28

Web Services

A client is tome a mere unit,
a factor in a problem.

—Sir Arthur Conan Doyle

They also serve who only stand and wait.

—John Milton

...if the simplest things of nature have a message that you understand, rejoice, for your soul is alive.

—Eleonora Duse

Protocol is everything.

—Francoise Giuliani

OBJECTIVES

In this chapter you will learn:

- What a web service is.
- How to publish and consume Java web services in Netbeans.
- The elements that comprise web services, such as service descriptions and classes that implement web services.
- How to create client desktop and web applications that invoke web service methods.

OBJECTIVES

- The important part that XML and the Simple Object Access Protocol (SOAP) play in enabling web services.
- How to use session tracking in web services to maintain client state information.
- How to connect to databases from web services.
- How to pass objects of user-defined types to and return them from a web service.
- How to build a REST-based web service in ASP.NET.

Outline

Outline

Outline

28.1 Introduction

- **Web service**
  - **A software component stored on one computer that can be accessed via method calls by an application (or other software component) on another computer over a network**
- **Web services communicate using such technologies as XML and HTTP**
- **Simple Object Access Protocol (SOAP)**
  - **An XML-based protocol that allows web services and clients to communicate in a platform-independent manner**

28.1 Introduction

- **Companies**
  - **Amazon, Google, eBay, PayPal and many others make their server-side applications available to partners via web services**
- **By using web services, companies can spend less time developing new applications and can create innovative new applications**
- **Netbeans 5.5.1 enables programmers to “publish” and/or “consume” web services**

28.2 Java Web Services Basics

- **Remote machine or server**
  - **The computer on which a web service resides**
- **A client application that accesses a web service sends a method call over a network to the remote machine, which processes the call and returns a response over the network to the application**
- **In Java, a web service is implemented as a class that resides on a server**
- **Publishing a web service**
  - **Making a web service available to receive client requests**
- **Consuming a web service**
  - **Using a web service from a client application**

28.2 Java Web Services Basics (Cont.)

- **An application that consumes a web service consists of two parts**
  - **An object of a proxy class for interacting with the web service**
  - **A client application that consumes the web service by invoking methods on the proxy object**
  - **The proxy object handles the details of communicating with the web service on the client's behalf**
- **JAX-WS 2.0**
  - **Requests to and responses from web services are typically transmitted via SOAP**
  - **Any client capable of generating and processing SOAP messages can interact with a web service, regardless of the language in which the web service is written**

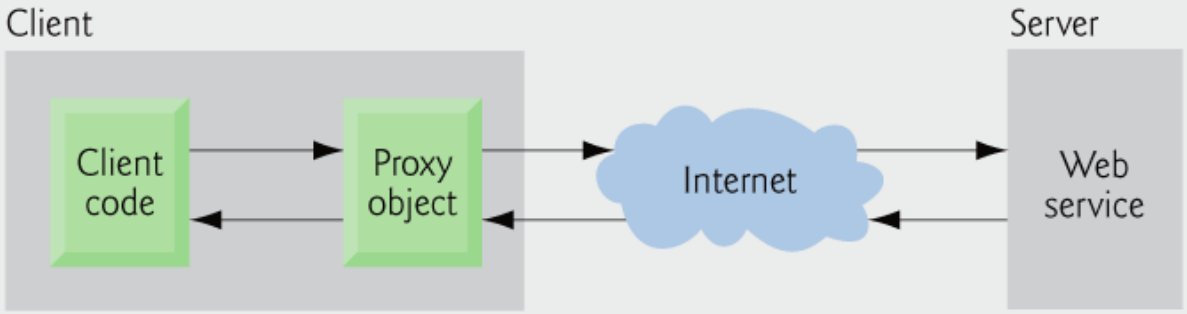


Fig. 28.1 | Interaction between a web service client and a web service.

28.3 Creating, Publishing, Testing and Describing a Web Service

- **HugeInteger web service**
  - **Provides methods that take two “huge integers” (represented as Strings)**
  - **Can determine their sum, their difference, which is larger, which is smaller or whether the two numbers are equal**

28.3.1 Creating a Web Application Project and Adding a Web Service Class in Netbeans

- **In Netbeans, you focus on the logic of the web service and let the IDE handle the web service's infrastructure**
- **To create a web service in Netbeans**
  - **Create a project of type Web Application**
  - **The IDE generates additional files that support the web application**

28.3.2 Defining the `HugeInteger` Web Service in Netbeans

- **Each new web service class created with the JAX-WS APIs is a POJO (plain old Java object)**
  - **You do not need to extend a class or implement an interface to create a Web service**
- **When you compile a class that uses these JAX-WS 2.0 annotations, the compiler creates the compiled code framework that allows the web service to wait for and respond to client requests**

28.3.2 Defining the `HugeInteger` Web Service in Netbeans (Cont.)

- **`@WebService` annotation**
  - **Indicates that a class represents a web service**
  - **Optional element `name` specifies the name of the proxy class that will be generated for the client**
  - **Optional element `serviceName` specifies the name of the class that the client uses to obtain a proxy object.**

28.3.2 Defining the `HugeInteger` Web Service in Netbeans (Cont.)

- Netbeans places the `@WebService` annotation at the beginning of each new web service class you create
- You can add the optional `name` and `serviceName` elements in the annotation's parentheses
- Methods that are tagged with the `@WebMethod` annotation can be called remotely
- Methods that are not tagged with `@WebMethod` are not accessible to clients that consume the web service

28.3.2 Defining the `HugeInteger` Web Service in Netbeans (Cont.)

- **@WebMethod annotation**
  - **Optional `operationName` element to specify the method name that is exposed to the web service's client**
- **Parameters of web methods are annotated with the `@WebParam` annotation**
  - **Optional element `name` indicates the parameter name that is exposed to the web service's clients**

Outline

```java
// Fig. 28.2: HugeInteger.java
// HugeInteger web service that performs operations on large integers.
package com.deitel.iw3htp4.ch28.hugeinteger;

import javax.jws.WebService; // program uses the annotation @WebService
import javax.jws.WebMethod; // program uses the annotation @WebMethod
import javax.jws.WebParam; // program uses the annotation @WebParam

@WebService( // annotates the class as a web service
   name = "HugeInteger", // sets class name
   serviceName = "HugeIntegerService" ) // sets the service name
public class HugeInteger
{
   private final static int MAXIMUM = 100; // maximum number of digits
   public int[] number = new int[ MAXIMUM ]; // stores the huge integer

   // returns a String representation of a HugeInteger
   public String toString()
   {
      String value = "";

      // convert HugeInteger to a String
      for ( int digit : number )
         value = digit + value; // places next digit at beginning of value

      // locate position of first non-zero digit
      int length = value.length();
      int position = -1;

```

Import the annotations used in this example

Indicate that class `HugeInteger` is a web service

```
30       for ( int i = 0; i < length; i++ )
31       {
32          if ( value.charAt( i ) != '0' )
33          {
34             position = i; // first non-zero digit
35             break;
36          }
37       } // end for
38
39       return ( position != -1 ? value.substring( position ) : "0" );
40    } // end method toString
41
42    // creates a HugeInteger from a String
43    public static HugeInteger parseHugeInteger( String s )
44    {
45       HugeInteger temp = new HugeInteger();
46       int size = s.length();
47
48       for ( int i = 0; i < size; i++ )
49          temp.number[ i ] = s.charAt( size - i - 1 ) - '0';
50
51       return temp;
52    } // end method parseHugeInteger
53
```

Outline

HugeInteger.java

(2 of 6)

```
54   // WebMethod that adds huge integers represented by String arguments
55   @WebMethod( operationName = "add" )
56   public String add( @WebParam( name = "first" ) String first,
57      @WebParam( name = "second" ) String second )
58   {
59      int carry = 0; // the value to be carried
60      HugeInteger operand1 = HugeInteger.parseHugeInteger( first );
61      HugeInteger operand2 = HugeInteger.parseHugeInteger( second );
62      HugeInteger result = new HugeInteger(); // stores addition result
63
64      // perform addition on each digit
65      for ( int i = 0; i < MAXIMUM; i++ )
66      {
67         // add corresponding digits in each number and the carried value;
68         // store result in the corresponding column of HugeInteger result
69         result.number[ i ] =
70            ( operand1.number[ i ] + operand2.number[ i ] + carry ) % 10;
71
72         // set carry for next column
73         carry =
74            ( operand1.number[ i ] + operand2.number[ i ] + carry ) / 10;
75      } // end for
76
77      return result.toString();
78   } // end WebMethod add
79
```

