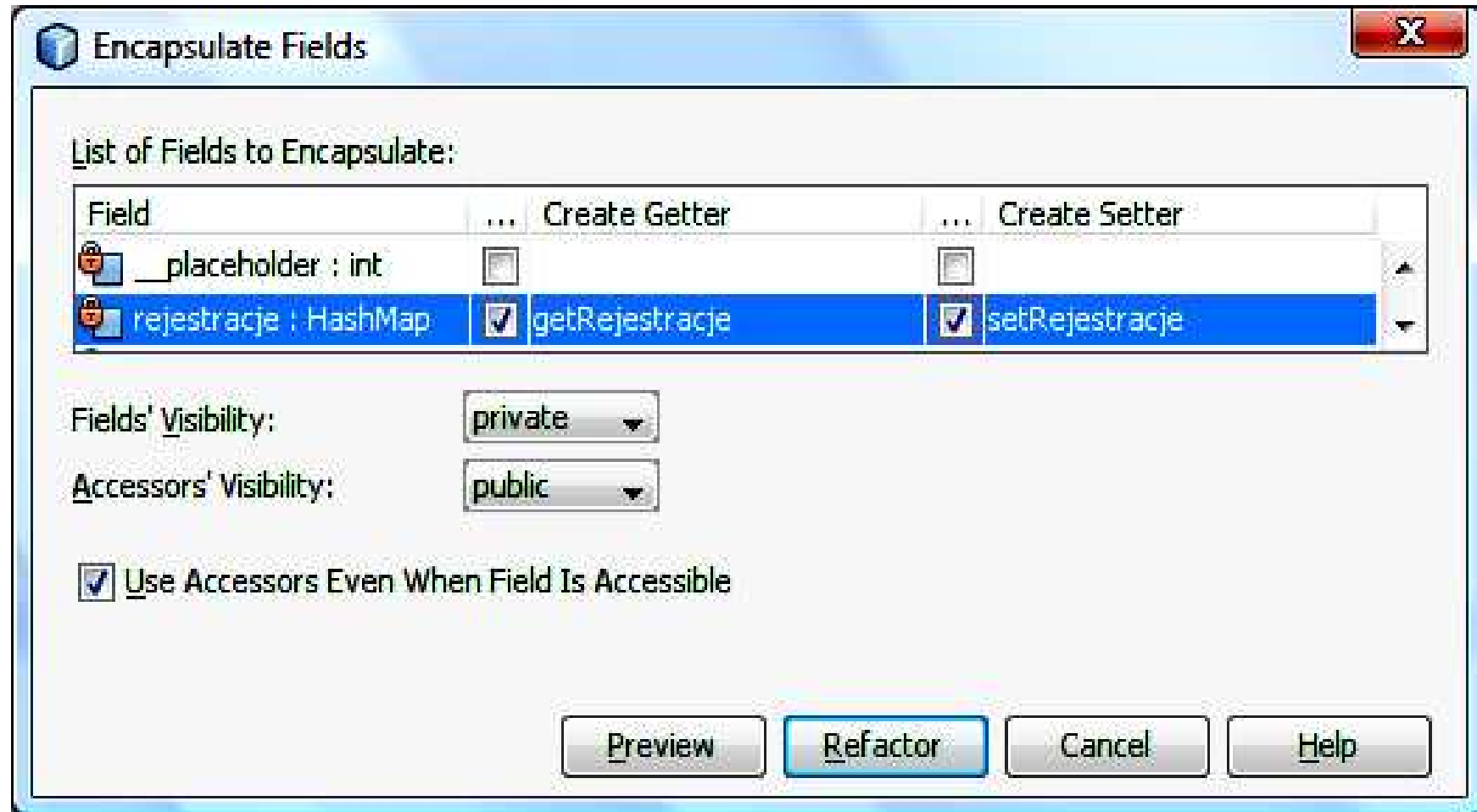


2.1. Dodanie metod typu set i get dla atrybutu **rejestracje** typu **HashMap** do klasy **ApplicationBean1** w celu przechowania danych rejestrowanych klientów podczas każdej sesji aplikacji internetowej – kliknąć prawym klawiszem myszy na ciało klasy w trybie Java, wybrać **Refactor**, następnie **Encapsulate Fields**

The screenshot displays the NetBeans IDE 6.0.1 interface. The main editor window shows the source code for `ApplicationBean1.java`. A context menu is open over the `private HashMap rejestracje` field at line 113. The menu items include 'Navigate', 'Show Javadoc', 'Find Usages', 'Refactor', 'Format', 'Fix Imports', 'Insert Code...', 'Reverse Engineer...', 'Run File', 'Debug File', 'Run Into Method', 'New Watch...', 'Toggle Line Breakpoint', 'Profiling', 'Cut', 'Copy', 'Paste', 'Code Folds', 'Select in', 'EJB Methods', 'Enterprise Resources', and 'Web Service Client Resources'. The 'Refactor' option is highlighted, and a sub-menu is visible with 'Encapsulate Fields...' selected. The IDE also shows the project structure on the left, the Properties window on the right, and the Output window at the bottom.

```
87      * at any later time during the lifetime of the application.</p>
88      */
89      public void destroy() {
90      }
91
92      /**
93       * <p>Return an appropriate character encoding based on the
94       * <code>Locale</code> defined for the current JavaServer
95       * view. If no more suitable encoding can be found, return
96       * "UTF-8" as a general pur
97       *
98       * <p>The default implement
99       * our superclass, <code>Ab
100     */
101     public String getLocaleChar
102         return super.getLocaleC
103     }
104
105     /**
106     * Setter for property ball
107     * @param ballotOptions New
108     */
109
110     /**
111     * Holds value of property
112     */
113     private HashMap rejestracje
114
115     /**
```

2.2. Dodanie metod typu set i get dla atrybutu **rejestracje** typu **HashMap** do klasy **ApplicationBean1** w celu przechowania danych rejestrowanych klientów podczas każdej sesji aplikacji internetowej cd. – zaznaczyć **Create Getter** oraz **Create Setter** i nacisnąć **Refactor**



2.3. Rezultat działań – obiekt `ApplicationBean1` posiada atrybut rejestracje typu `HashMap`.

The screenshot displays the NetBeans IDE 5.5 interface. The main editor window shows the source code of `ApplicationBean1.java`. The code includes a `getLocaleCharacterEncoding()` method, a `private HashMap rejestracje;` field, and `getRejestracje()` and `setRejestracje()` methods. The `setRejestracje()` method is currently selected, with the cursor at line 109, column 4.

```
100  */
101  public String getLocaleCharacterEncoding() {
102      return super.getLocaleCharacterEncoding();
103  }
104
105  /**
106   * Setter for property ballotOptions.
107   * @param ballotOptions New value of property ballotOptions.
108   */
109
110  /**
111   * Holds value of property rejestracje.
112   */
113  private HashMap rejestracje;
114
115  /**
116   * Getter for property rejestracje.
117   * @return Value of property rejestracje.
118   */
119  public HashMap getRejestracje() {
120      return this.rejestracje;
121  }
122
123  /**
124   * Setter for property rejestracje.
125   * @param rejestracje New value of property rejestracje.
126   */
127  public void setRejestracje(HashMap rejestracje) {
128      this.rejestracje = rejestracje;
129  }
130
131  ...
```

The Properties window on the right shows the following details for `ApplicationBean1`:

Properties	
Modifiers	public
Name	aplikacijainter...
Type Parameters	
Extends	AbstractAppl...
Implements	
Javadoc Comment	<p>Aplicat...

The Navigator window shows the project structure, with `ApplicationBean1` selected. The Outline window shows the class hierarchy, with `localeCharacterEncoding` and `rejestracje` listed as attributes.

At the bottom of the IDE, the status bar shows "109:4 INS".

2.4. Uzupełnienie kodu metody `init` w klasie `ApplicationBean1` – metoda `init` w każdym obiekcie uruchamiana jest w momencie tworzenia obiektu

The screenshot displays the NetBeans IDE 5.5 interface. The main editor window shows the `ApplicationBean1.java` file with the following code:

```
48 * managed bean facility to instantiate this bean and store it
49 * application scope.</p>
50 *
51 * <p>You may customize this method to initialize and cache ap
52 * data values (such as the lists of valid options for dropdw
53 * components), or to allocate resources that are required for
54 * lifetime of the application.</p>
55 */
56
57 public void init() {
58     // populate ballot items
59     // initialize counters for ballot choices
60     // Perform initializations inherited from our superclass
61     super.init();
62     // Perform application initialization that must complete
63     // *before* managed components are initialized
64     // TODO - add your own initialization code here
65     rejestracje = new HashMap();
66     Managed Component Initialization
67     // Perform application initialization that must complete
68     // *after* managed components are initialized
69     // TODO - add your own initialization code here
70 }
71
72 /**
73 * <p>This method is called when this bean is removed from
74 * application scope. Typically, this occurs as a result of
75 * the application being shut down by its owning container.</p>
76 *
77 * <p>You may customize this method to clean up resources alloc
78 * during the execution of the <code>init()</code> method, or
79 * at any later time during the lifetime of the application.</p>
```

The `super.init();` line is highlighted in red. The `Managed Component Initialization` line is highlighted in yellow. The `64:36 INS` status bar is visible at the bottom.

The `init - Properties` window on the right shows the following properties:

Properties	
Modifiers	public
Name	init
Type Parameters	
Parameters	
Return Type	void
Exceptions	
Javadoc Comment	<p>This me...

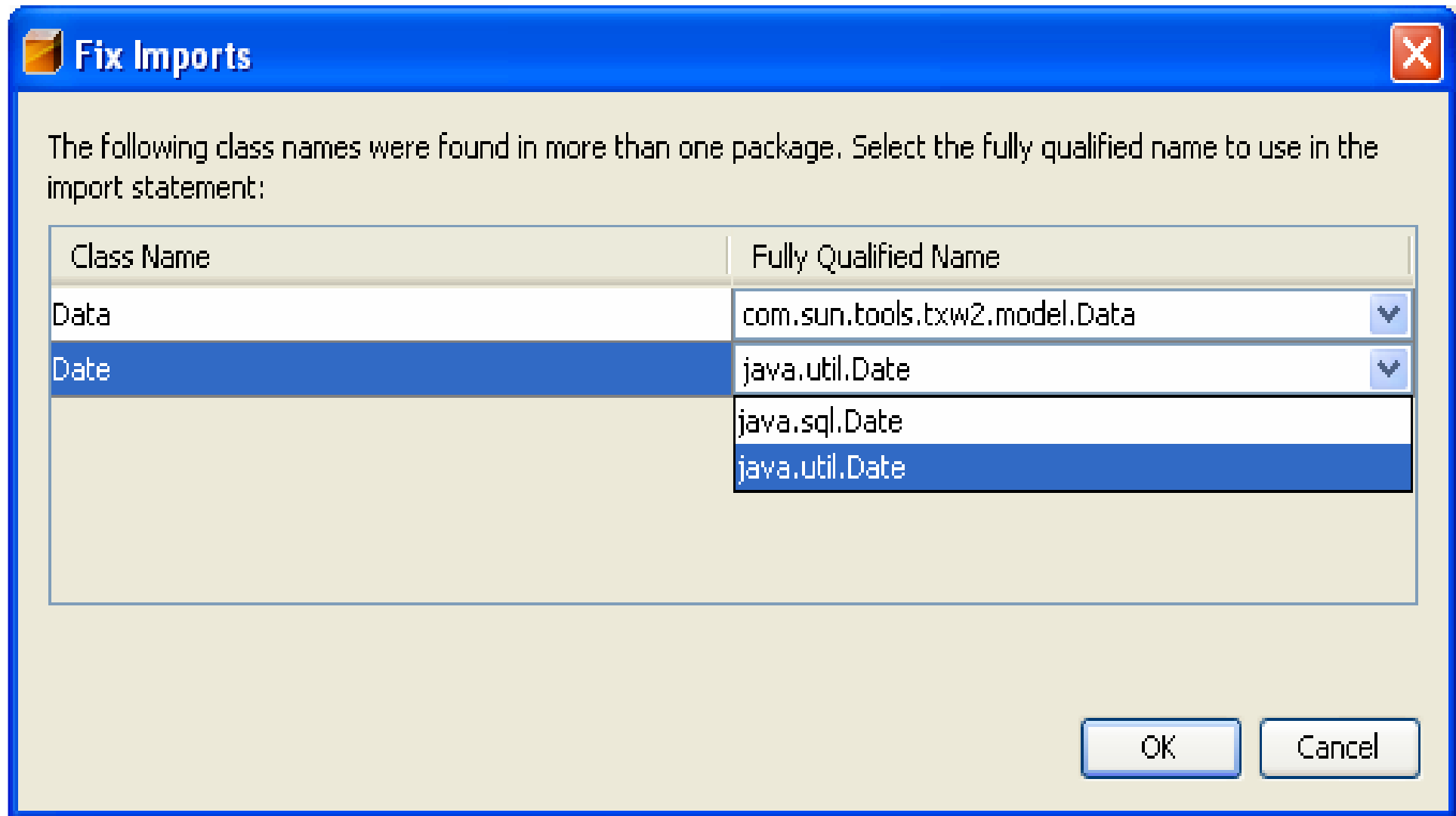
The `Navigator - init` window at the bottom left shows the project structure:

- Page1
 - RequestBean1
 - SessionBean1
 - ApplicationBean1
 - localeCharacterEncoding
 - rejestracje

2.5. Definicja metody zarejestruj w klasie `ApplicationBean1`, która rejestruje klientów za pomocą pary danych wstawianych do kolekcji `rejestracje` typu `HashMap` wstawiając jako klucz datę bieżącą i dane, które stanowią identyfikator klienta

```
/**
 * rejestracja */
public void zarejestruj(String id_klient)
{
    Date data = new Date();
    rejestracje.put(data, id_klient);
}
```


2.6. Import brakujących pakietów - klawisze **CTRL+Shift+I** oraz wybór w formularzu, który pokazuje się w sytuacji, kiedy należy wybrać właściwą klasę



2.7. Rezultat działań – obiekt `ApplicationBean1` posiada metodę `zarejestruj`.

The screenshot displays the NetBeans IDE 5.5 interface for a project named "AplikacjaInternetowa7". The main editor window shows the source code for `ApplicationBean1.java`. The code includes a private `HashMap` for registration, a getter method `getRejestracje()`, and a setter method `setRejestracje()`. A new `zarejestruj()` method is implemented, which creates a `Date` object and adds it to the registration map.

```
113 private HashMap rejestracje;
114
115 /**
116  * Getter for property rejestracje.
117  * @return Value of property rejestracje.
118  */
119 public HashMap getRejestracje() {
120     return this.rejestracje;
121 }
122
123 /**
124  * Setter for property rejestracje.
125  * @param rejestracje New value of property rejestracje.
126  */
127 public void setRejestracje(HashMap rejestracje) {
128     this.rejestracje = rejestracje;
129 }
130
131 /**
132  * rejestracja */
133 public void zarejestruj(String id_klient) {
134     Date data = new Date();
135     rejestracje.put(data, id_klient);
136 }
137
```

The left sidebar shows the project structure, including the `ApplicationBean1` class and its associated properties. The right sidebar shows the properties of the `ApplicationBean1` class, including its name, file size, and classpaths.

Output: HTTP Monitor
C:\Documents and Settings\Kruczkiewicz\rozne\dydaktyka\Komponenty\pkw5\AplikacjaInternetowa7\src\java\aplikacjainternetowa7\ApplicationBean1.java

3. Dodawanie atrybutu rejestracja typu boolean do klasy typu SessionBean1 - ręczne wpisanie kodu do ciała klasy

The screenshot shows the NetBeans IDE interface with the following components:

- Project Explorer:** Shows the project structure for 'AplikacjaInternetowa7', including 'Web Pages', 'resources', and 'Source Packages'. The file 'SessionBean1.java' is selected under 'Source Packages'.
- Code Editor:** Displays the source code for 'SessionBean1.java'. The code includes:

```
public void activate() {  
    }  
  
/**  
 * <p>This method is called when this bean i  
 * session scope. Typically, this occurs as  
 * the session timing out or being terminate  
 *  
 * <p>You may customize this method to clean  
 * during the execution of the <code>init()<  
 * at any later time during the lifetime of  
 */  
public void destroy() {  
    }  
  
/**  
 * <p>Return a reference to the scoped data  
 */  
protected ApplicationBean1 getApplicationBea  
    return (ApplicationBean1) getBean("Applic  
}  
  
/**  
 * Holds value of property rejestracja.  
 */  
private boolean rejestracja;  

```
- Properties Window:** Shows details for 'SessionBe...' including Name, All Files, File Size (4924), Modification Time (2007-0...), and Classpaths.
- Navigator:** Shows the class hierarchy for 'SessionBean1.java', including 'AbstractSessionBea' and 'SessionBean1()'. The 'SessionBean1()' class is selected.
- Output Window:** Shows the status of the application, including 'AplikacjaInternetowa7 (run)', 'Java DB Database Process', and 'GlassFish V2'.

3.1. Dodanie metod typu **set** i **get** dla atrybutu **rejestracja** typu **boolean** do klasy **SessionBean1** w celu umożliwienia rejestracji klientowi podczas każdej sesji tylko raz – kliknąć prawym klawiszem myszy na ciało klasy w trybie Java, wybrać **Refactor**, następnie **Encapsulate Fields**.

3.2. Dodanie metod typu **set** i **get** dla atrybutu **rejestracja** typu **boolean** do klasy **SessionBean1** w celu umożliwienia rejestracji klientowi podczas każdej sesji tylko raz cd. – zaznaczyć **Create Getter** oraz **Create Setter** i nacisnąć **Refactor**

Uwagi:

W momencie wywołania głównej strony aplikacji **Page1** tworzony jest obiekt typu **SessionBean1** i zmienna **rejestracja** przyjmuje wartość **false**. Podczas rejestracji klienta zmienna **rejestracja** przyjmuje wartość **true**, co blokuje ponowne wprowadzanie danych do rejestracji. Dopiero, kiedy obiekt typu **SesionBean1** zostanie zniszczony (po czasie wyznaczonym przez **TimeOut**) i ponownie utworzony przy kolejnym wywołaniu strony internetowej, zmienna **rejestracja** przyjmuje wartość **false**. Klient może teraz zarejestrować się, ponieważ formularz rejestracji jest odblokowany dzięki wartości zmiennej **rejestracja** równej **false**.