Declare that method `add` is a web method

HugeInteger.java

(3 of 6)

```
80    // WebMethod that subtracts integers represented by String arguments
81    @WebMethod( operationName = "subtract" )
82    public String subtract( @WebParam( name = "first" ) String first,
83       @WebParam( name = "second" ) String second )
84    {
85       HugeInteger operand1 = HugeInteger.parseHugeInteger( first );
86       HugeInteger operand2 = HugeInteger.parseHugeInteger( second );
87       HugeInteger result = new HugeInteger(); // stores difference
88
89       // subtract bottom digit from top digit
90       for ( int i = 0; i < MAXIMUM; i++ )
91       {
92          // if the digit in operand1 is smaller than the corresponding
93          // digit in operand2, borrow from the next digit
94          if ( operand1.number[ i ] < operand2.number[ i ] )
95             operand1.borrow( i );
96
97          // subtract digits
98          result.number[ i ] = operand1.number[ i ] - operand2.number[ i ];
99       } // end for
100
101      return result.toString();
102   } // end WebMethod subtract
103
```

Declare that method `subtract` is a web method

HugeInteger.java

(4 of 6)

Outline

HugeInteger.java

(5 of 6)

```
104   // borrow 1 from next digit
105   private void borrow( int place )
106   {
107      if ( place >= MAXIMUM )
108         throw new IndexOutOfBoundsException();
109      else if ( number[ place + 1 ] == 0 ) // if next digit is zero
110         borrow( place + 1 ); // borrow from next digit
111
112      number[ place ] += 10; // add 10 to the borrowing digit
113      --number[ place + 1 ]; // subtract one from the digit to the left
114   } // end method borrow
115
116   // WebMethod that returns true if first integer is greater than second
117   @WebMethod( operationName = "bigger" )
118   public boolean bigger( @WebParam( name = "first" ) String first,
119      @WebParam( name = "second" ) String second )
120   {
121      try // try subtracting first from second
122      {
123         String difference = subtract( first, second );
124         return !difference.matches( "^[0]+$" );
125      } // end try
126      catch ( IndexOutOfBoundsException e ) // first is less than second
127      {
128         return false;
129      } // end catch
130   } // end WebMethod bigger
131
```

Declare that method `bigger` is a web method

```
132   // WebMethod that returns true if the first integer is less than second
133   @WebMethod( operationName = "smaller" )
134   public boolean smaller( @WebParam( name = "first" ) String first,
135      @WebParam( name = "second" ) String second )
136   {
137      return bigger( second, first );
138   } // end WebMethod smaller
139
140   // WebMethod that returns true if the first integer equals the second
141   @WebMethod( operationName = "equals" )
142   public boolean equals( @WebParam( name = "first" ) String first,
143      @WebParam( name = "second" ) String second )
144   {
145      return !( bigger( first, second ) || smaller( first, second ) );
146   } // end WebMethod equals
147} // end class HugeInteger
```

Declare that method `smaller` is a web method

Declare that method `equals` is a web method

HugeInteger.java

(6 of 6)

Common Programming Error 28.1

Failing to expose a method as a web method by declaring it with the `@WebMethod` annotation prevents clients of the web service from accessing the method.

Common Programming Error 28.2

Methods with the `@WebMethod` annotation cannot be `static`. An object of the web service class must exist for a client to access the service's web methods.

28.3.3 Publishing the `HugeInteger` Web Service from Netbeans

- **Netbeans handles all the details of building and deploying a web service for you**
  - **Includes creating the framework required to support the web service**
- **To build project**
  - **Right click the project name in the Netbeans Projects tab**
  - **Select Build Project**
- **To deploy**
  - **Select Deploy Project**
  - **Deploys to the server you selected during application setup**
  - **Also builds the project if it has changed and starts the application server if it is not already running**
- **To Execute**
  - **Select Run Project**
  - **Also builds the project if it has changed and starts the application server if it is not already running**
- **To ensure a clean re-build of the entire project**
  - **Select Clean Project or Clean and Build Project**

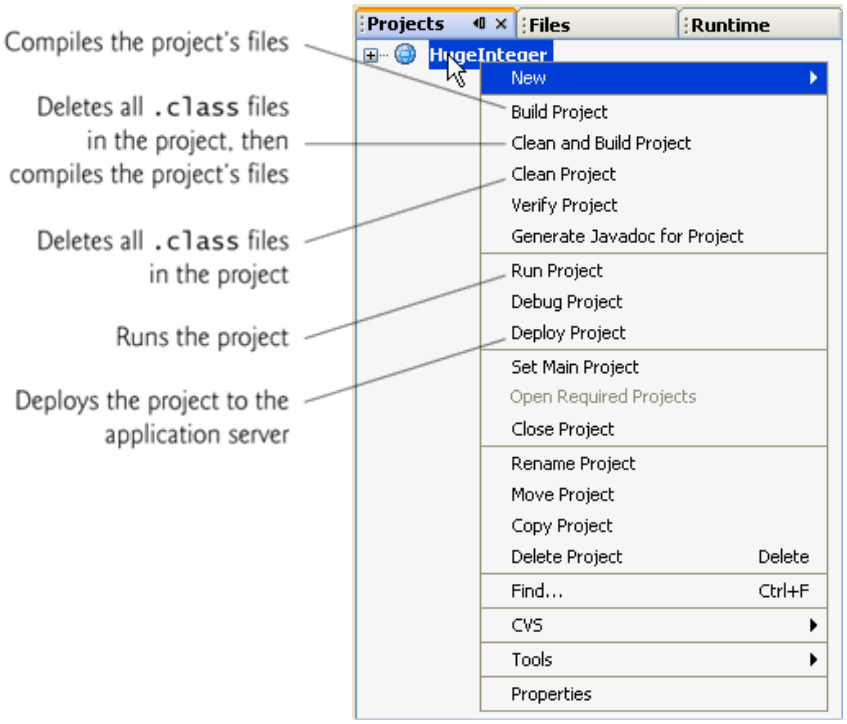


Fig. 28.3 | Pop-up menu that appears when you right click a project name in the Netbeans **Projects** tab.

28.3.4 Testing the `HugeInteger` Web Service with Sun Java System Application Server's Tester Web page

- **Sun Java System Application Server**
  - **Can dynamically create a `Tester` web page for testing a web service's methods from a web browser**
  - **Enable this feature via the project's Run options**
- **To display the `Tester` web page**
  - **Run the web application from Netbeans, or**
  - **Type web service's URL in browser's address field followed by `?Tester`**
- **Web server must be running for a client to access a web service**
  - **If Netbeans launches the application server for you, the server will shut down when you close Netbeans**
  - **To keep it running, launch it independently of Netbeans**