3.3. Rezultat działań – obiekt **SessionBean1** posiada atrybut **rejestracja** typu **boolean**

The screenshot displays the NetBeans IDE 5.5 interface. The main editor window shows the source code for `SessionBean1.java`. The code includes a `private boolean rejestracja;` attribute and a `public boolean isRejestracja()` getter method. The `isRejestracja()` method returns `this.rejestracja;`. The `setRejestracja(boolean rejestracja)` setter method is also present, setting `this.rejestracja = rejestracja;`. The `Properties` window on the right shows the `id` property set to `SessionBean1`. The `Navigator` window on the left shows the project structure, with `SessionBean1` selected under `ApplikacjaInternetowa7`. The `Outline` window shows the `rejestracja` attribute under `SessionBean1`.

```
public void destroy() {
    109
    110
    111 /**
    112  * <p>Return a reference to the scoped data bean.</p>
    113  */
    114 protected ApplicationBean1 getApplicationBean1() {
    115     return (ApplicationBean1) getBean("ApplicationBean1");
    116 }
    117
    118 /**
    119  * Holds value of property rejestracja.
    120  */
    121 private boolean rejestracja;
    122
    123 /**
    124  * Getter for property rejestracja.
    125  * @return Value of property rejestracja.
    126  */
    127 public boolean isRejestracja() {
    128     return this.rejestracja;
    129 }
    130
    131 /**
    132  * Setter for property rejestracja.
    133  * @param rejestracja New value of property rejestracja.
    134  */
    135 public void setRejestracja(boolean rejestracja) {
    136     this.rejestracja = rejestracja;
    137 }
    138 }
    139
```


3.4. Uzupełnienie kodu metody `init` w klasie `SessionBean1` – metoda `init` w każdym obiekcie uruchamiana jest w momencie tworzenia obiektu i przypisuje wartość `false` do atrybutu `rejestracja`

The screenshot displays the NetBeans IDE 5.5 interface. The main editor window shows the `SessionBean1.java` file with the following code:

```
39 }
40
41 /**
42  * <p>This method is called when this bean is initially added to
43  * session scope. Typically, this occurs as a result of evaluating
44  * a value binding or method binding expression, which utilizes the
45  * managed bean facility to instantiate this bean and store it in
46  * session scope.</p>
47  *
48  * <p>You may customize this method to initialize and cache data
49  * or resources that are required for the lifetime of a particular
50  * user session.</p>
51  */
52
53 public void init() {
54     // Perform initializations inherited from our superclass
55     super.init();
56     setRejestracja(false);
57     // Perform application initialization that must complete
58     // *before* managed components are initialized
59     // TODO - add your own initialization code here
60
61     Managed Component Initialization
62
63     // Perform application initialization that must complete
64     // *after* managed components are initialized
65     // TODO - add your own initialization code here
66 }
67
68 /**
69  * <p>This method is called when the session containing it is
70  * passivated. Typically, this occurs in a distributed servlet
71  * environment when the session is about to be transferred to a different
72  * container instance, after which the <code>activate()</code>
```

The `setRejestracja(false);` line is highlighted in yellow. The `Managed Component Initialization` section is also highlighted.

The `Navigator - init` panel on the left shows the project structure, with `SessionBean1` selected. The `init - Properties` panel on the right shows the properties of the `init` method, including its name, return type, and Javadoc comment.

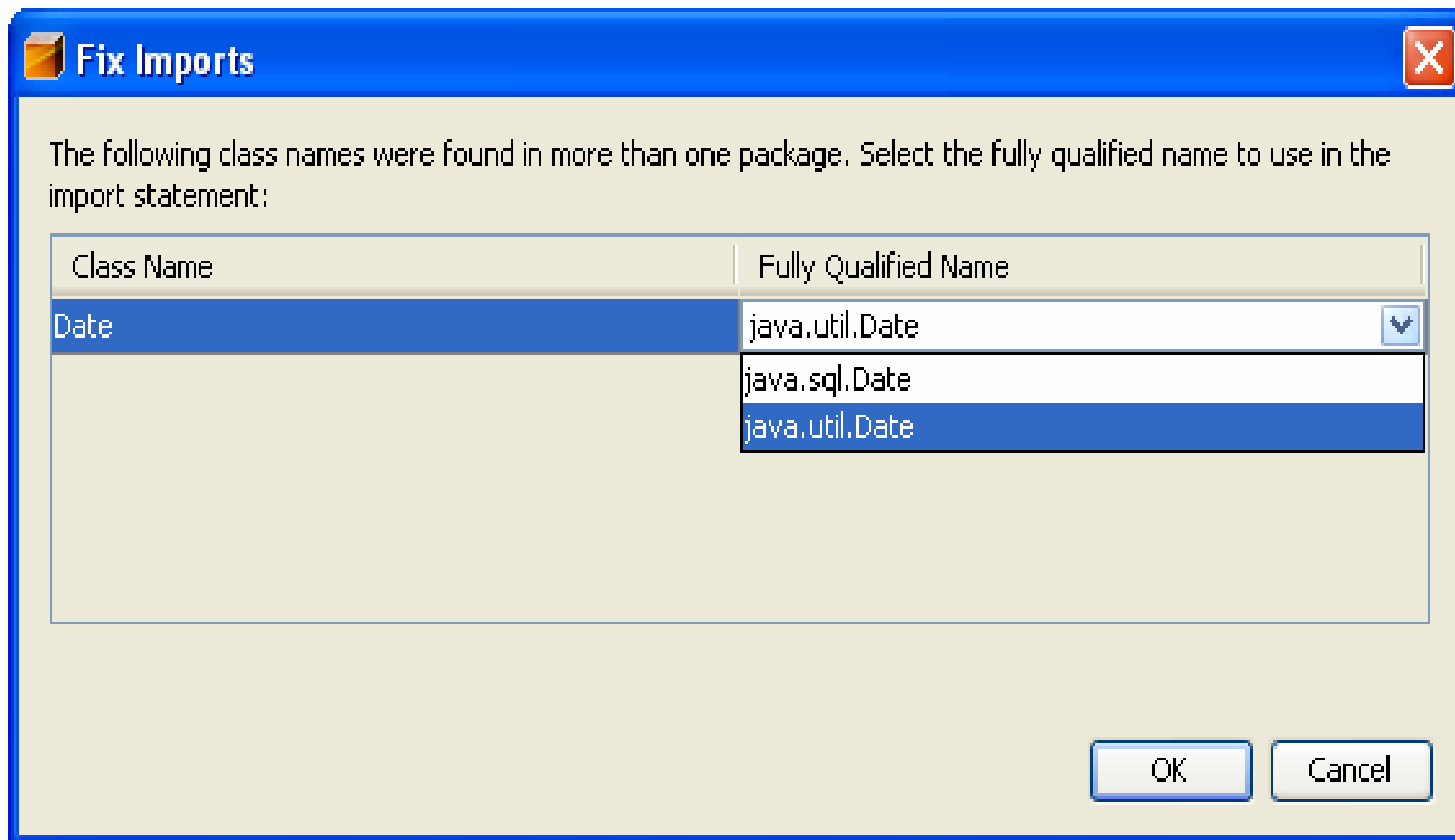
4. Dodawanie atrybutu **czas** typu **Date** do klasy typu **RequestBean1** – ręczne wstawienie kodu

The screenshot shows the NetBeans IDE 6.0.1 interface. The main editor window displays the `RequestBean1.java` file. The code is as follows:

```
82      * <p>You may customize this method to clean
83      * during the execution of the <code>init()<
84      * at any later time during the lifetime of
85      */
86
87      public void destroy() {
88      }
89
90      /**
91      * <p>Return a reference to the scoped data
92      */
93      protected SessionBean1 getSessionBean1() {
94          return (SessionBean1) getBean("SessionBea
95      }
96
97      /**
98      * <p>Return a reference to the scoped data
99      */
100     protected ApplicationBean1 getApplicationBea
101         return (ApplicationBean1) getBean("Applic
102     }
103
104     /**
105     * Holds value of property czas.
106     */
107     private Date czas;
```

The `private Date czas;` line is highlighted in green. The left sidebar shows the project structure with `RequestBean1.java` selected. The right sidebar shows the Properties window for `RequestBean1.java`. The bottom status bar shows the Output window with the application running.

4.1. Import brakujących pakietów - klawisze **CTRL+Shift+I** oraz wybór w formularzu, który pokazuje się w sytuacji, kiedy należy wybrać właściwą klasę



4.2. Dodanie atrybutu **czas** typu **Date** do klasy **RequestBean1** w celu przechowania czasu rejestracji klienta pamiętany jedynie podczas przesłania jego wartości oraz danych rejestracji ze strony głównej **Page1** do strony **Rejestracje**. Wartość atrybutu **czas** zawiera odczytany czas systemowy – kliknąć prawym klawiszem myszy na ciało klasy w trybie Java, wybrać **Refactor**, następnie **Encapsulate Fields**.

4.3. Dodanie metod typu **set** i **get** dla atrybutu **czas** typu **Date** do klasy **RequestBean1** w celu przechowania czasu rejestracji klienta pamiętany jedynie podczas przesłania jego wartości oraz danych rejestracji ze strony głównej **Page1** do strony **Rejestracje** cd. – zaznaczyć **Create Getter** oraz **Create Setter** i nacisnąć **Refactor**

Uwagi

W momencie wywołania strony **Page1** tworzony jest obiekt typu **RequestBean1**. Odczytany wtedy czas systemowy zapisany w atrybucie **czas** typu **Date** jest pamiętany do momentu, kiedy kończy się obsługa wysłania wartości tego czasu na stronę **Rejestracje**.

4.4. Rezultat działań – obiekt RequestBean1 posiada atrybut czas typu Date

The screenshot displays the NetBeans IDE 5.5 interface. The main editor window shows the source code for `RequestBean1.java`. The following code is visible:

```
92     return (sessionBean1.getBean( sessionBean1 ));
93 }
94
95 /**
96  * <p>Return a reference to the scoped data bean.</p>
97  */
98 protected ApplicationBean1 getApplicationBean1() {
99     return (ApplicationBean1) getBean("ApplicationBean1");
100 }
101
102 /**
103  * Holds value of property czas.
104  */
105 private Date czas;
106
107 /**
108  * Getter for property czas.
109  * @return Value of property czas.
110  */
111 public Date getCzas() {
112     return this.czas;
113 }
114
115 /**
116  * Setter for property czas.
117  * @param czas New value of property czas.
118  */
119 public void setCzas(Date czas) {
120     this.czas = czas;
121 }
122 }
123
```

The line `108` is highlighted in yellow. The status bar at the bottom shows `108:32 INS`.

The **RequestBean1 - Properties** window on the right shows the following details:

Property	Value
Modifiers	public
Name	aplikacijainter...
Type Parameters	
Extends	AbstractReq...
Implements	
Javadoc Comment	<p>Reques...

The **RequestBean1** class declaration is also visible in the bottom right:

```
public class RequestBean1 extends com.sun.rave.web.ui.appbase.AbstractRequestBean
```

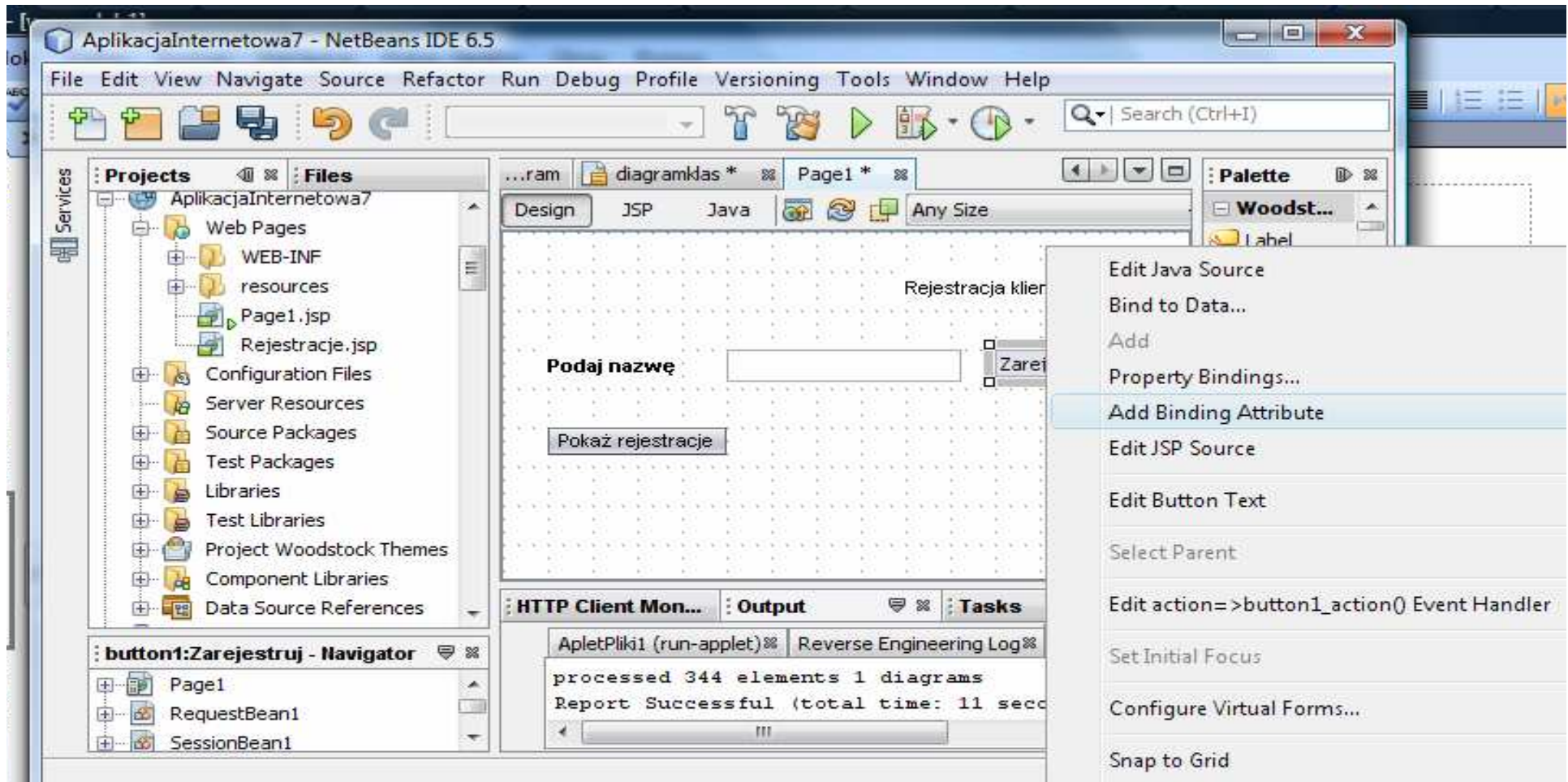
The **Navigator** window on the left shows the project structure, with `RequestBean1` and its attribute `czas` highlighted.

At the bottom of the IDE, there are tabs for **Output** and **HTTP Monitor**. A message at the bottom left states: "Imports were fixed."

5. Wstawienie komponentów w trybie Design do strony Page1 typu: StaticText, Label, TextField oraz dwa komponenty typu Button

The screenshot displays the NetBeans IDE 6.5 interface. The main workspace is in Design mode, showing a web page titled "Rejestracja klientów". The page layout includes a label "Podaj nazwę" followed by a text input field and a "Zarejestruj" button. Below this is a "Pokaż rejestracje" button. The left sidebar shows the project structure for "AplikacjaInternetowa7", with "Page1" selected in the Navigator. The Palette on the right lists components like Label, Static Text, Text Field, Text Area, Button, and Hyperlink. The "Page1 - Prop..." window shows properties for the selected page, including "id" (Page1), "Page Layout" (Grid...), and "Response Encoding" (UTF-8).

5.1. Wstawianie do pliku typu Java strony JSP obiektów reprezentujących komponenty wizualne interfejsu graficznego użytkownika



Należy zaznaczyć kolejno **każdy** komponent, następnie kliknąć na niego prawym klawiszem myszy i z wyskakującego menu kliknąć na pozycję **Add Binding Attribute**. W pliku typu Java pojawi się kod obiekt typu komponent oraz 2 metody typu **set** i **get**.

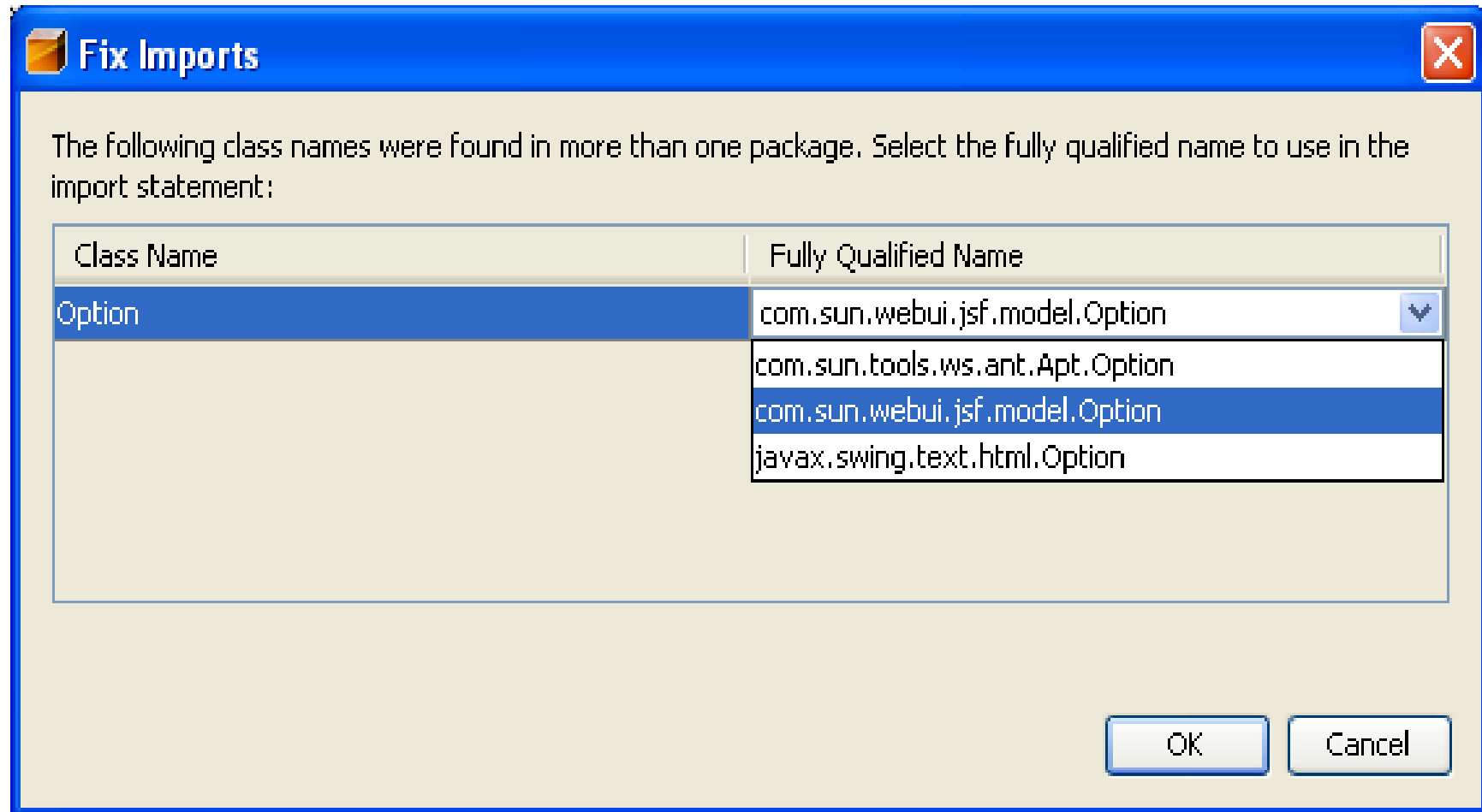
6. Przygotowanie do wstawienia atrybutu **danerejestracji** jako tablicy elementów **Option** do klasy typu **ApplicationBean1** – ręczne wstawienie atrybutu

The screenshot shows the NetBeans IDE 6.0.1 interface. The main editor displays the source code of `ApplicationBean1.java`. The code is as follows:

```
115  /**
116   * Getter for property rejestracje.
117   * @return Value of property rejestracje.
118   */
119  public HashMap getRejestracje() {
120      return this.rejestracje;
121  }
122
123  /**
124   * Setter for property rejestracje.
125   * @param rejestracje New value of property
126   */
127  public void setRejestracje(HashMap rejestrac
128      this.rejestracje = rejestracje;
129  }
130
131  /**
132   * rejestracja */
133  public void zarejestruj(String id_klient) {
134      Date data = new Date();
135      rejestracje.put(data, id_klient); }
136
137  /**
138   * Holds value of property danerejestracji.
139   */
140  private Option[] danerejestracji;
```

The IDE interface includes a Project Explorer on the left showing the project structure, a Properties window on the right, and an Output window at the bottom. The status bar at the bottom indicates the current cursor position at line 103, column 6.

6.1. Import brakujących pakietów - klawisze **CTRL+Shift+I** oraz wybór w formularzu, który pokazuje się w sytuacji, kiedy należy wybrać właściwą klasę



6.2. Wstawienie atrybutu **danerejestracji** jako tablicy elementów **Option** do klasy typu **ApplicationBean1** w celu umożliwienia wyświetlania danych rejestracji przechowywanych w obiekcie **ApplicationBean1** w atrybucie **rejestracje** typu **HashMap** – kliknąć prawym klawiszem myszy na ciało klasy w trybie Java, wybrać **Refactor**, następnie **Encapsulate Fields**.

6.3. Dodanie metod typu **set** i **get** dla atrybutu **danerejestracji**, jako tablicy elementów **Option**, do klasy typu **ApplicationBean1** w celu umożliwienia wyświetlania danych rejestracji przechowywanych w obiekcie **ApplicationBean1** w atrybucie **rejestracje** typu **HashMap** cd. – zaznaczyć **Create Getter** oraz **Create Setter** i nacisnąć **Refactor**

6.4. Rezultat po wstawieniu atrybutu `danerejestracji` jako tablicy elementów `Option` do klasy typu `ApplicationBean1`

The screenshot displays the NetBeans IDE 5.5 interface. The main editor window shows the source code for `ApplicationBean1.java`. The code includes the following methods and property:

```
125 * @param rejestracje New value of property rejestracje.
126 */
127 public void setRejestracje(HashMap rejestracje) {
128     this.rejestracje = rejestracje;
129 }
130 /**
131  * rejestracja */
132 public void zarejestruj(String id_klient) {
133     Date data = new Date();
134     rejestracje.put(data, id_klient); }
135
136 /**
137  * Holds value of property danerejestracji.
138  */
139 private Option[] danerejestracji;
140
141 /**
142  * Getter for property danerejestracji.
143  * @return Value of property danerejestracji.
144  */
145 public Option[] getDanerejestracji() {
146     return this.danerejestracji;
147 }
148
149 /**
150  * Setter for property danerejestracji.
151  * @param danerejestracji New value of property danerejestr
152  */
153 public void setDanerejestracji(Option[] danerejestracji) {
154     this.danerejestracji = danerejestracji;
155 }
156
```

The `danerejestracji` property is listed in the `Properties` window on the right, with the value `ApplicationBean1`. The `Navigator` window on the left shows the project structure, including the `ApplicationBean1` class and its associated properties: `localeCharacterEncoding`, `rejestracje`, and `danerejestracji`.

6.5. Uzupełnienie metody `init` w klasie typu `ApplicationBean1` – inicjowanie tablicy danerejestracji

The screenshot displays the NetBeans IDE 5.5 interface. The main editor window shows the `ApplicationBean1.java` file with the following code:

```
54      * lifetime of the application.</p>
55      */
56      public void init() {
57          // populate ballot items
58          // initialize counters for ballot choices
59          // Perform initializations inherited from our superclass
60          super.init();
61          // Perform application initialization that must complete
62          // *before* managed components are initialized
63          // TODO - add your own initialization code here
64          rejestracje = new HashMap();
65          danerejestracji = new Option [] {new Option("0", "Brak danych")};
66          Managed Component Initialization
67          // Perform application initialization that must complete
68          // *after* managed components are initialized
69          // TODO - add your own initialization code here
70      }
71      /**
72      * <p>This method is called when this bean is removed from
73      * application scope. Typically, this occurs as a result of
74      * the application being shut down by its owning container.</p>
75      *
76      * <p>You may customize this method to clean up resources allocated
77      * during the execution of the <code>init()</code> method, or
78      * at any later time during the lifetime of the application.</p>
79      */
80      public void destroy() {
81      }
```

The `Managed Component Initialization` comment is highlighted in yellow. The `rejestracje = new HashMap();` line is highlighted in red. The `danerejestracji = new Option [] {new Option("0", "Brak danych")};` line is highlighted in light blue.

The left sidebar shows the `Projects` and `Navigator` views. The `Navigator` view shows the `Rejestracje` project structure, including `page1`, `html1`, `head1`, `body1`, `form1`, `staticText`, `listbox1`, `staticText`, and `button1:p`. The `ApplicationBean1` class is highlighted in the `Navigator` view.

The right sidebar shows the `Dynamic Help` panel for `ApplicationBean1`, displaying the `id` attribute and the `ApplicationBean1 (application)` class.

7. Dodanie nowej strony `Rejestracje.jsp` do prezentowania danych aplikacji oraz czasu życia atrybutów obiektów aplikacji

The screenshot shows the NetBeans IDE 6.0.1 interface. The 'New' menu is open, and 'Visual Web JSF Page...' is selected. The IDE displays a JSP page with the following code:

```
rejestracje.  
arty rejestracje.  
  
rejestracje() {  
    rejestracje;  
  
rejestracje.  
w value of property rejestracje.  
  
rejestracje(HashMap rejestracje) {  
    rejestracje;  
  
String id_klient) {  
    Date data = new Date();  
    rejestracje.put(data, id_klient); }  
  
/**  
 * Holds value of property danerejestracji.  
 */  
private Option[] danerejestracji;  
  
/**  
 * Getter for property danerejestracji.  
 * @return Value of property danerejestracji.  
 */
```

The properties window on the right shows '<No Properties>'. The bottom status bar shows 'Output' and 'HTTP Monitor'.

7.1. Dodanie nowej strony **Rejestracje.jsp** do prezentowania danych aplikacji oraz wywołania strony

New Page

Steps

1. Choose File Type
- 2. Name and Location**

Name and Location

File Name: Rejestracje

Project: AplikacjaInternetowa7

Folder: web

Created File: jewicz\rozne\dydaktyka\Komponenty\pkw5\AplikacjaInternetowa7\web\Rejestracje.jsp

< Back Next > Finish Cancel Help

8.1. Tworzenie połączenia komponentu typu List Box z obiektem danerejestracji z obiektu typu ApplicationBean1 – wywołanie opcji Bind to Data dla komponentu typu ListBox z wyskakującego menu

The screenshot displays the NetBeans IDE 5.5 interface for a web application project named 'AplikacjaInternetowa7'. The main workspace shows a JSP page with a table containing registration data. A context menu is open over the 'listbox1' component, with the 'Bind to Data...' option highlighted. The Properties window on the right shows the configuration for 'listbox1', including its width and height. The Navigator window on the left shows the project structure, including the 'Rejestracje' page and the 'ApplicationBean1' object.

NetBeans IDE 5.5 - AplikacjaInternetowa7

File Edit View Navigate Source Refactor Build Run CVS Tools Window Help

Projects: AplikacjaInternetowa6_5, AplikacjaInternetowa6_6, AplikacjaInternetowa7

Navigator: Rejestracje, page1, html1, head1, body1, form1, staticText1:Czas rejestracji, listbox1, staticText2:Dane rejestracji

Outline: listbox1DefaultOptions, RequestBean1, czas, SessionBean1, rejestracja, ApplicationBean1, localeCharacterEncoding, rejestracje, danerejestracji

Design JSP Java

Table content:

Czas rejestracji
Dane rejestracji
Item 1
Item 2
Item 3

Palette: Basic, Static Text, Text Area, Button, Image Hyperlink, Checkbox Group, Radio Button Group, Text Field, Label, Hyperlink, Drop Down List, Checkbox, Radio Button, Image, Table Column, Password Field, Calendar, Tree

Properties: listbox1, width, height, font size (Medium (2)), etc.

Context Menu:

- Edit Java Source
- Edit JSP Source
- Edit Label
- Select Parent
- Edit Event Handler
- Set Initial Focus
- Auto-Submit on Change
- Bind to Data...**
- Configure Default Options...
- Add
- Property Bindings...
- Configure Virtual Forms...
- Snap to Grid
- Bring to Front
- Send to Back
- Customize
- Cut (Ctrl+X)
- Copy (Ctrl+C)
- Paste (Ctrl+V)
- Delete
- Preview in Browser...
- Refresh
- Target Browser Window
- Page Navigation

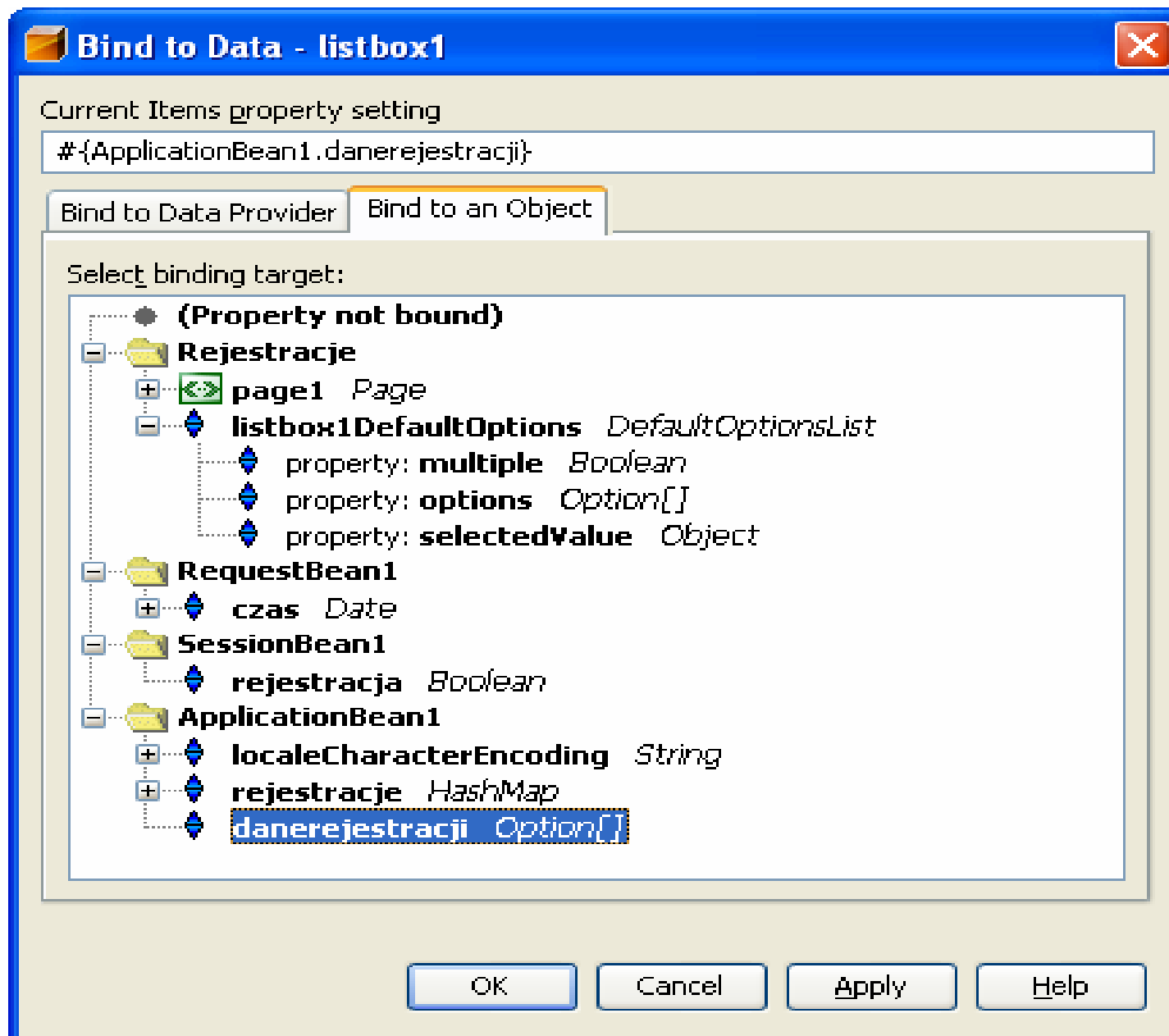
Output: This code creates an HTML table that contains the text property for the first Static Text component. component displays the date and time the vote was cast.

19. Right-click in the source and choose Fix Imports from the context menu.

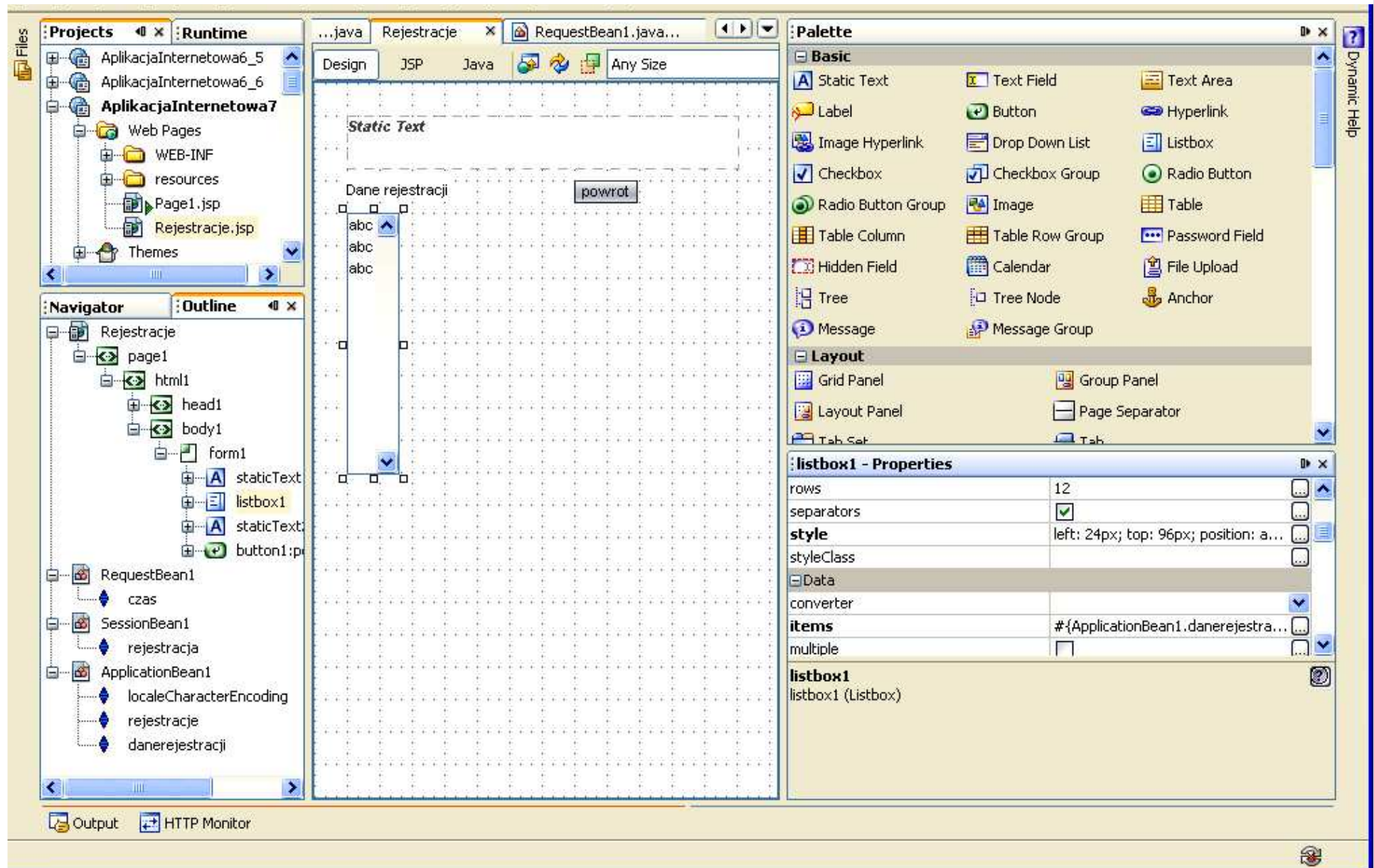
20. Select java.util.Date for from the Fully Qualified Name list, and com.sun.webui.jsf.model.Option for Class Name.