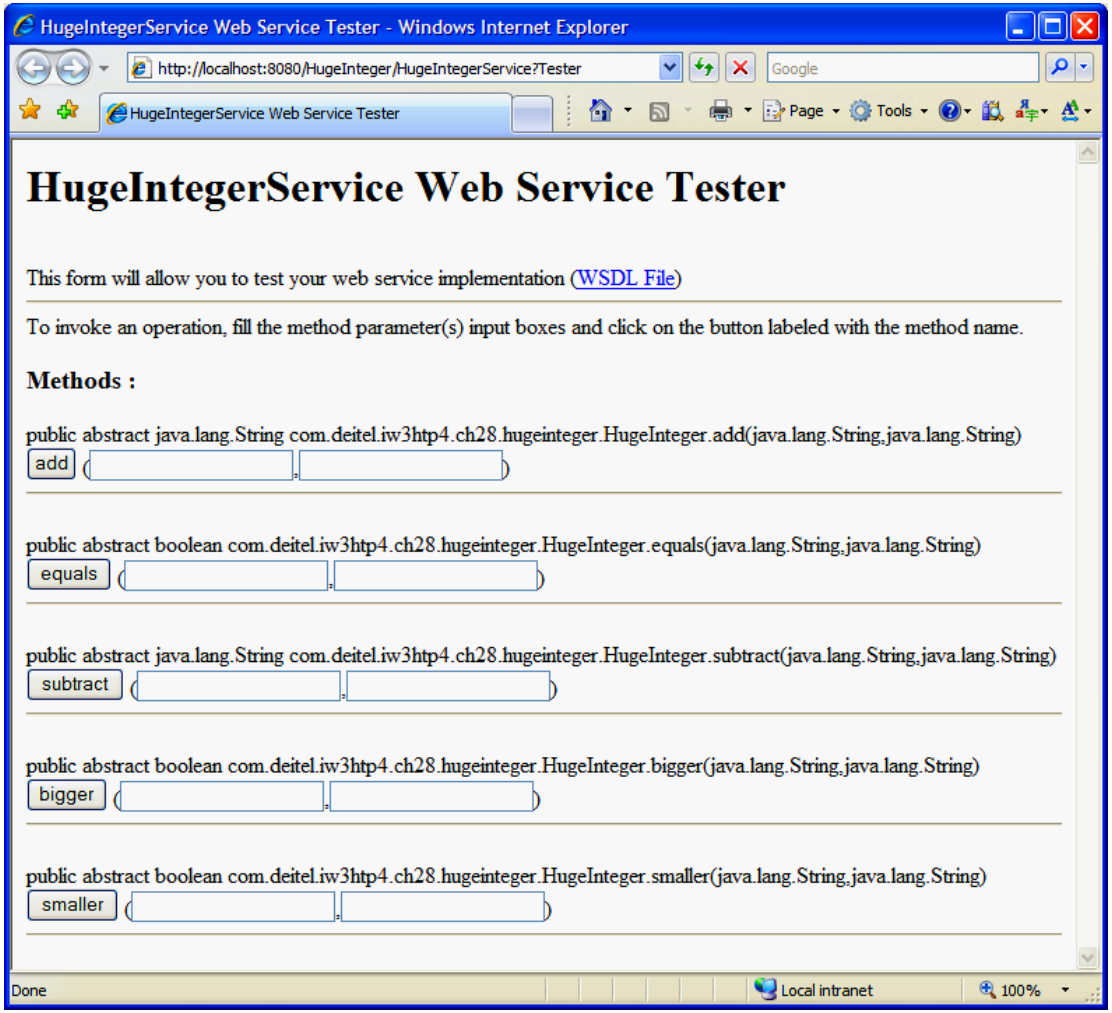
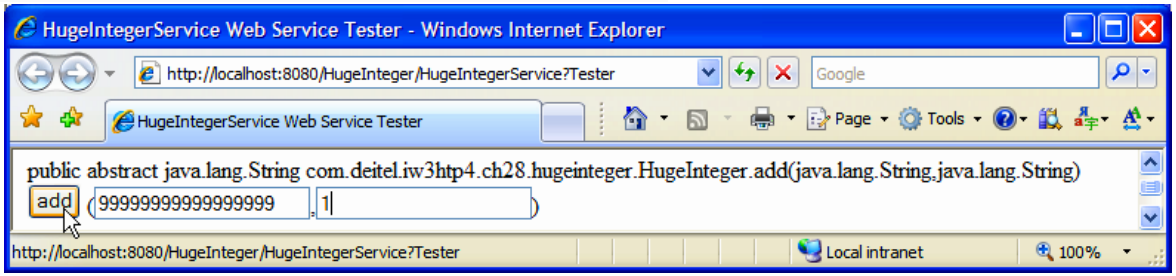


Fig. 28.4 | `Tester` web page created by Sun Java System Application Server for the `HugeInteger` web service.

a) Invoking the `HugeInteger` web service's `add` method.



b) Results of calling the `HugeInteger` web service's `add` method with `"99999999999999999"` and `"1"`

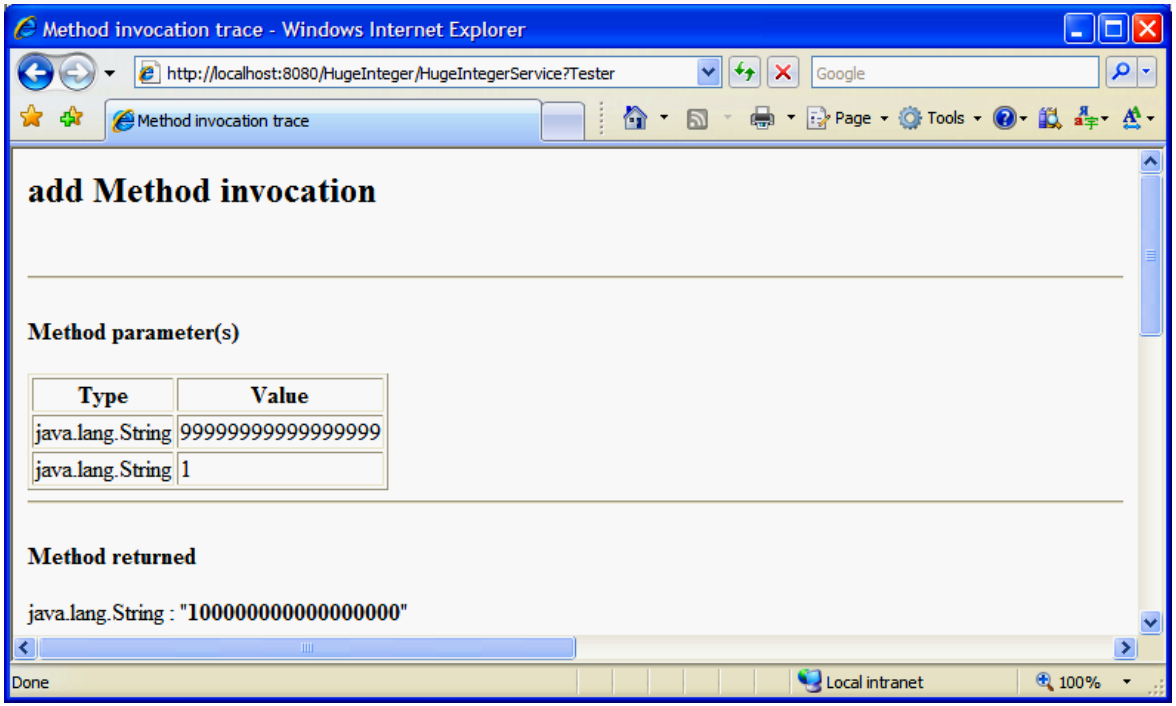


Fig. 28.5 | Testing `HugeInteger`'s `add` method.

28.3.5 Describing a Web Service with the Web Service Description Language (WSDL)

- **To consume a web service**
  - **Must know where to find the web service**
  - **Must be provided with the web service's description**
- **Web Service Description Language (WSDL)**
  - **Describe web services in a platform-independent manner**
  - **The server generates a web service's WSDL dynamically for you**
  - **Client tools parse the WSDL to create the client-side proxy class that accesses the web service**
- **To view the WSDL for a web service**
  - **Type URL in the browser's address field followed by `?WSDL` or**
  - **Click the `WSDL File` link in the Sun Java System Application Server's `Tester` web page**

28.4 Consuming a Web Service

- **Web service client can be any type of application or even another web service**
- **Web service reference**
  - **Enables a client application to consume a web service**
  - **Defines the client-side proxy class**