Start | NetBeans IDE 5.5 - A... | Microsoft PowerPoint ... | Windows Commander ... | NetBeans Visual Web ... | PL | 08:32

8.2. Połączenie komponentu typu `ListBox` z obiektem danerejestracji z obiektu typu `ApplicationBean1`



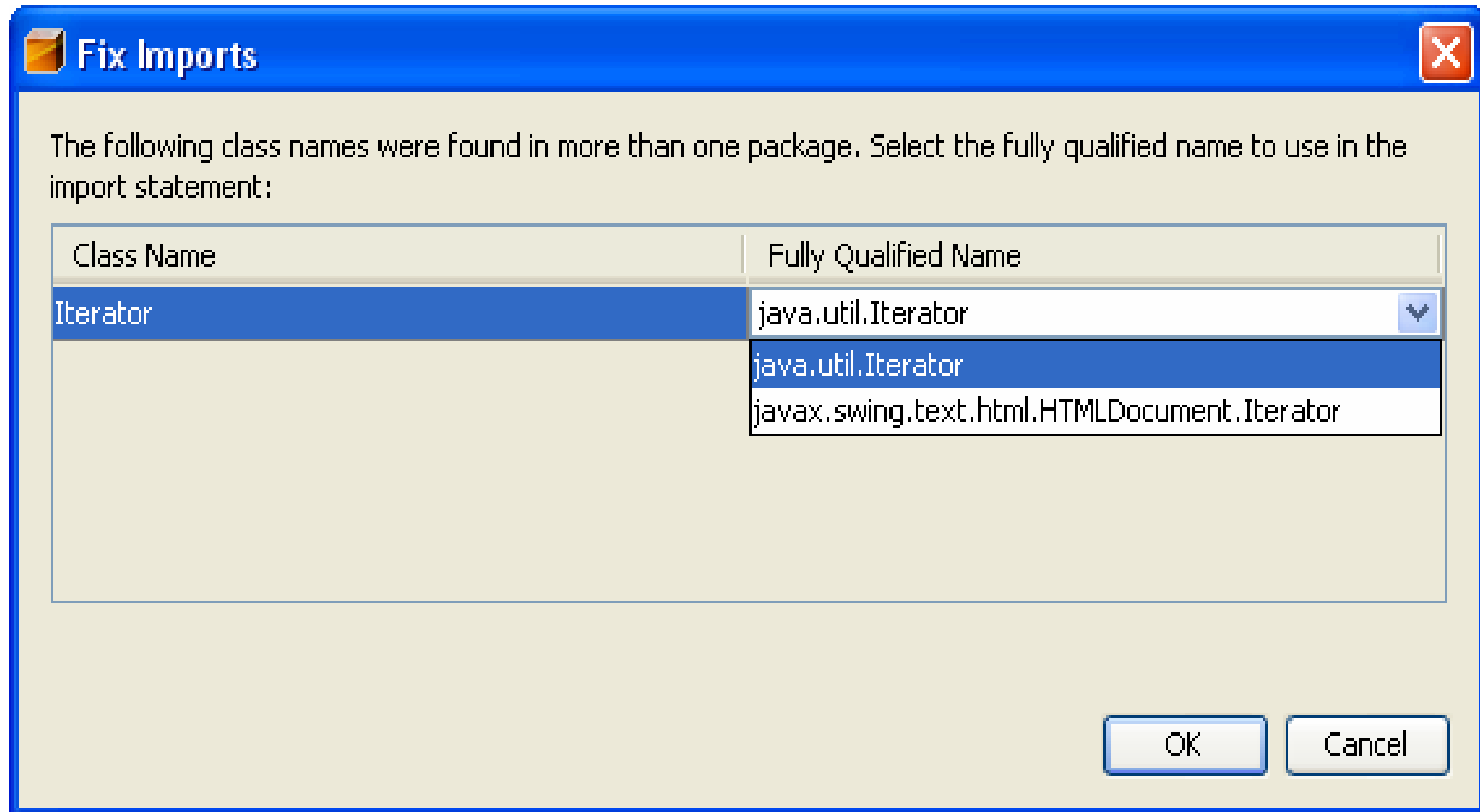
8.3. Wstawienie przycisku powrot typu Button do strony Rejestracje. Powinien on mieć ustawioną opcję Add Binding Attribute (patrz punkt 5.1)



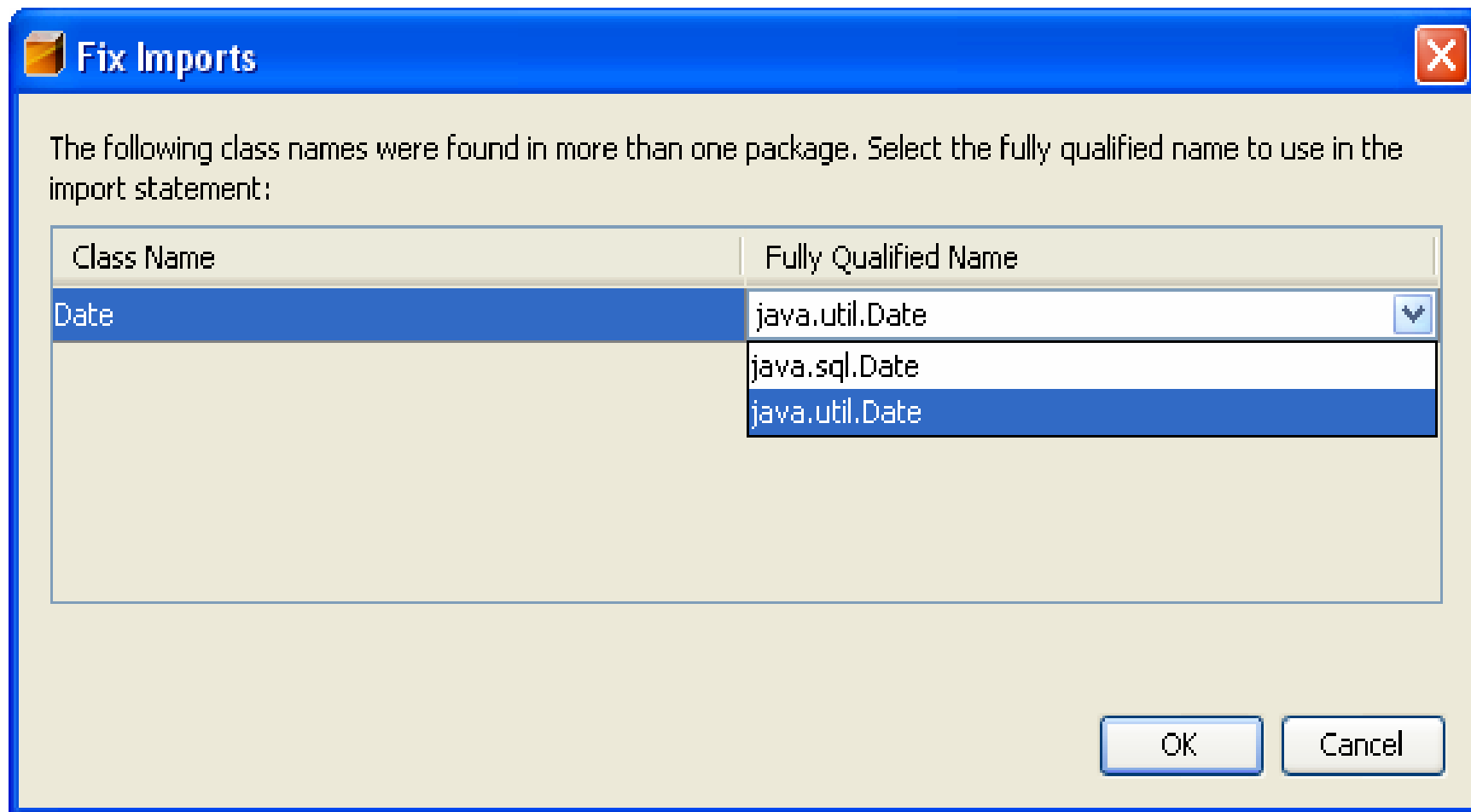
8.4. Wstawienie metody `zawartosc_tablicy_danerejestracji()` do klasy `ApplicationBean1` przygotowującej tablicę `danerejestracji` do „bindowania” z komponentem typu `ListBox` ze strony `Rejestracje`

```
public void zawartosc_tablicy_danerejestracji()
{
    int ile = rejestracje.size();
    if (ile >0) {
        Option klienci[] = new Option[ile];
        Set zbior = rejestracje.entrySet();
        Iterator iterator = zbior.iterator();
        int i=0;
        while(iterator.hasNext())
            klienci[i++] =
                new Option(Integer.toString(i),iterator.next().toString());
        setDanerejestracji(klienci);
    }
}
```

8.5. Import brakujących pakietów - klawisze **CTRL+Shift+I** oraz wybór w formularzu, który pokazuje się w sytuacji, kiedy należy wybrać właściwą klasę



8.6. Import brakujących pakietów - klawisze **CTRL+Shift+I** oraz wybór w formularzu, który pokazuje się w sytuacji, kiedy należy wybrać właściwą klasę



AplikacjaInternetowa7 - NetBeans IDE 6.5

File Edit View Navigate Source Refactor Run Debug Profile Versioning Tools Window Help

Search (Ctrl+I)

Projects Files

AplikacjaInternetowa7

- Web Pages
 - WEB-INF
 - resources
 - Page1.jsp
 - Rejestracje.jsp
- Configuration Files
- Server Resources
- Source Packages
 - aplikacjainternetowa7
 - ApplicationBean1.java
 - Bundle.properties
 - Page1.java
 - Rejestracje.java
 - RequestBean1.java
 - SessionBean1.java
- Test Packages
- Libraries
- Test Libraries
- Project Woodstock Themes
- Component Libraries
- Data Source References

Services

apletplik2.java aplet.policy ApplicationBean1.java * Rejestracje...

```
142 private Option[] danerejestracji;
143 /**...*/
147 public Option[] getDanerejestracji() {
148     return this.danerejestracji;
149 }
150 /**...*/
154 public void setDanerejestracji(Option[] danerejestracji) {
155     this.danerejestracji = danerejestracji;
156 }
157 public void zawartosc_tablicy_danerejestracji()
158 {
159     int ile=rejestracje.size();
160     if (ile >0) {
161         Option klienci[]=new Option[ile];
162         Set zbior=rejestracje.entrySet();
163         Iterator iterator=zbior.iterator();
164         int i=0;
165         while(iterator.hasNext())
166             klienci[i++]=
167                 new Option(Integer.toString(i),iterator.next().toString());
168         setDanerejestracji(klienci);
169     }
170 }
```

150:1 INS

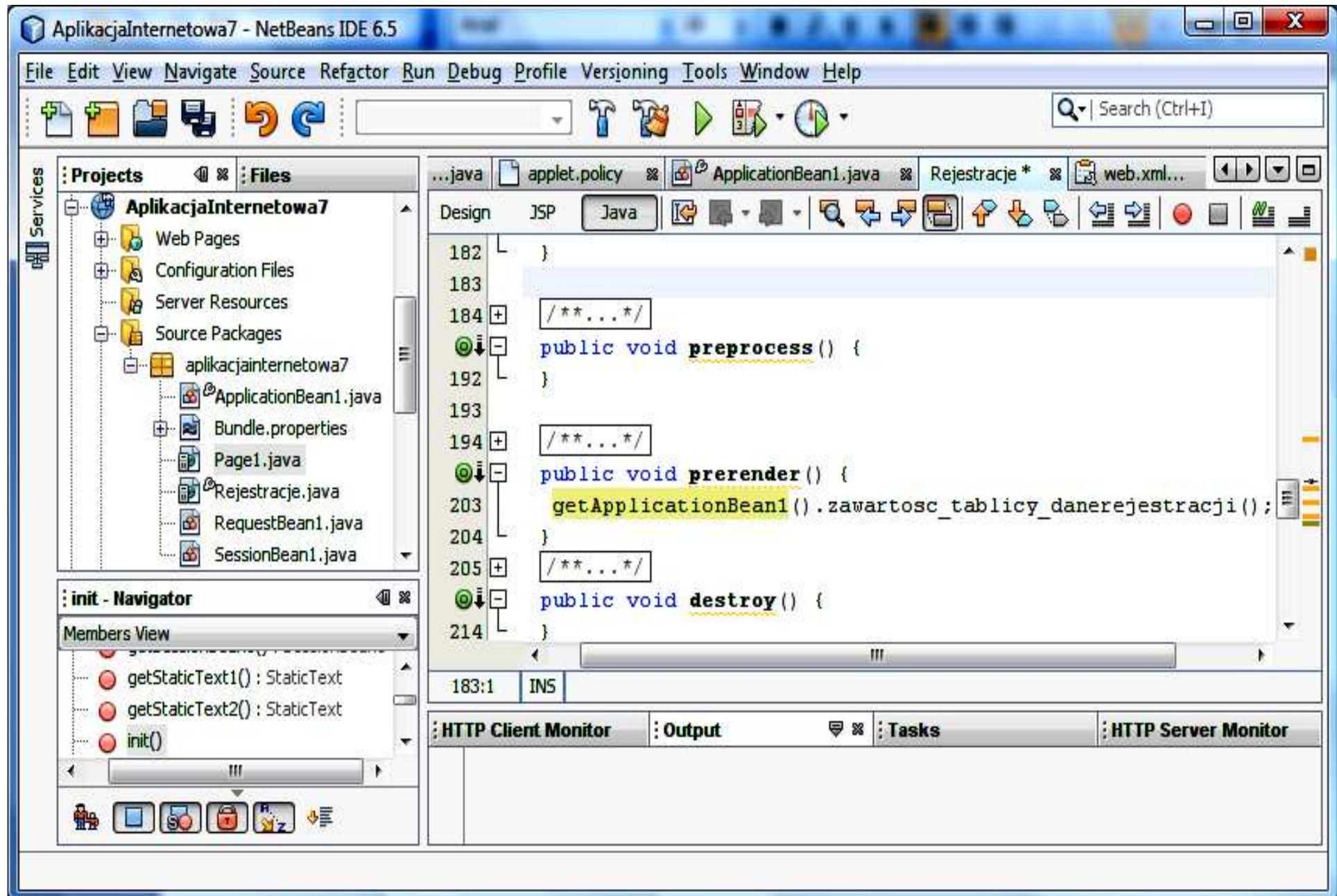
HTTP Client Monitor Output Tasks HTTP Server Monitor

Java DB Database Process GlassFish V2 AplikacjaInternetowa7 (run)

8.7. Uzupelnienie kodu metody istniejacej metody **prerender** dla strony **Rejestracje (plik Rejestracja.java)** – metoda wykonywana podczas wywoływania strony za pomocą przycisków **Zarejestruj** oraz **Pokaż rejestracje strony głównej Page1**

```
public void prerender()  
{  
    getApplicationBean1().zawartosc_tablicy_danerejestracji();  
}
```

8.8. Rezultat działań – strona **Rejestracje** posiada metodę **prerender** do prezentacji danych z obiektów typu **ApplicationBean1**



8.9. Tworzenie połączenia komponentu `staticText1` typu `Static Text` ze strony `Rejestracje.jsp` z atrybutem `czas` z obiektu typu `RequestBean1` – wywołanie opcji `Bind to Data` dla komponentu typu `Static Text` z wyskakującego menu

The screenshot displays the NetBeans IDE 6.0.1 interface. The main workspace shows a JSP page in Design mode with a `Static Text` component. A context menu is open over the component, with the `Bind to Data...` option selected. The `Properties` window on the right shows the `StaticText1` component's properties, including the `dataConverter` property.

Projects: AplikacjaInternetowa7

- Web Pages
 - WEB-INF
 - resources
 - Page1.jsp
 - Rejestracje.jsp
 - Configuration Files
 - Server Resources
 - Source Packages
 - Test Packages
 - Libraries
 - Test Libraries
 - Themes
 - Component Libraries
 - Data Source References
- JavaApplication5
- JavaLibrary1

staticText1 - Navigator

- Rejestracje
 - page1
 - html1
 - head1
 - body1
 - form1
 - staticText1
 - listbox1
 - staticText2:Dane rejestracji
 - button1:powrot
- RequestBean1
- SessionBean1
- ApplicationBean1

Static Text Properties

Property	Value
name	staticText1
appearance	
height	height: 22px; left: 24px; top: ...
width	
z-index	
className	
id	
data	
converter	
escape	<input checked="" type="checkbox"/>
behavior	
tip	

Output

HTTP Monitor

8.10. Połączenie komponentu typu **Static Text** z obiektem **czas** z obiektu typu **RequestBean1** – czas będzie wyświetlał się tylko wtedy, gdy naciska się klawisz **Zarejestruj** na stronie **Page1.jsp** pozwalający przejść do strony **Rejestracje.jsp**. Obsługa zdarzenia tego przycisku (slajd 9.2 – metoda **button1_action**) tworzy nowy obiekt **czas** typu **Date** w obiekcie **RequestBean1**, przechowywany jedynie do ponownego wywołania tej strony, kiedy powstaje nowy obiekt typu **RequestBean1**. Po naciśnięciu przycisku **Pokaż rejestracje** po przejściu na stronę **Rejestracja.jsp** nie wyświetlił się czas, gdyż w nowym obiekcie **RequestBean1** nie utworzono obiektu **czas** podczas obsługi zdarzenia związanego z naciśnięciem tego przycisku (slajd 9.2 – metoda **button2_action**).

8.11. Efekt bindowania pola typu StaticText

The screenshot displays the NetBeans IDE 6.0.1 interface for a project named "AplikacjaInternetowa7". The main workspace shows a JSP page in Design view. The page contains a date "Sat Mar 29 22:16:44 CET 2008", a label "Dane rejestracji", a listbox with three items "abc", and a button labeled "powrot".

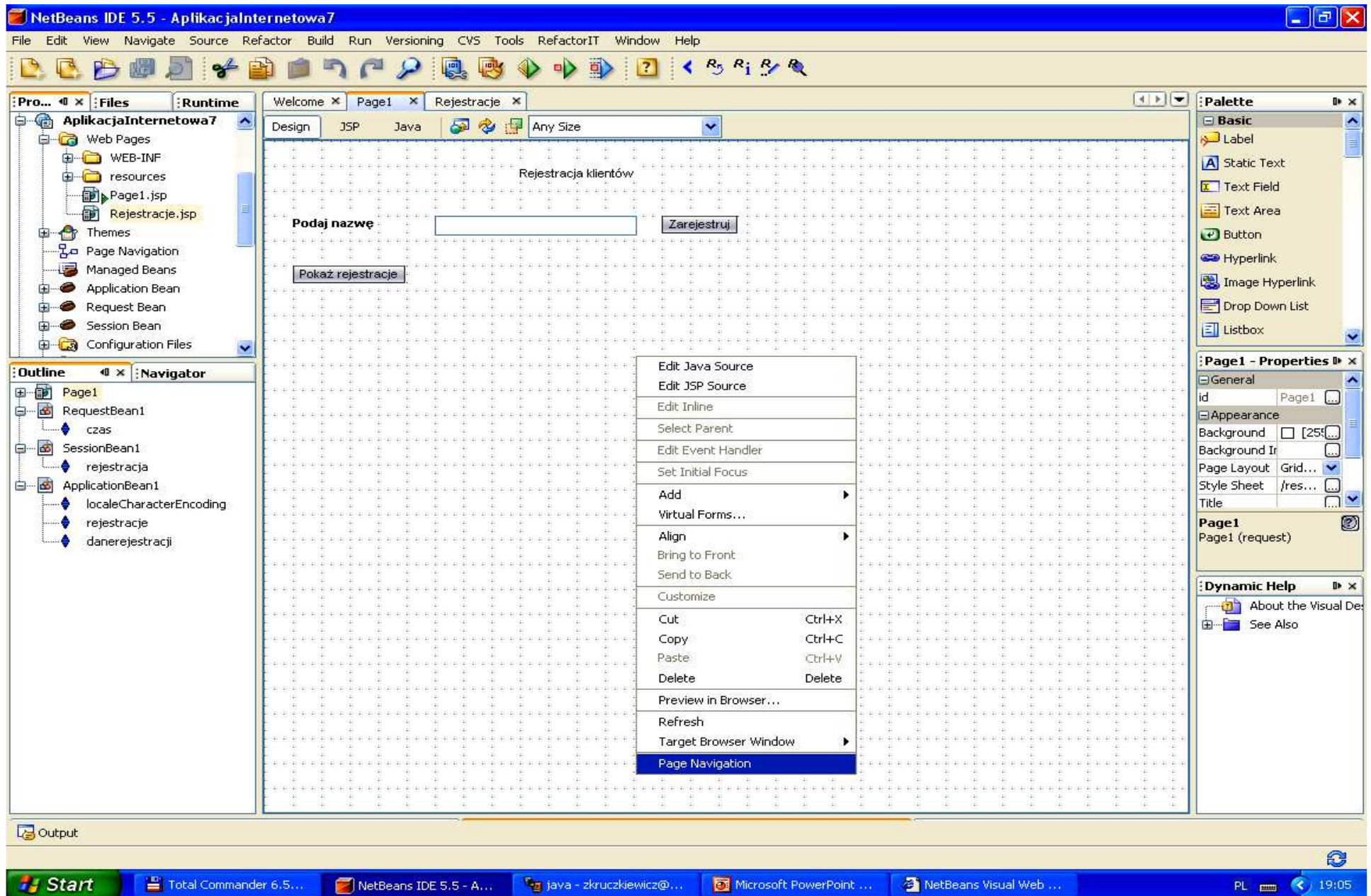
The left sidebar shows the project structure under "AplikacjaInternetowa7", including "Web Pages", "WEB-INF", "resources", "Page1.jsp", and "Rejestracje.jsp". Below it is the "staticText1 - Navigator" showing the component hierarchy: "Rejestracje" > "page1" > "html1" > "body1" > "form1" > "staticText1", "listbox1", "staticText2:Dane rejestracji", and "button1:powrot".

The right sidebar contains the "Palette" with various UI components like Label, Text Field, Button, etc., and the "staticText1 - Properties" window. The Properties window shows the following details for the selected "staticText1" component:

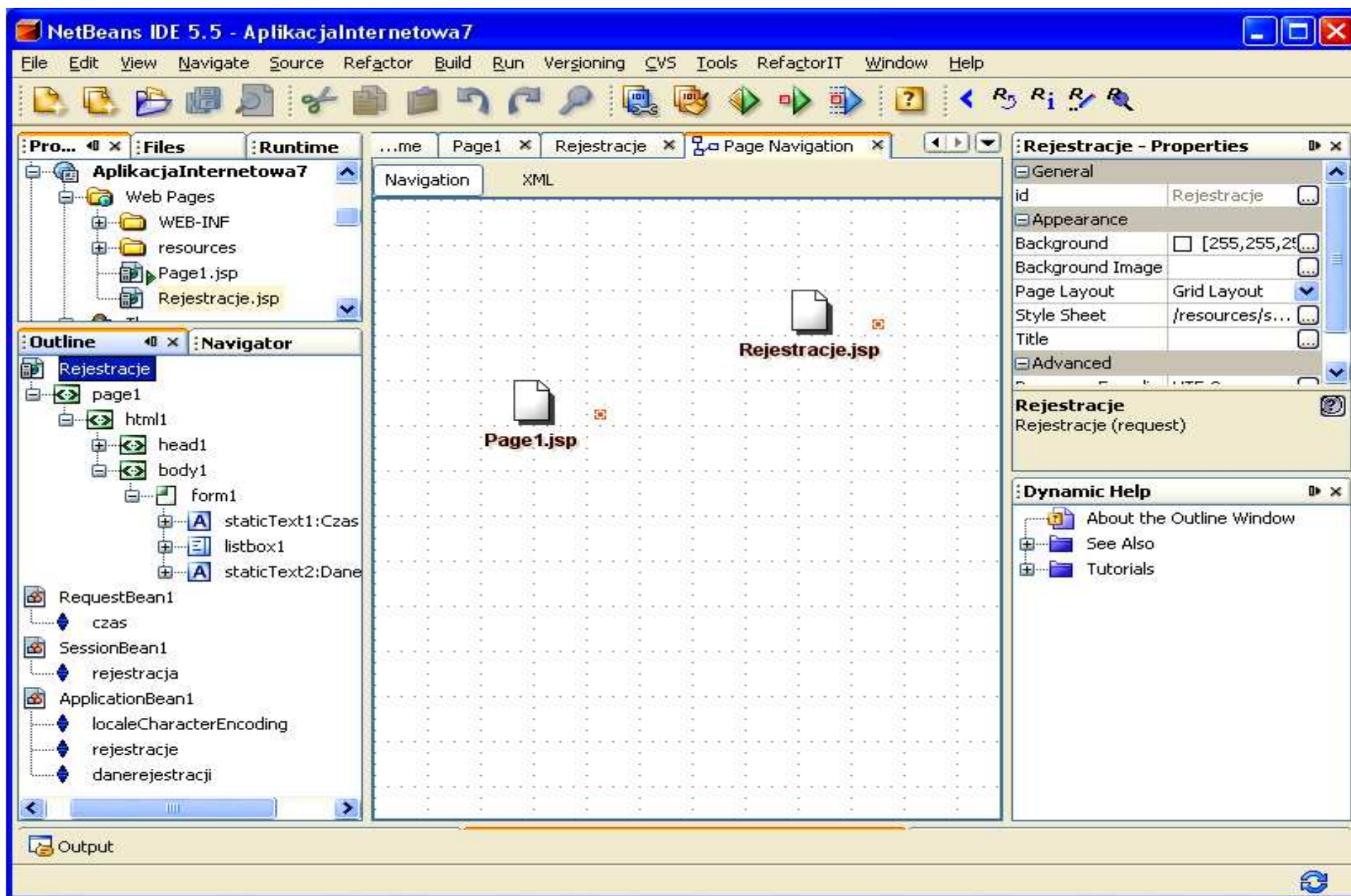
staticText1 - Properties	
General	
id	staticText1
Appearance	
style	height: 22px; left: 24p...
styleClass	
text	#{RequestBean1.cz...
Data	
converter	
escape	<input checked="" type="checkbox"/>
Behavior	
staticText1 staticText1 (StaticText)	

At the bottom of the IDE, there is an "Output" window and an "HTTP Monitor" window.

9. Wywołanie edytora Page Navigation z wyskakującego menu po naciśnięciu prawego klawisza myszy w obszarze Design strony



9.1. Utworzenie połączeń ze strony Page1 do strony Rezerwacje uruchamianych za pomocą klawiszy Pokaż rejestracje i Zarejestruj



NetBeans IDE 5.5 - AplikacjaInternetowa7

File Edit View Navigate Source Refactor Build Run Versioning CVS Tools RefactorIT Window Help

Navigation XML

Page1.jsp - Properties

Properties	
Name	Page1
All Files	C:\Dydaktyka\...
File Size	1943
Modification Time	2007-05-30 08:3...
Text	
Encoding	UTF-8

Page1.jsp
JSF JSP File

Dynamic Help

- About the Page Navigation Editor
- See Also
- Tutorials

NetBeans IDE 5.5

File Edit View Navigate Source Refactor Build Run Versioning CVS Tools RefactorIT Window Help

Navigation XML

No Properties

Dynamic Help

- About the Page Navigation Editor
- See Also
- Tutorials

NetBeans IDE 5.5 - AplikacjaInternetowa7

File Edit View Navigate Source Refactor Build Run Versioning CVS Tools RefactorIT Window Help

Navigation XML

Page1.jsp - Properties

Properties
Name Page1
Page1.jsp
JSF JSP File

Dynamic Help

- About the Page Navigation Editor
- See Also
- Tutorials

NetBeans IDE 5.5 - AplikacjaInternetowa7

File Edit View Navigate Source Refactor Build Run Versioning CVS Tools RefactorIT Window Help

Navigation XML

case1 - Properties

General
case1

Dynamic Help

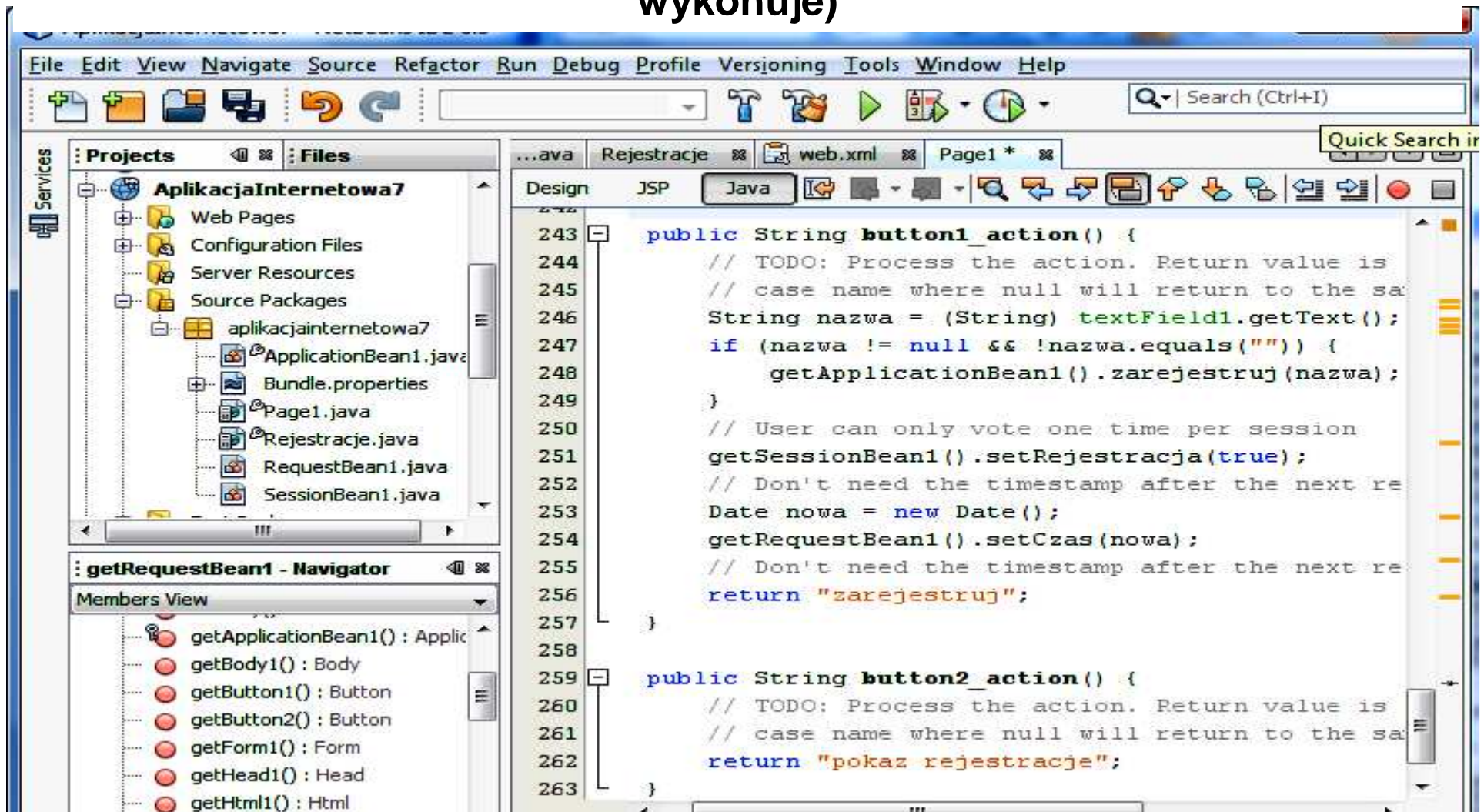
- About the Page Navigation Editor
- See Also
- Tutorials

Output

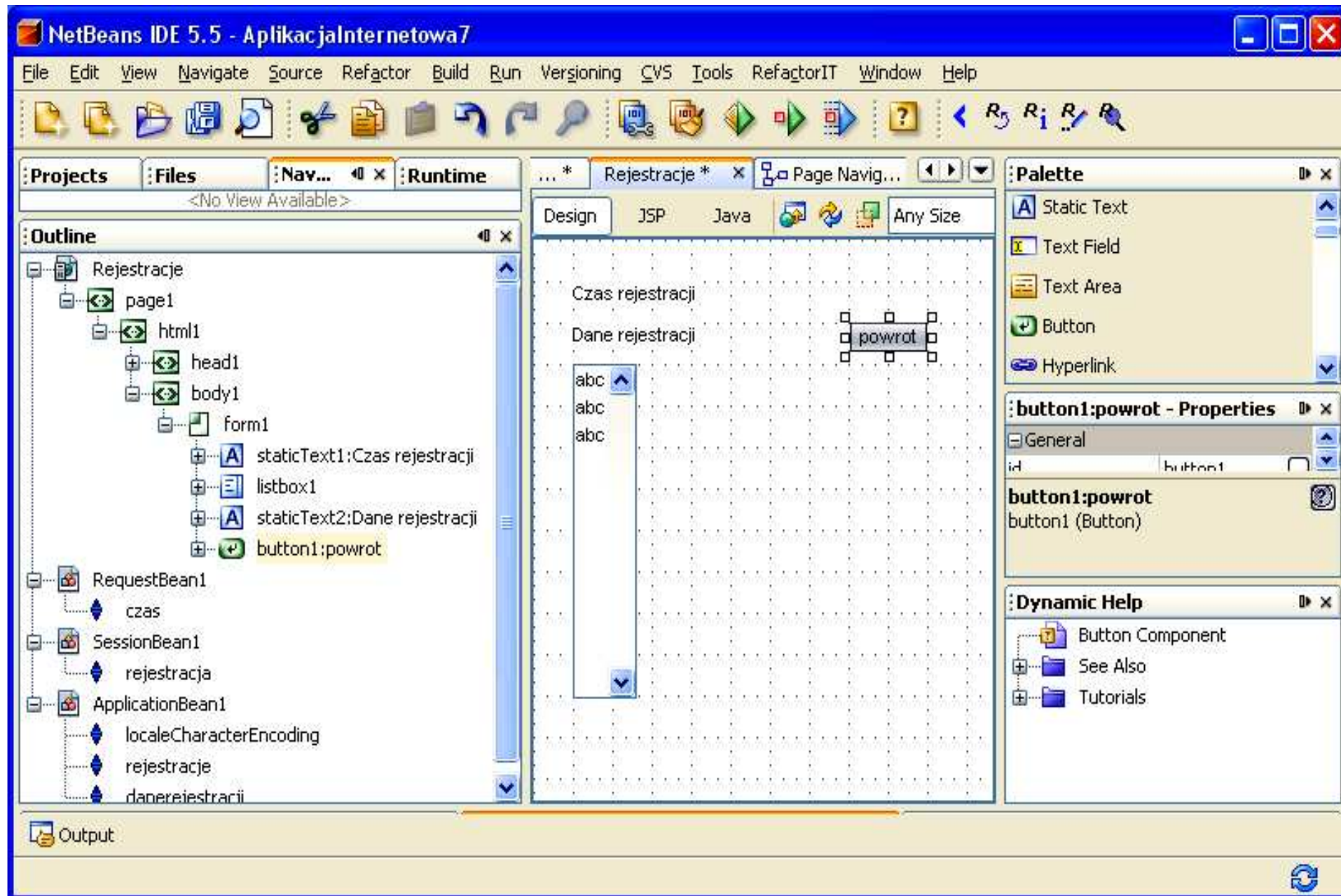
9.2. Obsługa zdarzeń wraz z obsługą połączeń strony Page1 ze stroną Rejestracje klawisza Zarejestruj (button1_action) oraz klawisza Pokaż rejestracje (button2_action)

```
public String button1_action() {  
    // TODO: Process the action. Return value is a navigation  
    // case name where null will return to the same page.  
    String nazwa = (String)textField1.getText();  
    if (nazwa !=null && !nazwa.equals(""))  
        getApplicationBean1().zarejestruj( nazwa);  
    // User can only vote one time per session  
    getSessionBean1().setRejestracja(true);  
    // Don't need the timestamp after the next request ends  
    Date nowa = new Date();  
    getRequestBean1().setCzas(nowa);  
    // Don't need the timestamp after the next request ends  
    return "zarejestruj"; //obsługa polaczenia ze strona Rejestracje  
}  
  
public String button2_action() {  
    // TODO: Process the action. Return value is a navigation  
    // case name where null will return to the same page.  
    return "pokaz rejestracje"; //obsługa polaczenia ze strona Rejestracje  
}
```


9.3. Rezultat działań – strona **Page1** posiada metody obsługi zdarzeń przycisków **Zarejestruj** (metoda **button1_action** podczas połączenia ze stroną **Rejestracje** realizuje rejestrację klienta oraz wyłącza przycisk **Zarejestruj** oraz ustawia **czas rejestracji**) i **Pokaż rejestracje** (metoda **button2_action** podczas połączenia ze stroną **Rejestracje** nic nie wykonuje)



9.4. Obsługa klawisza powrot – wstawienie połączenia do strony Page1 ze strony Rejestracje



NetBeans IDE 5.5 - Aplikacja Internetowa7

File Edit View Navigate Source Refactor Build Run Versioning CVS Tools RefactorIT Window Help

Projects Files Navig... Runtime

Outline

- Rejestracje
 - page1
 - html1
 - head1
 - body1
 - form1
 - staticText1:Czas rejestracji
 - listbox1
 - staticText2:Dane rejestracji
 - button1:powrot

Navigation XML

Page1.jsp

Rejestracje.jsp

button1

case1

zarejestruj

pokaz rejestracje

Rejestr... Properties

Name R

Rejestracje.jsp JSF JSP File

Dynam... About th See Also Tutorials

NetBeans IDE 5.5 - Aplikacja Internetowa7

File Edit View Navigate Source Refactor Build Run Versioning CVS Tools RefactorIT Window Help

Projects Files Navig... Runtime

Outline

- Rejestracje
 - page1
 - html1
 - head1
 - body1
 - form1
 - staticText1:Czas rejestracji
 - listbox1
 - staticText2:Dane rejestracji
 - button1:powrot