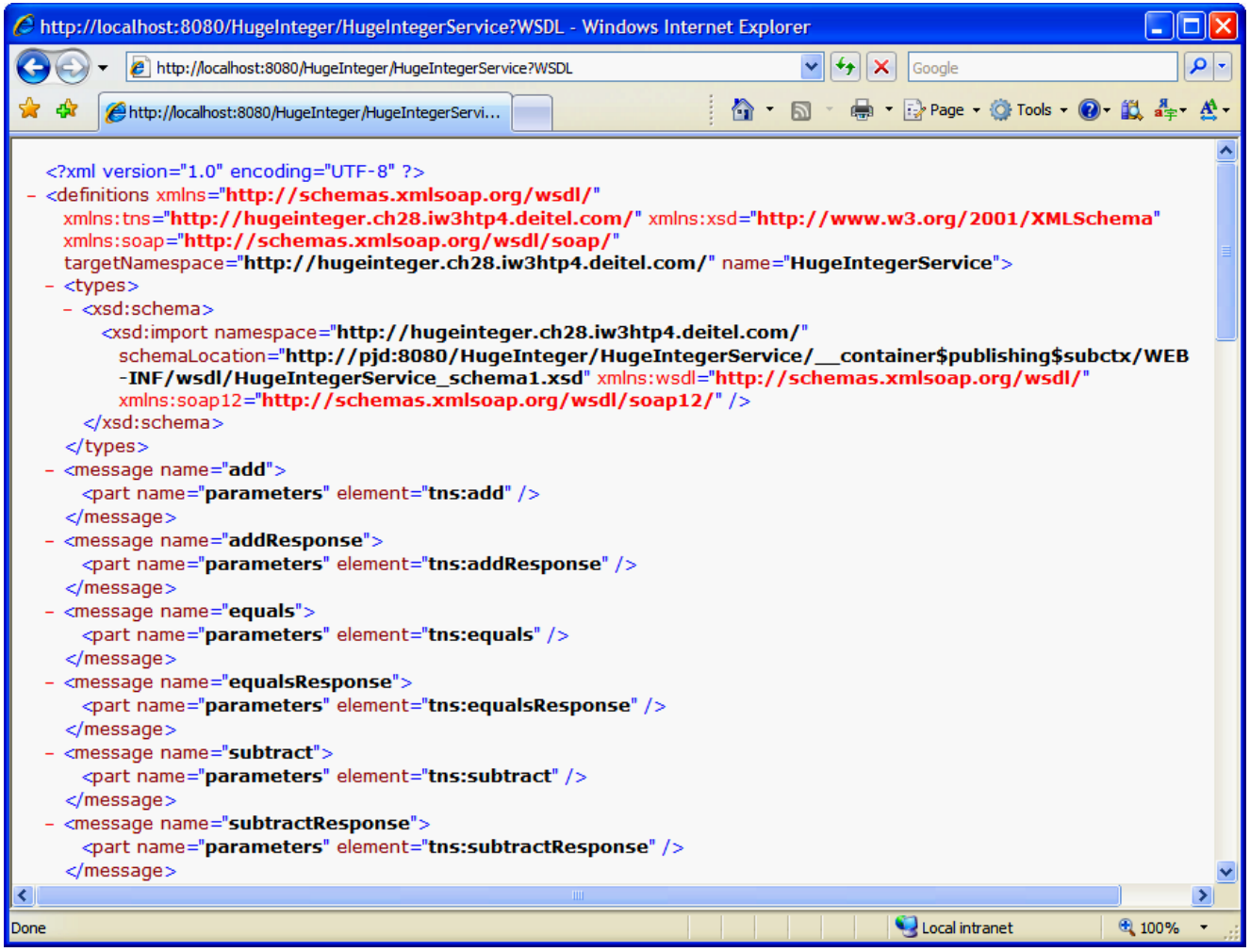


Fig. 28.6 | A portion of the `.wsdl` file for the `HugeInteger` web service.

28.4.1 Creating a Client in Netbeans to Consume the `HugeInteger` Web Service

- **When you add a web service reference**
  - **IDE creates and compiles the client-side artifacts—the framework of Java code that supports the client-side proxy class**
- **Client calls methods on a proxy object**
  - **Proxy uses client-side artifacts to interact with the web service**
- **To add a web service reference**
  - **Right click the client project name in the Netbeans Projects tab**
  - **Select New > Web Service Client…**
  - **Specify the URL of the web service's WSDL in the dialog's WSDL URL field**

28.4.1 Creating a Client in Netbeans to Consume the `HugeInteger` Web Service (Cont.)

- Netbeans uses the WSDL description to generate the client-side proxy class and artifacts
- Netbeans copies the web service's WSDL into a file in your project
  - Can view this file from the Netbeans **Files** tab
  - Expand the nodes in the project's **xml-resources** folder.
- To update client-side artifacts and client's WSDL copy
  - Right click the web service's node in the Netbeans **Projects** tab
  - Select **Refresh Client**
- To view the IDE-generated client-side artifacts
  - Select the Netbeans **Files** tab
  - Expand the project's **build** folder

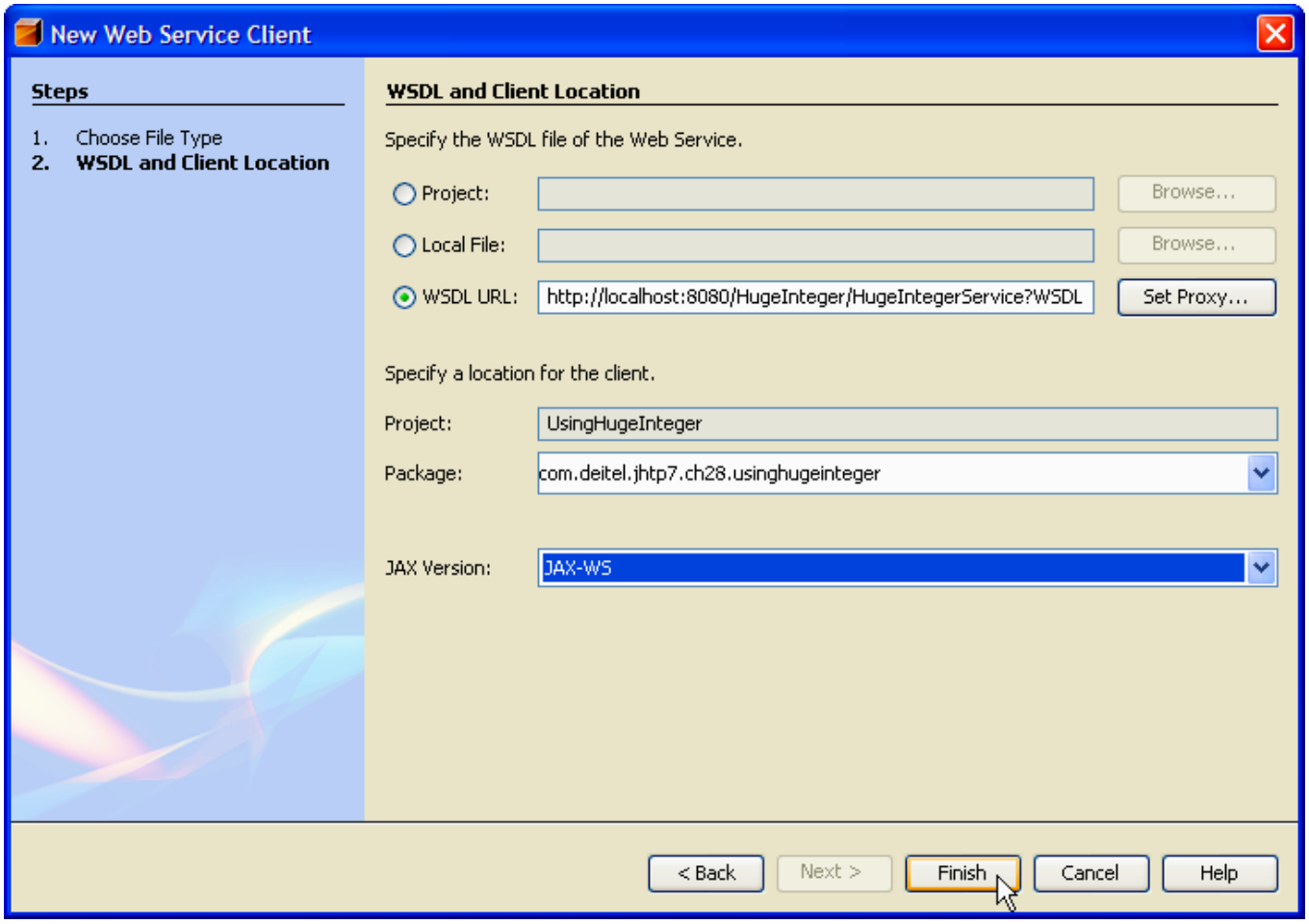


Fig. 28.7 | New Web Service Client dialog.

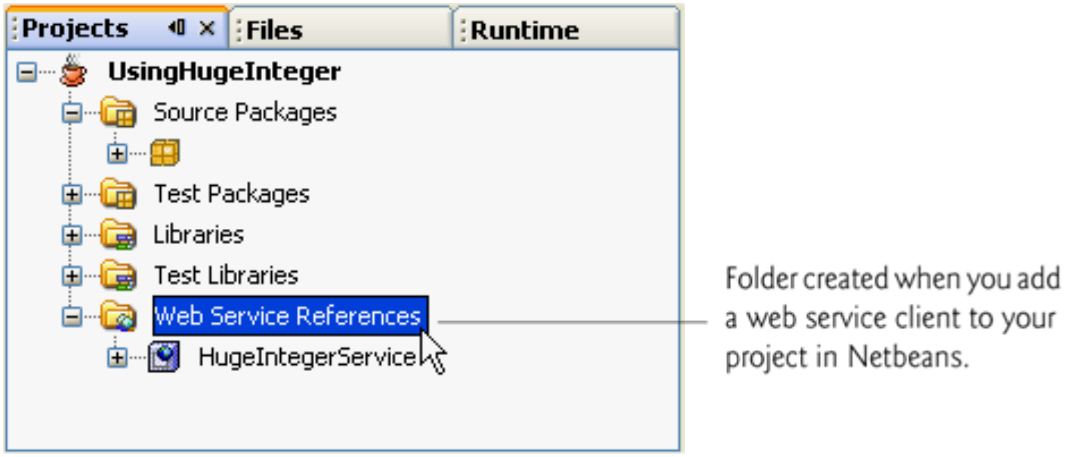


Fig. 28.8 | Netbeans **Project** tab after adding a web service reference to the project.

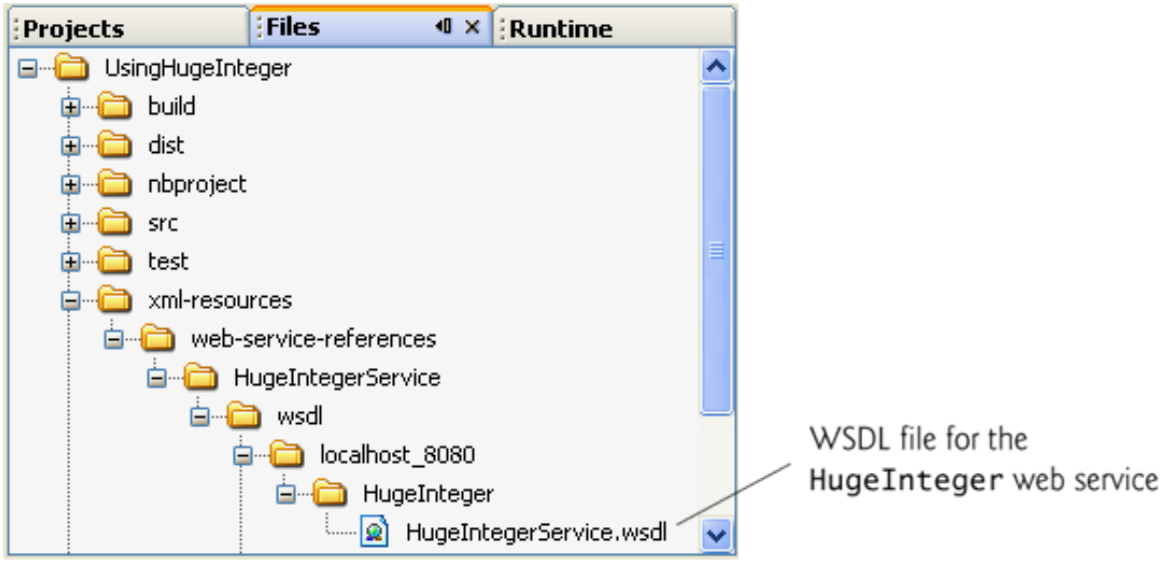


Fig. 28.9 | Locating the `HugeIntegerService.wsdl` file in the Netbeans Files tab.

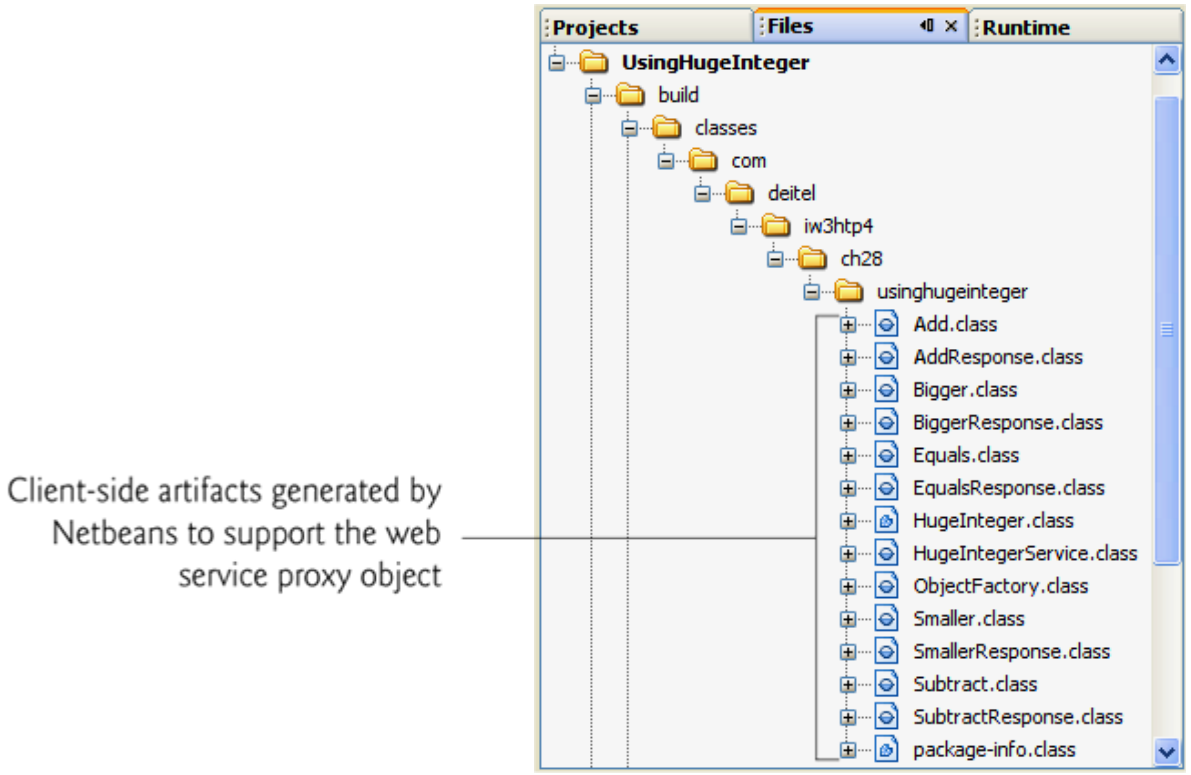


Fig. 28.10 | Viewing the `HugeInteger` web service's client-side artifacts generated by Netbeans.

28.4.2 Consuming the `HugeInteger` Web Service

- **Create an object of the web service type.**
- **Use this object's `get`*WebService*`Port` method to obtain the proxy object that the application uses to invoke the web service's methods.**

Outline

UsingHugeIntegerJFrame.java

(1 of 10)

```
1  // Fig. 28.11: UsingHugeIntegerJFrame.java
2  // Client desktop application for the HugeInteger web service.
3  package com.deitel.iw3htp4.ch28.hugeintegerclient;
4
5  // import classes for accessing HugeInteger web service's proxy
6  import com.deitel.iw3htp4.ch28.hugeintegerclient.HugeInteger;
7  import com.deitel.iw3htp4.ch28.hugeintegerclient.HugeIntegerService;
8
9  import javax.swing.JOptionPane; // used to display errors to the user
10
11 public class UsingHugeIntegerJFrame extends javax.swing.JFrame
12 {
13    private HugeIntegerService hugeIntegerService; // used to obtain proxy
14    private HugeInteger hugeIntegerProxy; // used to access the web service
15
16    // no-argument constructor
17    public UsingHugeIntegerJFrame()
18    {
19       initComponents();
20
21       try
22       {
23          // create the objects for accessing the HugeInteger web service
24          hugeIntegerService = new HugeIntegerService();
25          hugeIntegerProxy = hugeIntegerService.getHugeIntegerPort();
26       }
```