Navigation XML

Page1.jsp

Rejestracje.jsp

powrot

zarejestruj

pokaz rejestracje

case1 -... General

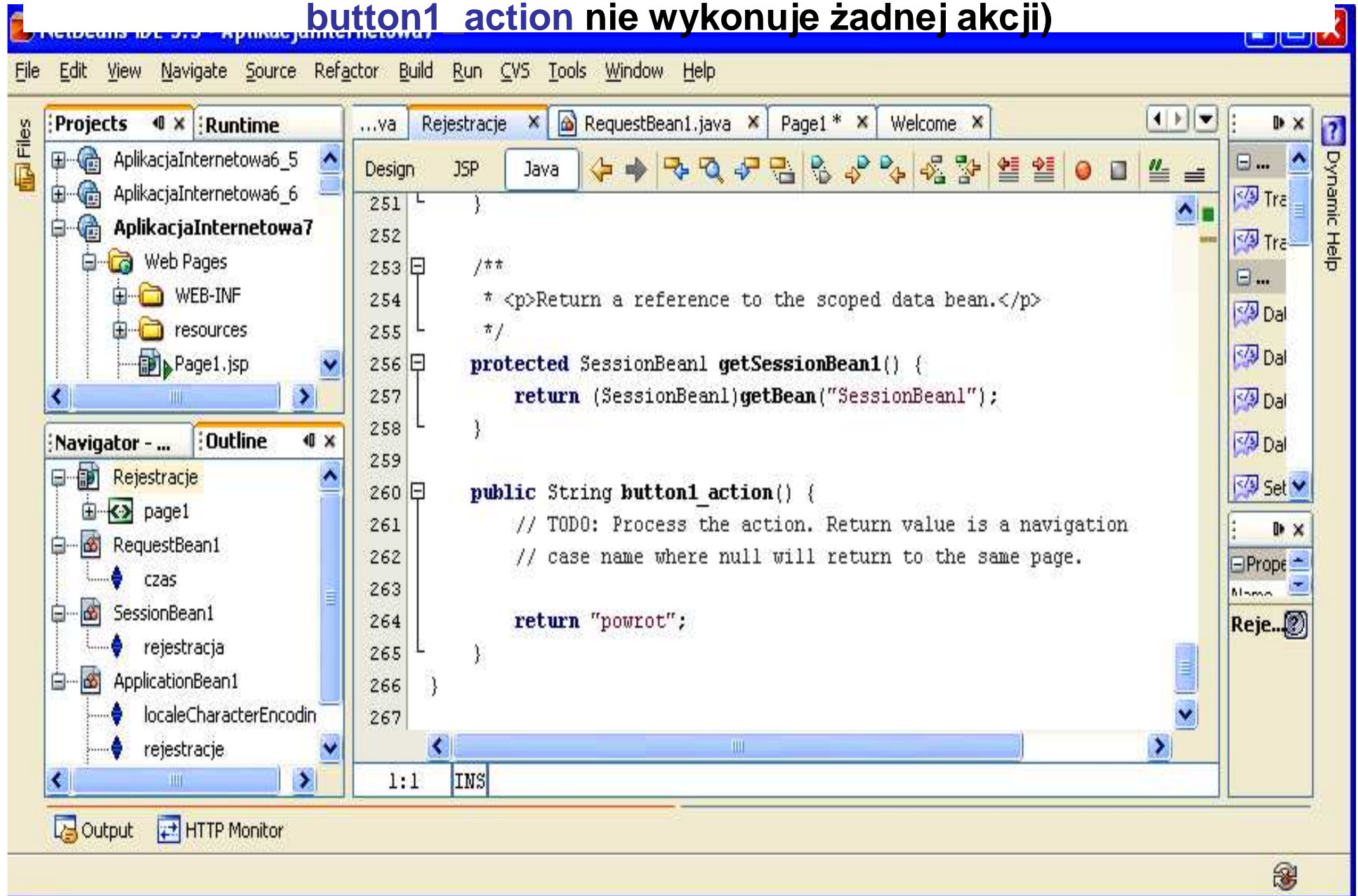
Outcome

case1

Dynam... About th See Also Tutorials

Output

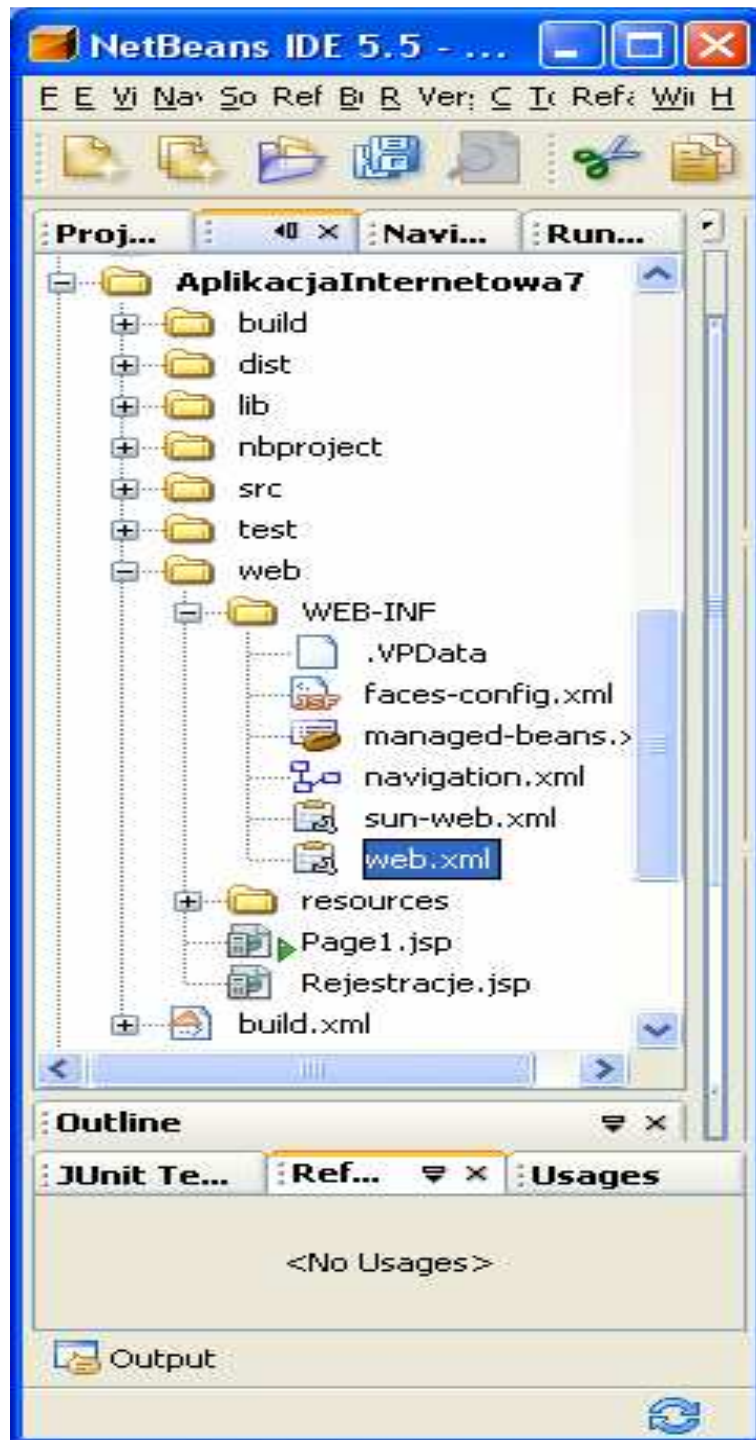
9.5. Obsługa połączenia ("powrot") strony Rejestracje ze stroną Page1 za pomocą klawisza powrot (oprócz połączenia ze stroną Page1 metoda `button1_action` nie wykonuje żadnej akcji)



The screenshot displays the NetBeans IDE interface. The main editor window shows the source code of `RequestBean1.java`. The code is as follows:

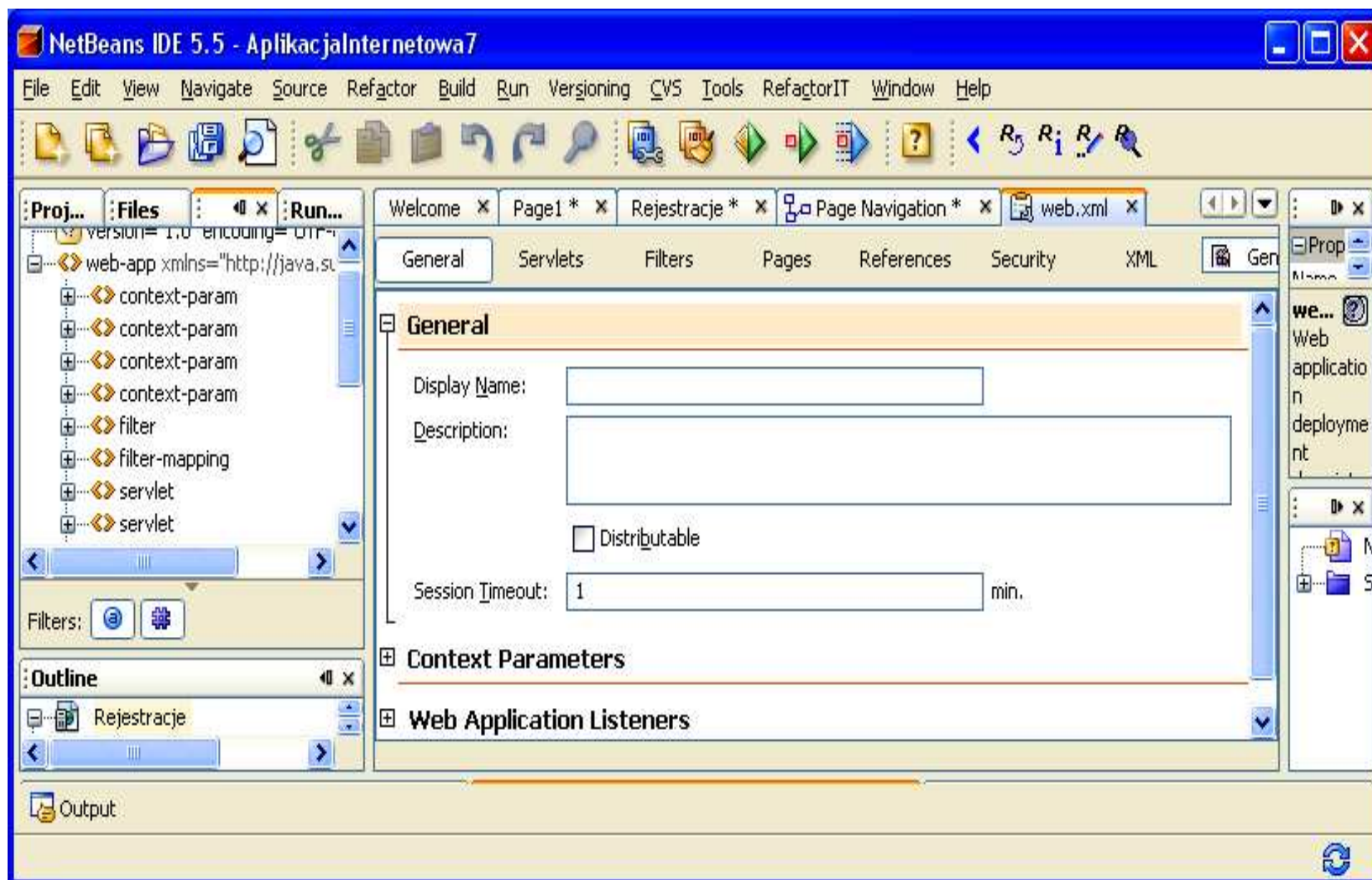
```
251     }
252
253     /**
254      * <p>Return a reference to the scoped data bean.</p>
255      */
256     protected SessionBean1 getSessionBean1() {
257         return (SessionBean1)getBean("SessionBean1");
258     }
259
260     public String button1_action() {
261         // TODO: Process the action. Return value is a navigation
262         // case name where null will return to the same page.
263
264         return "powrot";
265     }
266 }
267
```

The IDE interface includes a **Projects** pane on the left showing the project structure for `AplikacjaInternetowa7`, with `Page1.jsp` selected. Below it is the **Navigator** pane showing the `Rejestracje` page and its associated beans. The **Outline** pane is also visible. The **Dynamic Help** pane on the right shows the help content for the `button1_action` method. The **Output** and **HTTP Monitor** panes are visible at the bottom.