Declare variables used to obtain and access the proxy object

Obtain the proxy object

Outline

UsingHugeIntegerJFrame.java

(2 of 10)

```
27       catch ( Exception exception )
28       {
29          exception.printStackTrace();
30       }
31    } // end UsingHugeIntegerJFrame constructor
32
33    // The initComponents method is autogenerated by Netbeans and is called
34    // from the constructor to initialize the GUI. This method is not shown
35    // here to save space. Open UsingHugeIntegerJFrame.java in this
36    // example's folder to view the complete generated code (lines 37-153).
37
154   // invokes HugeInteger web service's add method to add HugeIntegers
155   private void addJButtonActionPerformed(
156      java.awt.event.ActionEvent evt )
157   {
158      String firstNumber = firstJTextField.getText();
159      String secondNumber = secondJTextField.getText();
160
161      if ( isValid( firstNumber ) && isValid( secondNumber ) )
162      {
163         try
164         {
165            resultsJTextArea.setText(
166               hugeIntegerProxy.add( firstNumber, secondNumber ) );
167         } // end try
```

Use the proxy to invoke web method `add`

Outline

UsingHugeInteger JFrame.java

(3 of 10)

```
168         catch ( Exception e )
169         {
170            JOptionPane.showMessageDialog( this, e.toString(),
171               "Add method failed", JOptionPane.ERROR_MESSAGE );
172            e.printStackTrace();
173         } // end catch
174      } // end if
175   } // end method addJButtonActionPerformed
176
177   // invokes HugeInteger web service's subtract method to subtract the
178   // second HugeInteger from the first
179   private void subtractJButtonActionPerformed(
180      java.awt.event.ActionEvent evt )
181   {
182      String firstNumber = firstJTextField.getText();
183      String secondNumber = secondJTextField.getText();
184
185      if ( isValid( firstNumber ) && isValid( secondNumber ) )
186      {
187         try
188         {
189            resultsJTextArea.setText(
190               hugeIntegerProxy.subtract( firstNumber, secondNumber ) );
191         } // end try
```

Use the proxy to invoke web method `subtract`

Outline

UsingHugeInteger JFrame.java

(4 of 10)

```
192         catch ( Exception e )
193         {
194            JOptionPane.showMessageDialog( this, e.toString(),
195               "Subtract method failed", JOptionPane.ERROR_MESSAGE );
196            e.printStackTrace();
197         } // end catch
198      } // end if
199   } // end method subtractJButtonActionPerformed
200
201   // invokes HugeInteger web service's bigger method to determine whether
202   // the first HugeInteger is greater than the second
203   private void biggerJButtonActionPerformed(
204      java.awt.event.ActionEvent evt )
205   {
206      String firstNumber = firstJTextField.getText();
207      String secondNumber = secondJTextField.getText();
208
209      if ( isValid( firstNumber ) && isValid( secondNumber ) )
210      {
211         try
212         {
213            boolean result =
214               hugeIntegerProxy.bigger( firstNumber, secondNumber );
215            resultsJTextArea.setText( String.format( "%s %s %s %s",
216               firstNumber, ( result ? "is" : "is not" ), "greater than",
217               secondNumber ) );
218         } // end try
```

Use the proxy to invoke web method `bigger`

Outline

UsingHugeInteger JFrame.java

(5 of 10)

```
219         catch ( Exception e )
220         {
221            JOptionPane.showMessageDialog( this, e.toString(),
222               "Bigger method failed", JOptionPane.ERROR_MESSAGE );
223            e.printStackTrace();
224         } // end catch
225      } // end if
226   } // end method biggerJButtonActionPerformed
227
228   // invokes HugeInteger web service's smaller method to determine
229   // whether the first HugeInteger is less than the second
230   private void smallerJButtonActionPerformed(
231      java.awt.event.ActionEvent evt )
232   {
233      String firstNumber = firstJTextField.getText();
234      String secondNumber = secondJTextField.getText();
235
236      if ( isValid( firstNumber ) && isValid( secondNumber ) )
237      {
238         try
239         {
240            boolean result =
241               hugeIntegerProxy.smaller( firstNumber, secondNumber );
242            resultsJTextArea.setText( String.format( "%s %s %s %s",
243               firstNumber, ( result ? "is" : "is not" ), "less than",
244               secondNumber ) );
245         } // end try
```

Use the proxy to invoke web method `smaller`

Outline

UsingHugeInteger JFrame.java

(6 of 10)

```
246         catch ( Exception e )
247         {
248            JOptionPane.showMessageDialog( this, e.toString(),
249               "Smaller method failed", JOptionPane.ERROR_MESSAGE );
250            e.printStackTrace();
251         } // end catch
252      } // end if
253   } // end method smallerJButtonActionPerformed
254
255   // invokes HugeInteger web service's equals method to determine whether
256   // the first HugeInteger is equal to the second
257   private void equalsJButtonActionPerformed(
258      java.awt.event.ActionEvent evt )
259   {
260      String firstNumber = firstJTextField.getText();
261      String secondNumber = secondJTextField.getText();
262
263      if ( isValid( firstNumber ) && isValid( secondNumber ) )
264      {
265         try
266         {
267            boolean result =
268               hugeIntegerProxy.equals( firstNumber, secondNumber );
269            resultsJTextArea.setText( String.format( "%s %s %s %s",
270               firstNumber, ( result ? "is" : "is not" ), "equal to",
271               secondNumber ) );
272         } // end try
```

Use the proxy to invoke web method `equals`

Outline

UsingHugeInteger JFrame.java

(7 of 10)

```
273         catch ( Exception e )
274         {
275            JOptionPane.showMessageDialog( this, e.toString(),
276               "Equals method failed", JOptionPane.ERROR_MESSAGE );
277            e.printStackTrace();
278         } // end catch
279      } // end if
280   } // end method equalsJButtonActionPerformed
281
282   // checks the size of a String to ensure that it is not too big
283   // to be used as a HugeInteger; ensure only digits in String
284   private boolean isValid( String number )
285   {
286      // check String's length
287      if ( number.length() > 100 )
288      {
289         JOptionPane.showMessageDialog( this,
290            "HugeIntegers must be <= 100 digits.", "HugeInteger Overflow",
291            JOptionPane.ERROR_MESSAGE );
292         return false;
293      } // end if
294
```

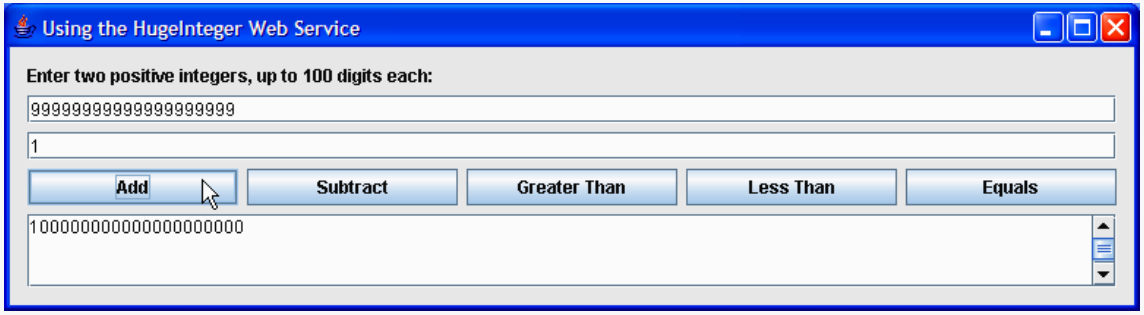
```
295       // look for nondigit characters in String
296       for ( char c : number.toCharArray() )
297       {
298          if ( !Character.isDigit( c ) )
299          {
300             JOptionPane.showMessageDialog( this,
301                "There are nondigits in the String",
302                "HugeInteger Contains Nondigit Characters",
303                JOptionPane.ERROR_MESSAGE );
304             return false;
305          } // end if
306       } // end for
307
308       return true; // number can be used as a HugeInteger
309    } // end method validate
310
311    // main method begins execution
312    public static void main( String args[] )
313    {
314       java.awt.EventQueue.invokeLater(
315          new Runnable()
316          {
317             public void run()
318             {
319                new UsingHugeIntegerJFrame().setVisible( true );
320             } // end method run
321          } // end anonymous inner class
322       ); // end call to java.awt.EventQueue.invokeLater
323    } // end method main
324
```

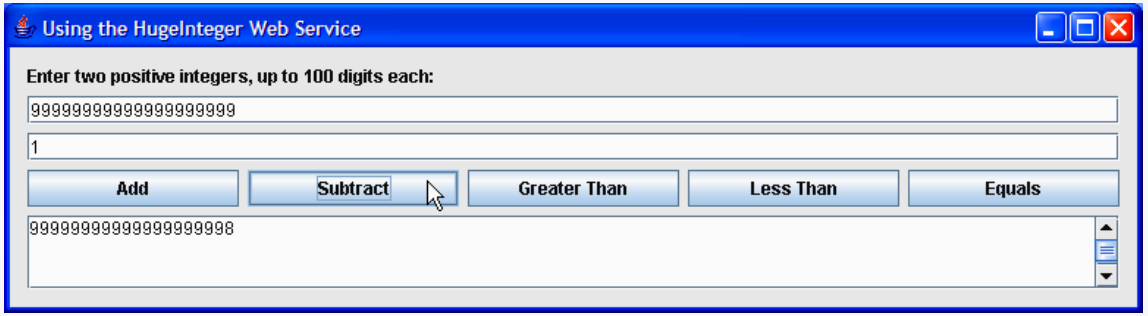
Outline

UsingHugeIntegerJFrame.java

(8 of 10)

```
325    // Variables declaration - do not modify
326    private javax.swing.JButton addJButton;
327    private javax.swing.JButton biggerJButton;
328    private javax.swing.JLabel directionsJLabel;
329    private javax.swing.JButton equalsJButton;
330    private javax.swing.JTextField firstJTextField;
331    private javax.swing.JScrollPane resultsJScrollPane;
332    private javax.swing.JTextArea resultsJTextArea;
333    private javax.swing.JTextField secondJTextField;
334    private javax.swing.JButton smallerJButton;
335    private javax.swing.JButton subtractJButton;
336    // End of variables declaration
337} // end class UsingHugeIntegerJFrame
```

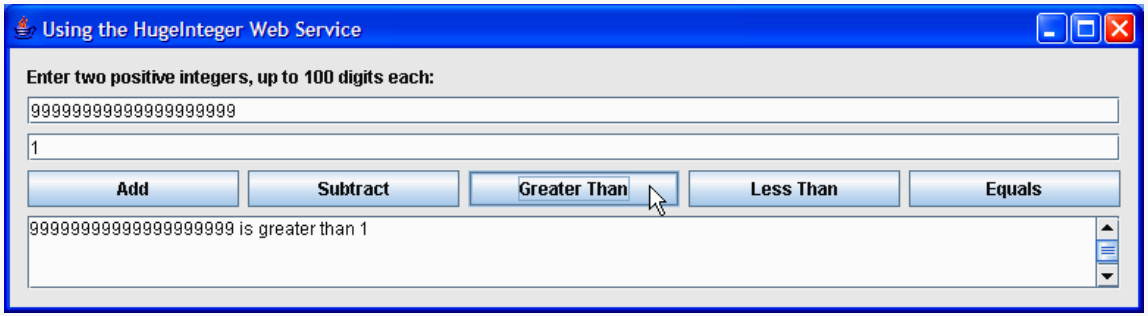


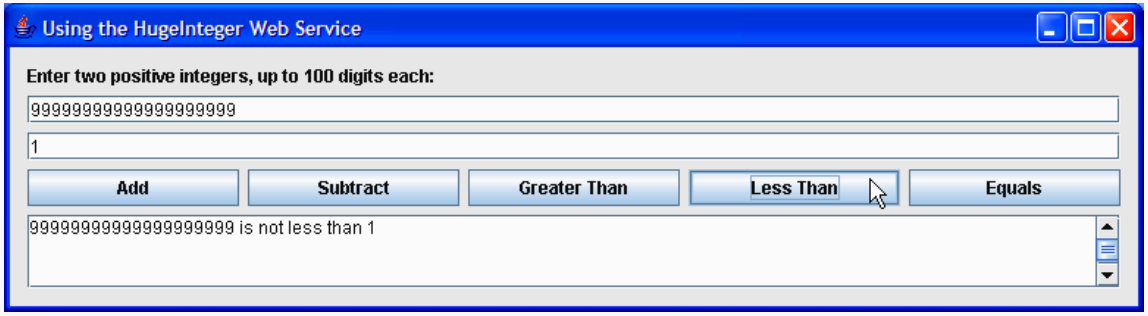


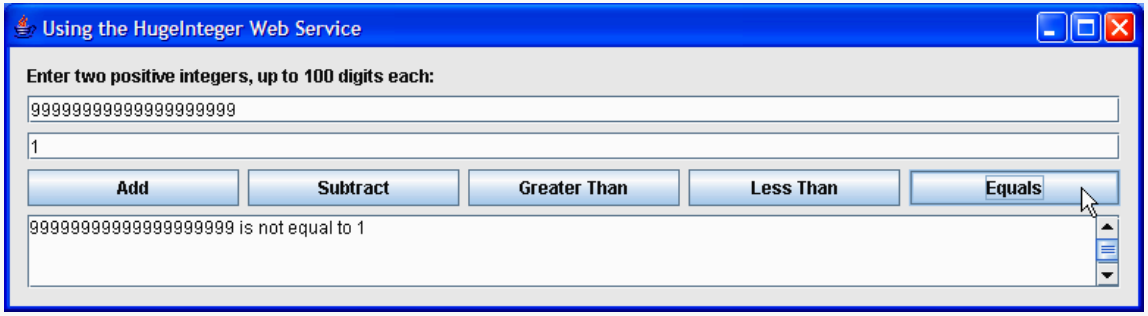
Outline

UsingHugeInteger JFrame.java

(9 of 10)







Outline

`UsingHugeInteger JFrame.java`

(10 of 10)

28.5 SOAP

- **SOAP (Simple Object Access Protocol)**
  - **Commonly used, platform-independent, XML-based protocol that facilitates remote procedure calls, typically over HTTP**
- **Wire format or wire protocol**
  - **Protocol that transmits request-and-response messages**
  - **Defines how information is sent “along the wire”**
- **SOAP message (also known as a SOAP envelope)**
  - **Each request and response is packaged in a SOAP message**
  - **Contains information that a web service requires to process the message**

28.5 SOAP (Cont.)

- Wire format must support all types passed between the applications
- SOAP supports
  - Primitive types and their wrapper types
  - Date, Time and others
  - Can also transmit arrays and objects of user-defined types
- Request SOAP message's contents
  - Method to invoke
  - Method's arguments
- Response SOAP message's contents
  - Result of method call
  - Client-side proxy parses the response and returns the result to the client application
- SOAP messages are generated for you automatically