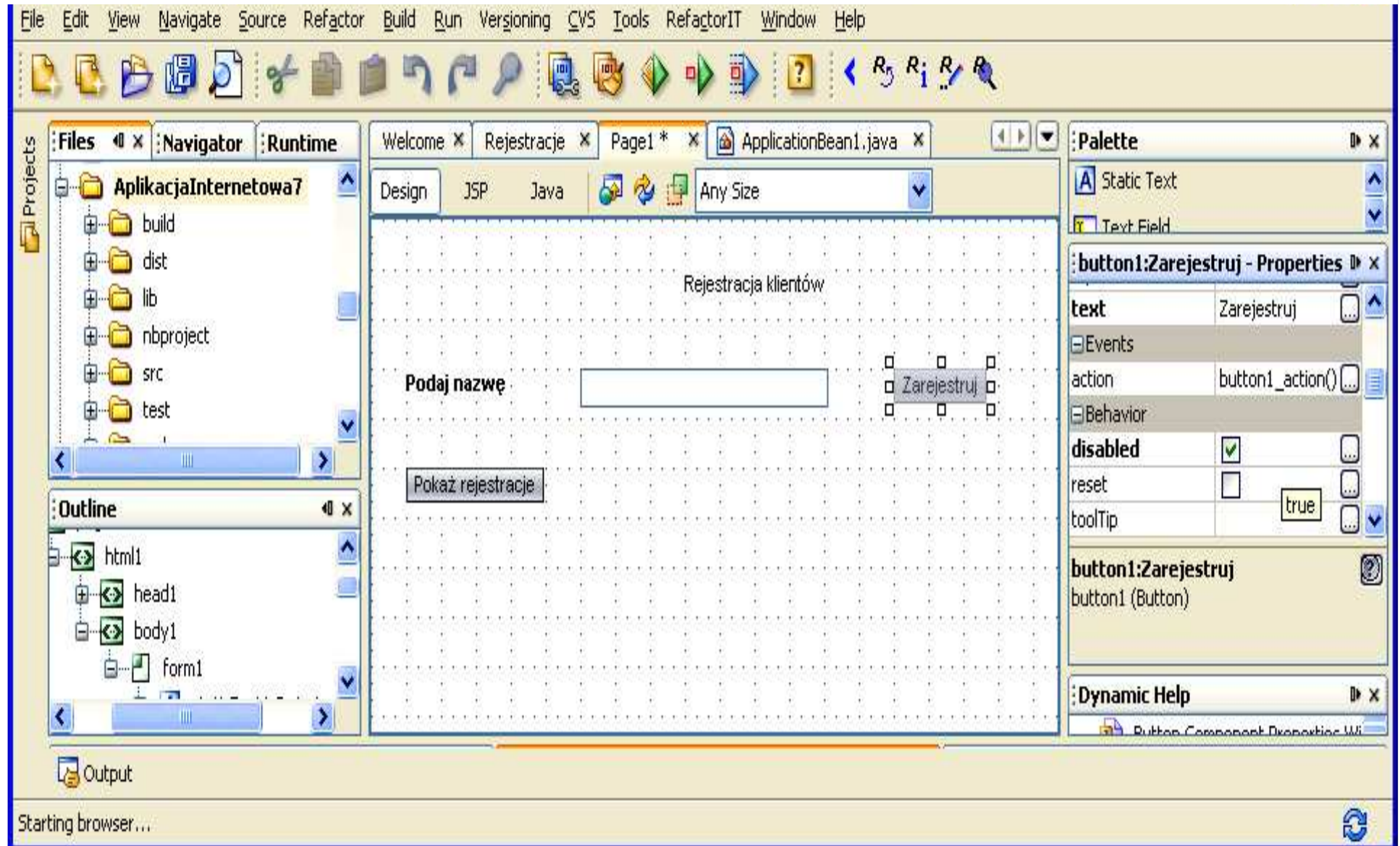


10. Ustawienie czasu typu Timeout dla obiektu typu Sessionbean1 w pliku web.xml

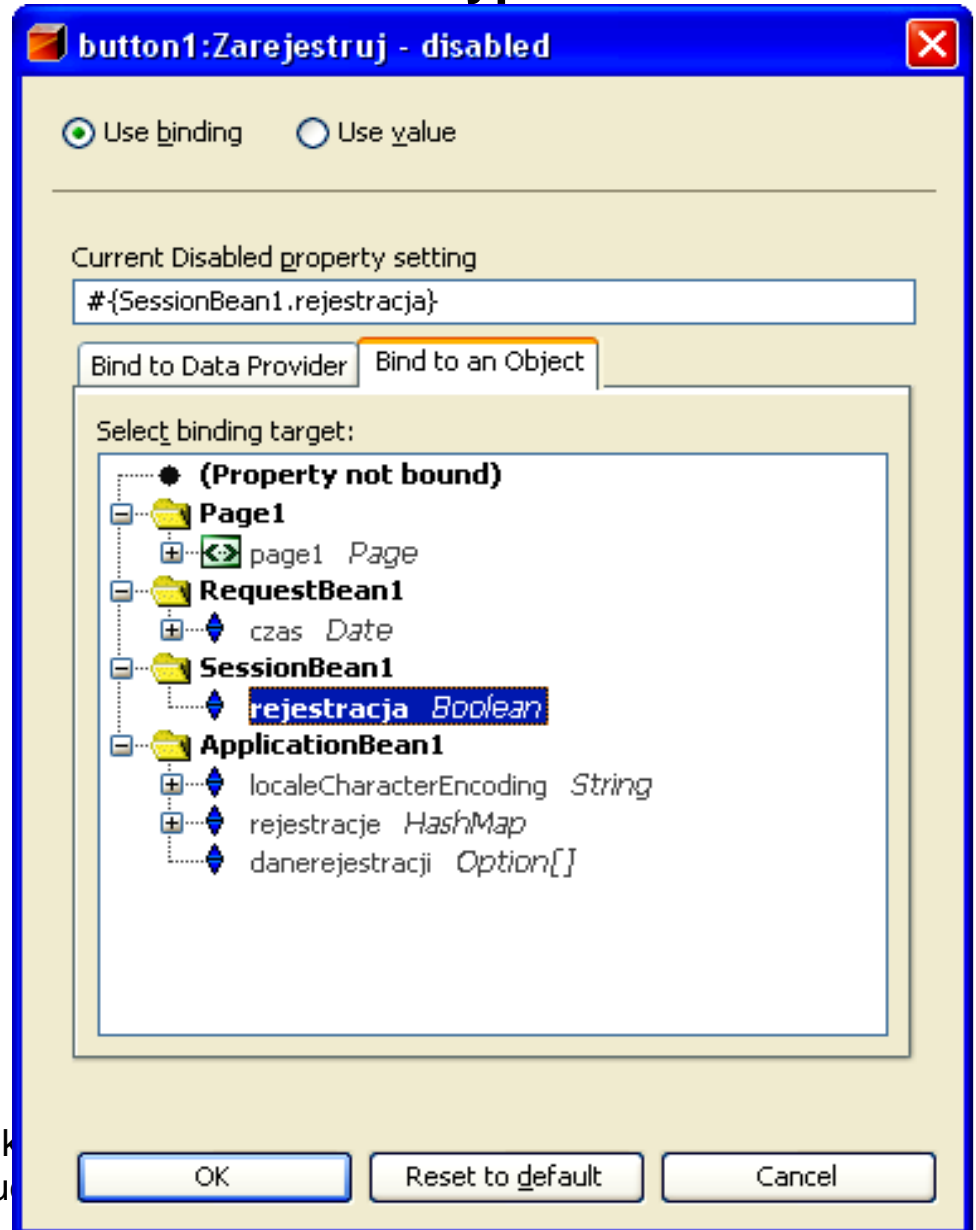
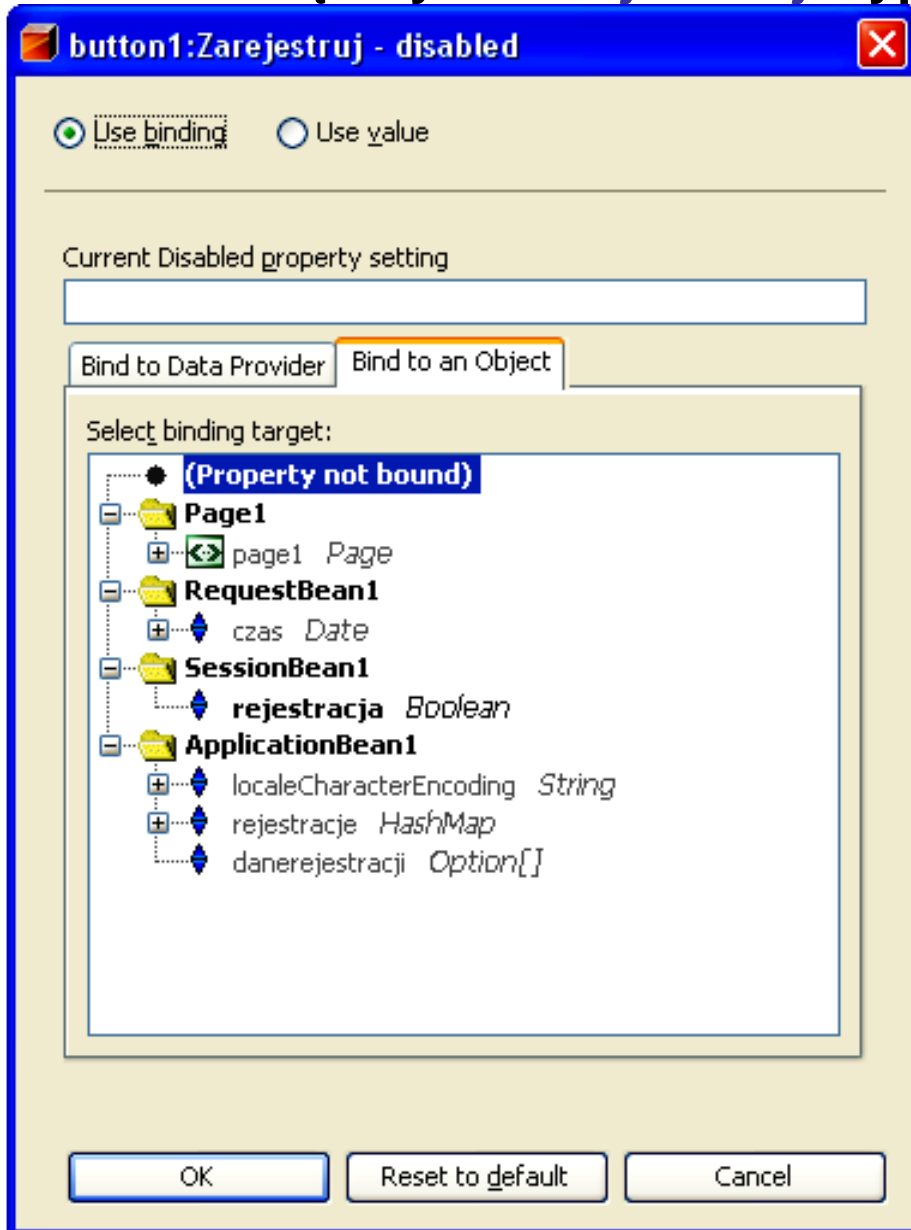
10.1. Formularz pliku web.xml



11. Ustawianie właściwości **disabled** klawisza **Zarejestruj** w celu zablokowania wywołania zdarzenia rejestracji za pomocą metody **button1_action**– ustawienie klawisza **checkbox** oraz wywołanie edytora tej właściwości

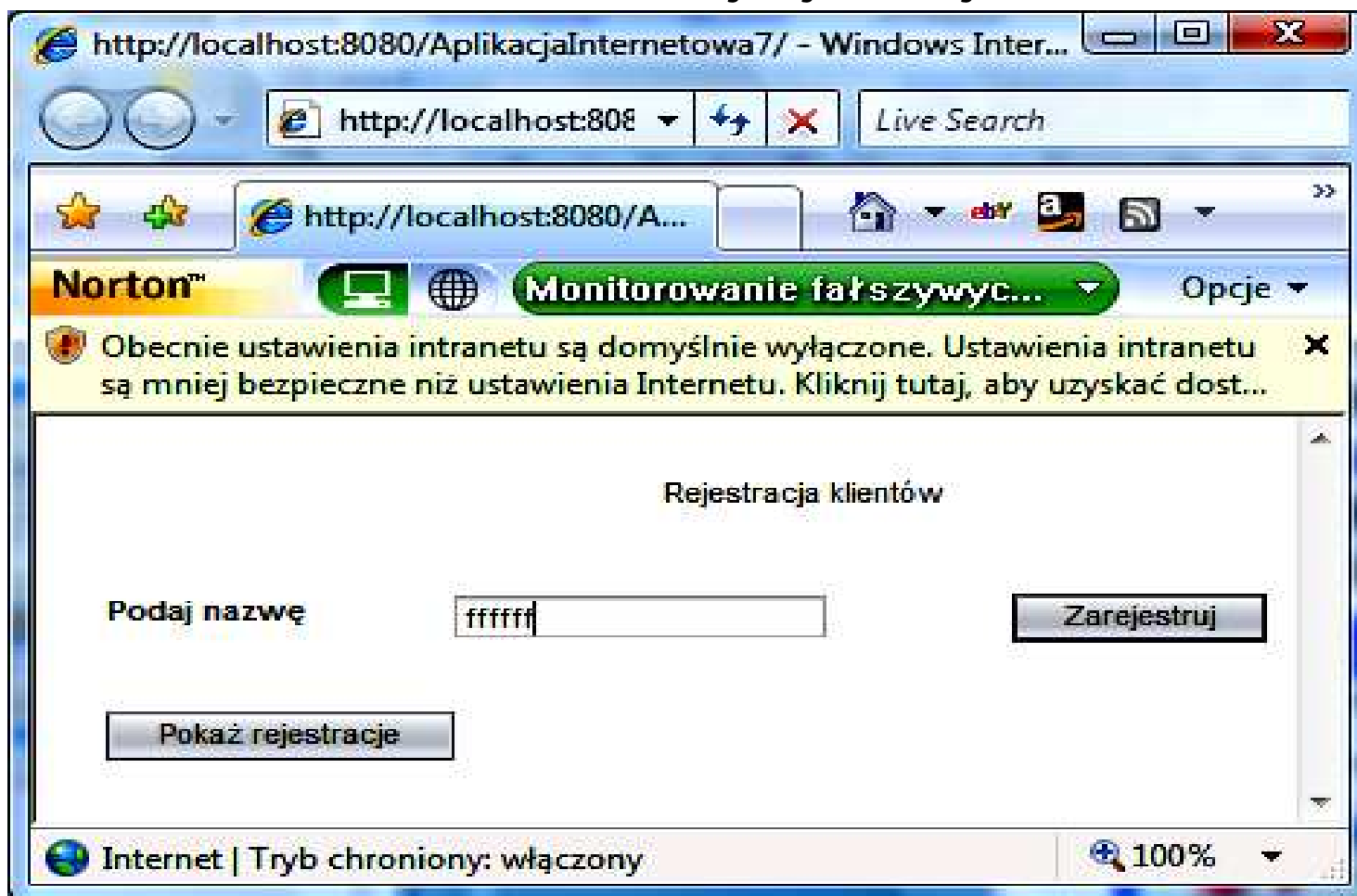


11.1. Edytor właściwości disabled klawisza Zarejestruj - ustawienie właściwości klawisza typu disabled w zakładce Bind to an Object zgodnie z wartością atrybutu rejestracja typu boolean obiektu typu SessionBean1

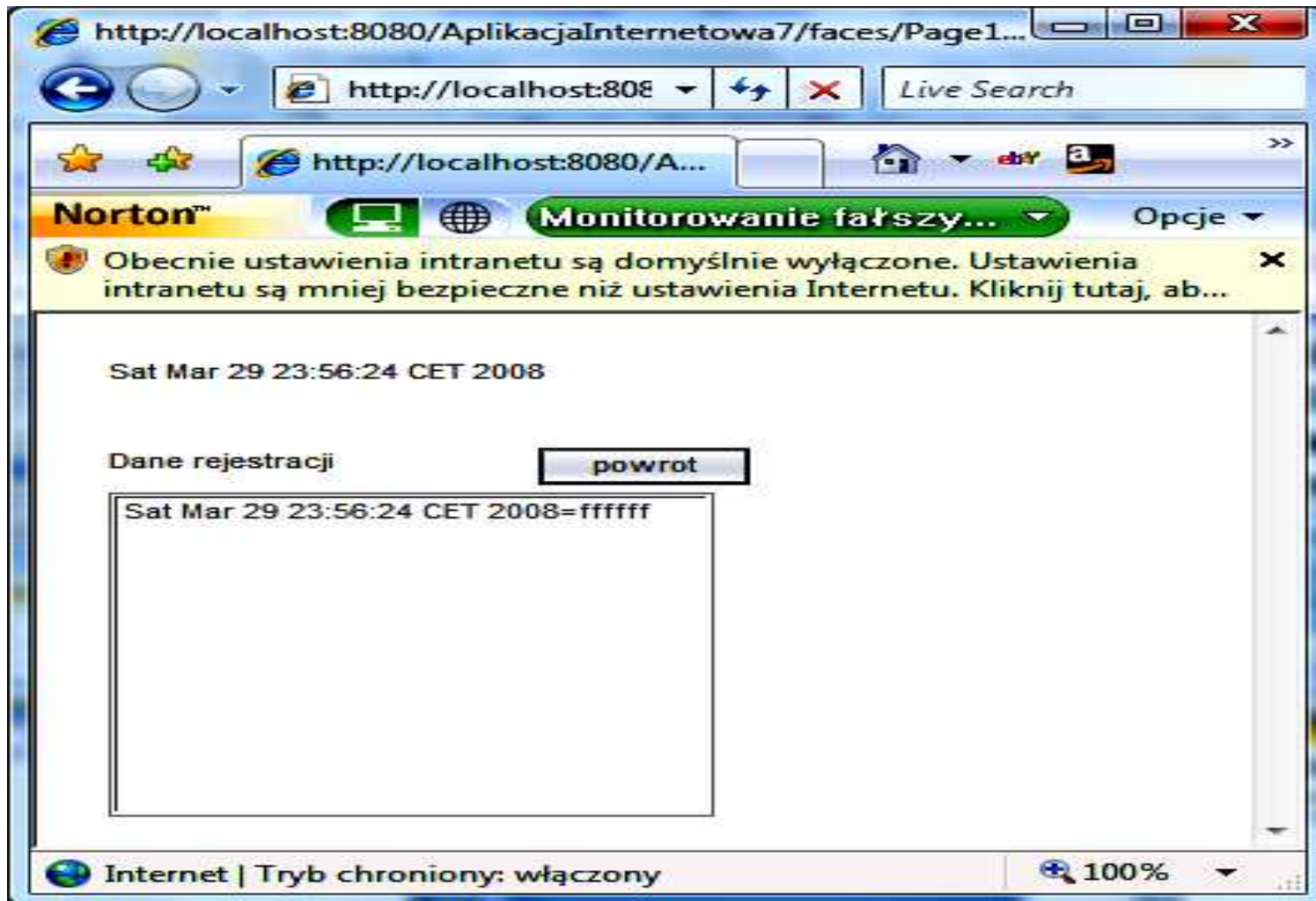


e k
Kru

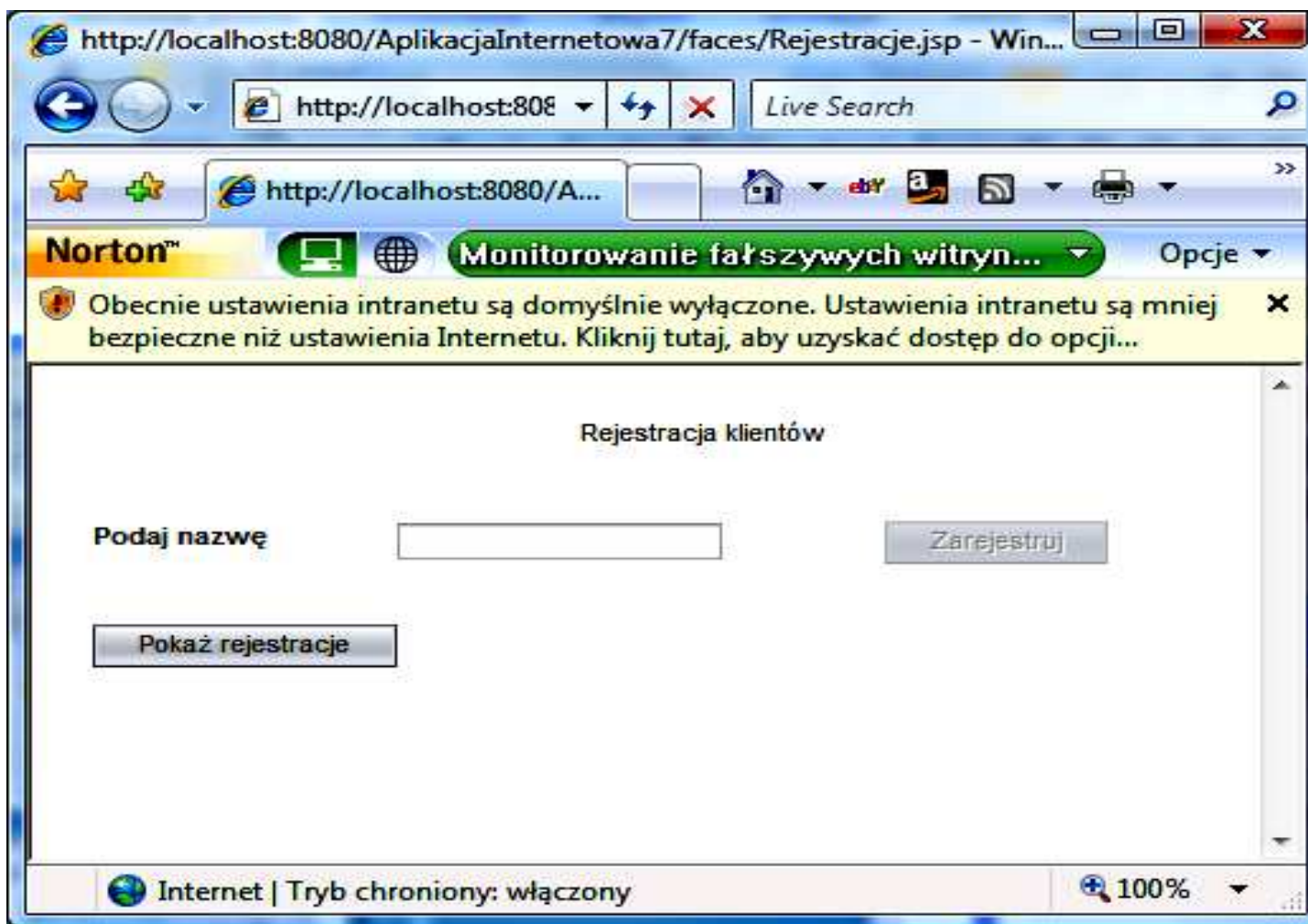
12. Stan strony głównej po wywołaniu strony głównej aplikacji – klawisz **Pokaż rejestracje** realizuje połączenie ze stroną **Rejestracje**, która prezentuje rejestracje klientów wykonanych podczas działania całej aplikacji. Klawisz **Zarejestruj** wywołuje obsługę rejestracji klienta podczas realizacji połączenia ze stroną **Rejestracje**, która prezentuje rejestracje klientów wykonanych podczas działania całej aplikacji i jednocześnie czas ostatniej rejestracji.



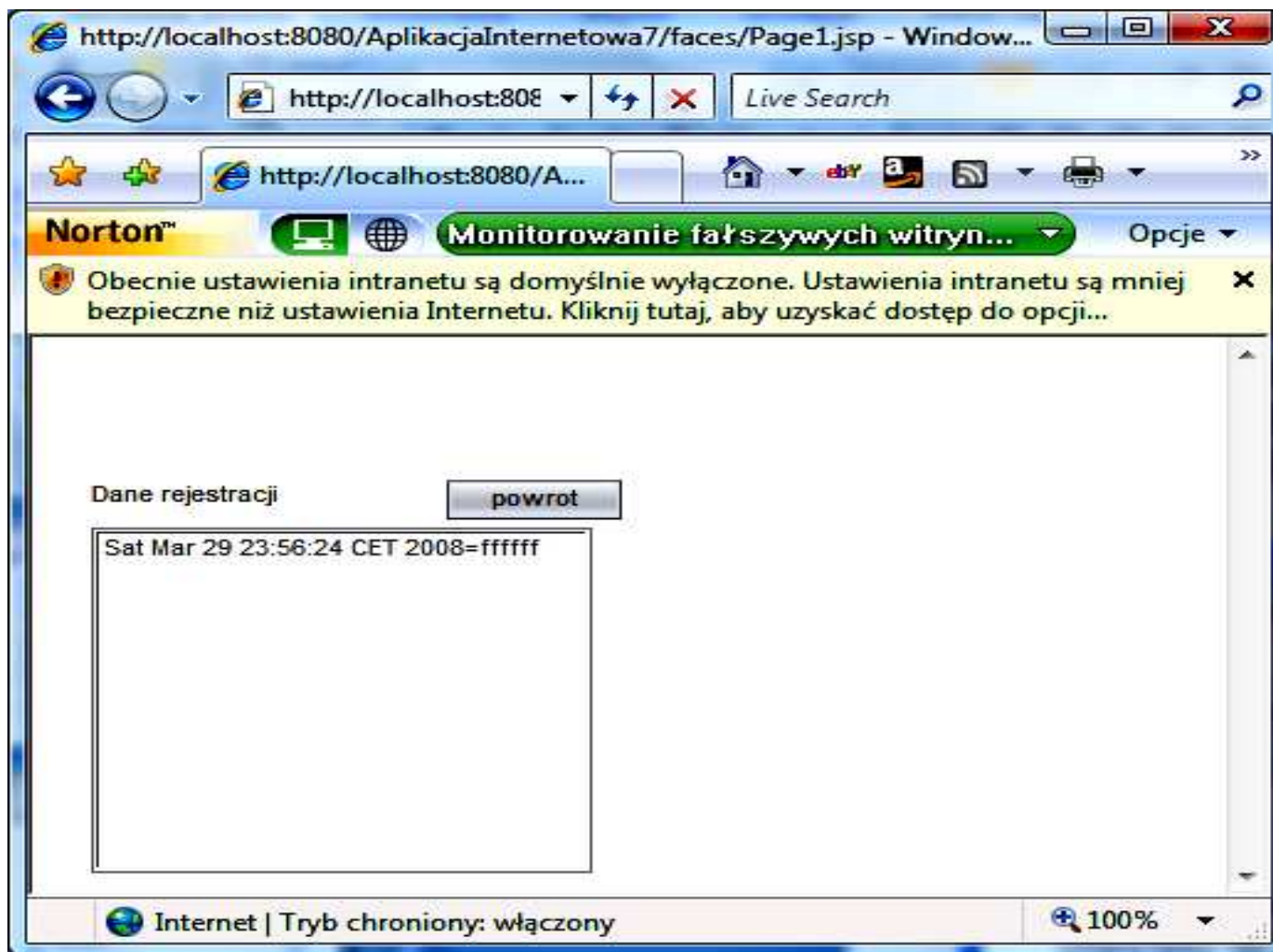
12.1. Stan po naciśnięciu klawisza **Zarejestruj** na stronie **Page1**, który wywołuje stronę **Rejestracje**, przekazuje zawartość atrybutu **danerejestracji** z danymi rejestracji z obiektu typu **ApplicationBean1** za pomocą komponentu typu **ListBox** oraz wartość atrybutu **czas** z obiektu typu **RequestBean1** (czas ostatniej rejestracji), które są wyświetlane w komponentach strony typu **List Box** oraz **StaticText**



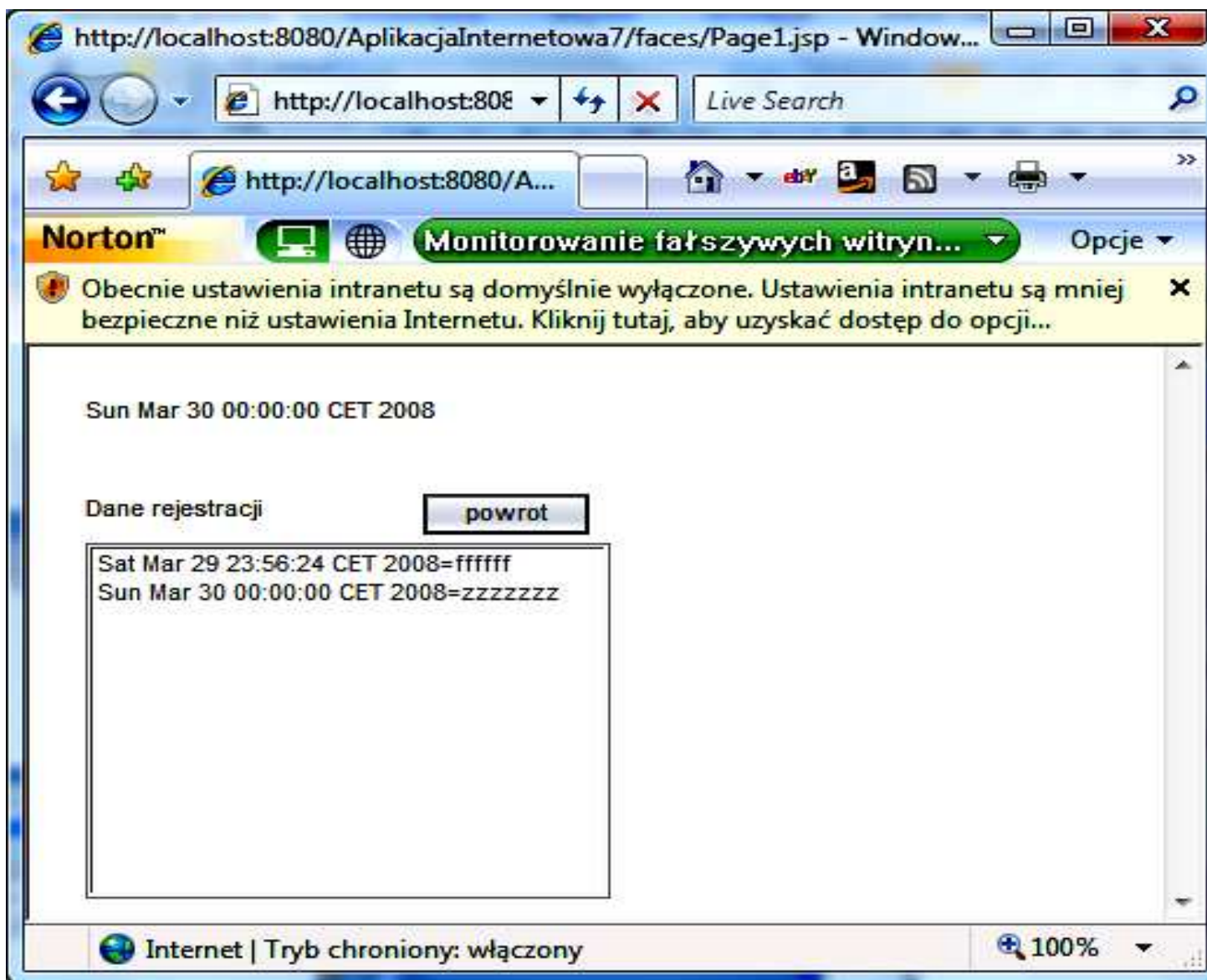
12.2. Stan zablokowania rejestracji z powodu stanu **Disable** klawisza **Zarejestruj**, który trwa od momentu wykonania rejestracji i połączenia się ze stroną **Rejestracje** do czasu zakończenia bieżącej instancji obiektu **SessionBean1** po upływie czasu wyznaczonego przez **Timeout** (1min) - kończy się w chwili utworzenia nowego obiektu typu **SessionBean1**



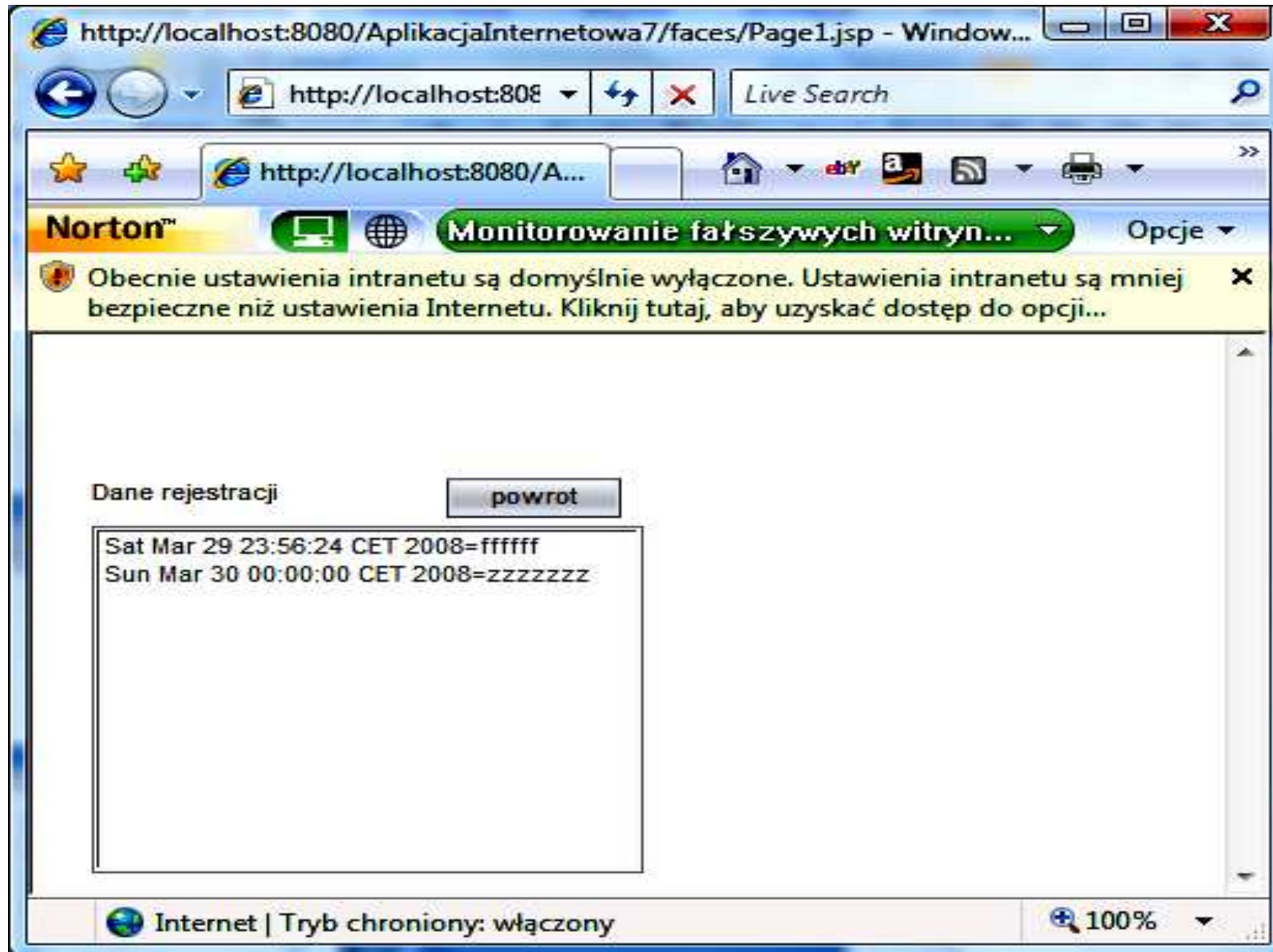
12.3. Wywołanie strony **Rejestracje** za pomocą klawisza **Pokaż rejestracje**.
Dane atrybutu **czas** nie można już wyświetlić, ponieważ istnieje już inny egzemplarz obiektu **RequestBean1**, w którym nie ma już zapisanego czasu ostatniej rejestracji w atrybucie **czas** w momencie obsługi klawisza **Zarejestruj** (czyli realizacji fazy **request** po połączeniu ze stroną **Rejestracje**)

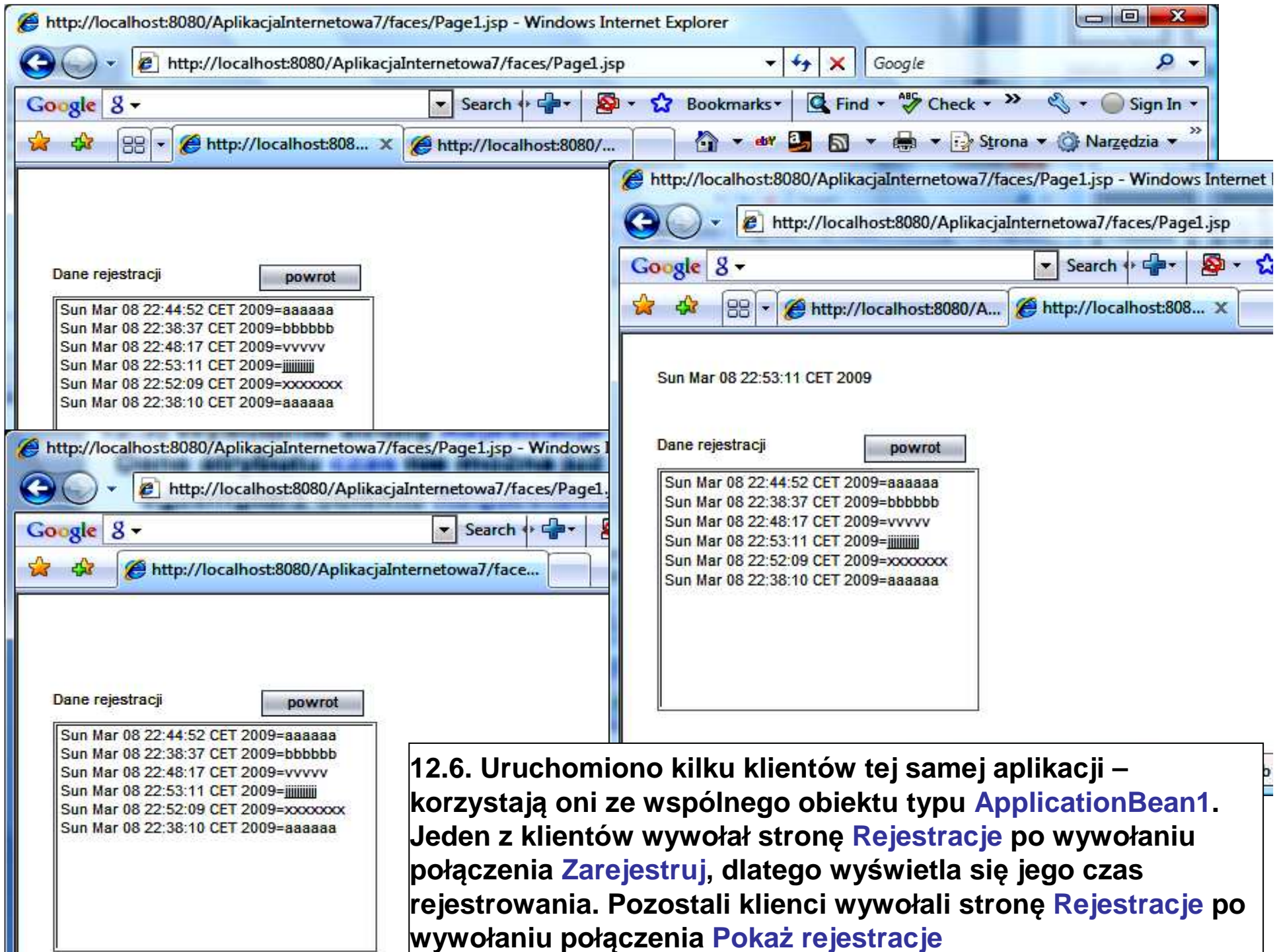


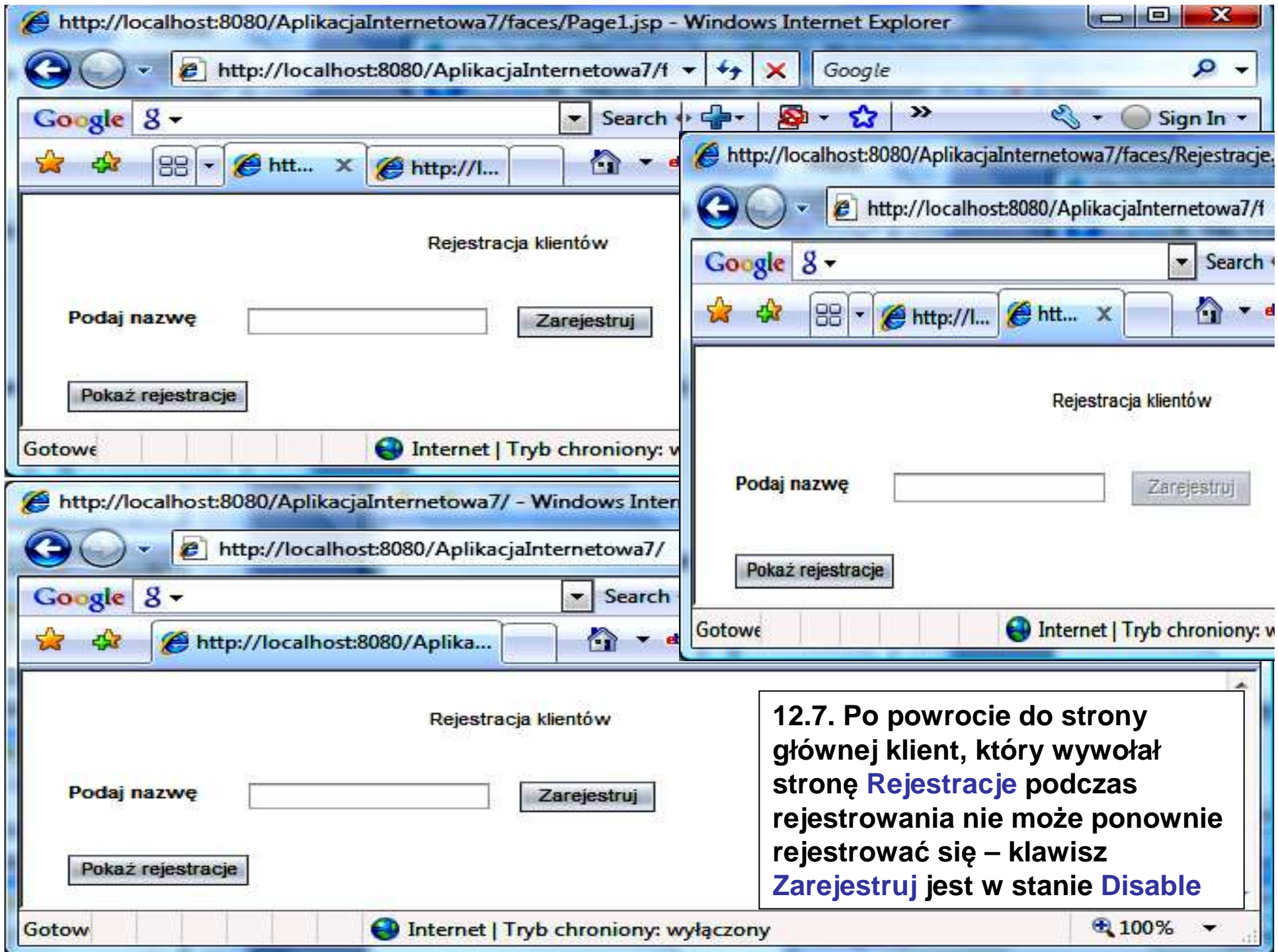
12.4. Ponowna obsługa klawisza **Zarejestruj** w momencie utworzenia nowych obiektów typu **SessionBean1** oraz typu **RequestBean1**



12.5. Ponowna obsługa klawisza Pokaz rejestracje w momencie utworzenia nowego obiektu typu RequestBean1







12.7. Po powrocie do strony głównej klient, który wywołał stronę **Rejestracje** podczas rejestrowania nie może ponownie rejestrować się – klawisz **Zarejestruj** jest w stanie **Disable**