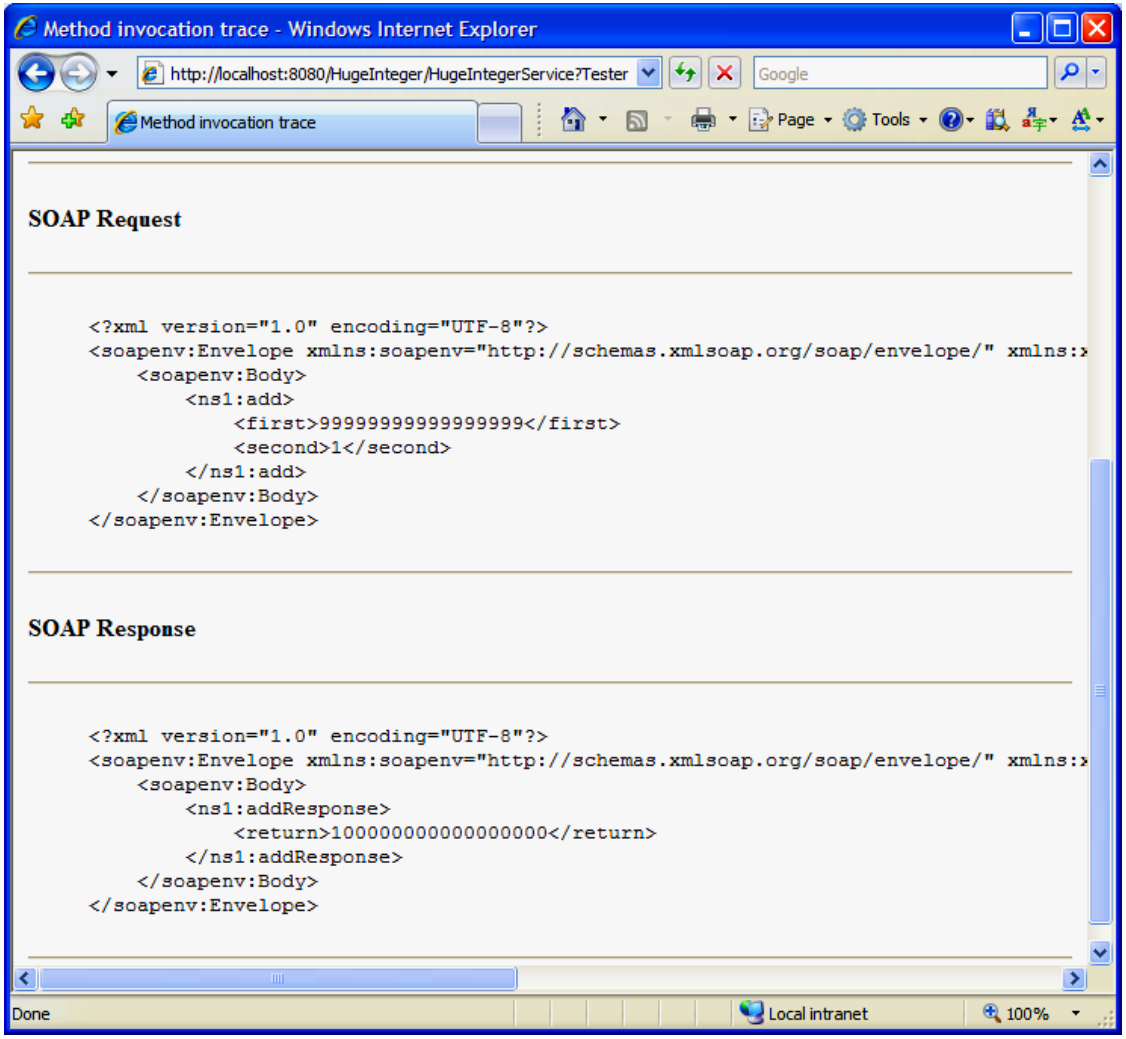


Fig. 28.12 | SOAP messages for the `HugeInteger` web service's `add` method as shown by the Sun Java System Application Server's `Tester` web page.

28.6 Session Tracking in Web Services

- **It can be beneficial for a web service to maintain client state information**
  - **Eliminates the need to pass client information between the client and the web service multiple times**
  - **Enables a web service to distinguish between clients**

28.6.1 Creating a Blackjack Web Service

- To use session tracking in a Web service
  - Must include code for the resources that maintain the session state information
  - JAX-WS handles this for you via the `@Resource` annotation
  - Enables tools like Netbeans to "inject" complex support code into your class
  - You focus on business logic rather than support code
- Using annotations to add code is known as dependency injection
- Annotations like `@WebService`, `@WebMethod` and `@WebParam` also perform dependency injection

28.6.1 Creating a Blackjack Web Service

- `WebServiceContext` object
  - Enables a web service to access and maintain information for a specific request, such as session state
  - `@Resource` annotation injects the code that creates a `WebServiceContext` object
- `MessageContext` object
  - Obtained from `WebServiceContext` object
  - `MessageContext` object's `get` method returns `HttpSession` object for the current client
    - Receives a constant indicating what to get from the `MessageContext`
    - `MessageContext.SERVLET_REQUEST` indicates that we'd like to get the `HttpServletRequest` object
    - Can then call `HttpServletRequest` method `getSession` to get the `HttpSession` object
- `HttpSession` method `getAttribute`
  - Receives a `String` that identifies the `Object` to obtain from the session state

Outline

Blackjack.java

(1 of 4)

```
1  // Fig. 28.13: Blackjack.java
2  // Blackjack web service that deals cards and evaluates hands
3  package com.deitel.iw3htp4.ch28.blackjack;
4
5  import java.util.ArrayList;
6  import java.util.Random;
7  import javax.annotation.Resource;
8  import javax.jws.WebService;
9  import javax.jws.WebMethod;
10 import javax.jws.WebParam;
11 import javax.servlet.http.HttpSession;
12 import javax.servlet.http.HttpServletRequest;
13 import javax.xml.ws.WebServiceContext;
14 import javax.xml.ws.handler.MessageContext;
15
16 @WebService( name = "Blackjack", serviceName = "BlackjackService" )
17 public class Blackjack
18 {
19    // use @Resource to create a WebServiceContext for session tracking
20    private @Resource WebServiceContext webServiceContext;
21    private MessageContext messageContext; // used in session tracking
22    private HttpSession session; // stores attributes of the session
23
24    // deal one card
25    @WebMethod( operationName = "dealCard" )
26    public String dealCard()
27    {
28       String card = "";
29
```

Import classes used for session handling

Inject code to create the `WebServiceContext` object

```
30       ArrayList< String > deck =
31          ( ArrayList< String > ) session.getAttribute( "deck" );
32
33       card = deck.get( 0 ); // get top card of deck
34       deck.remove( 0 ); // remove top card of deck
35
36       return card;
37    } // end WebMethod dealCard
38
39    // shuffle the deck
40    @WebMethod( operationName = "shuffle" )
41    public void shuffle()
42    {
43       // obtain the HttpSession object to store deck for current client
44       messageContext = webServiceContext.getMessageContext();
45       session = ( ( HttpServletRequest ) messageContext.get(
46          MessageContext.SERVLET_REQUEST ) ).getSession();
47
48       // populate deck of cards
49       ArrayList< String > deck = new ArrayList< String >();
50
51       for ( int face = 1; face <= 13; face++ ) // loop through faces
52          for ( int suit = 0; suit <= 3; suit++ ) // loop through suits
53             deck.add( face + " " + suit ); // add each card to deck
54
55       String tempCard; // holds card temporarily durring swapping
56       Random randomObject = new Random(); // generates random numbers
57       int index; // index of randomly selected card
58
```

Get an `ArrayList` of `String`s representing the current client's deck from the `session` object

(2 of 4)

Get the `MessageContext` and use it to obtain the `HttpSession` object for the current client

Outline

Blackjack.java

(3 of 4)

```
59       for ( int i = 0; i < deck.size() ; i++ ) // shuffle
60       {
61          index = randomObject.nextInt( deck.size() - 1 );
62
63          // swap card at position i with randomly selected card
64          tempCard = deck.get( i );
65          deck.set( i, deck.get( index ) );
66          deck.set( index, tempCard );
67       } // end for
68
69       // add this deck to user's session
70       session.setAttribute( "deck", deck );
71    } // end WebMethod shuffle
72
73    // determine a hand's value
74    @WebMethod( operationName = "getHandValue" )
75    public int getHandValue( @WebParam( name = "hand" ) String hand )
76    {
77       // split hand into cards
78       String[] cards = hand.split( "\t" );
79       int total = 0; // total value of cards in hand
80       int face; // face of current card
81       int aceCount = 0; // number of aces in hand
82
83       for ( int i = 0; i < cards.length; i++ )
84       {
85          // parse string and get first int in String
86          face = Integer.parseInt(
87             cards[ i ].substring( 0, cards[ i ].indexOf( " " ) ) );
88
```

Place the deck in the `session` object

Outline

Blackjack.java

(4 of 4)

```
89          switch ( face )
90          {
91             case 1: // in ace, increment aceCount
92                ++aceCount;
93                break;
94             case 11: // jack
95             case 12: // queen
96             case 13: // king
97                total += 10;
98                break;
99             default: // otherwise, add face
100               total += face;
101               break;
102         } // end switch
103      } // end for
104
105      // calculate optimal use of aces
106      if ( aceCount > 0 )
107      {
108         // if possible, count one ace as 11
109         if ( total + 11 + aceCount - 1 <= 21 )
110            total += 11 + aceCount - 1;
111         else // otherwise, count all aces as 1
112            total += aceCount;
113       } // end if
114
115      return total;
116   } // end WebMethod getHandValue
117} // end class Blackjack
```

28.6.2 Consuming the Blackjack Web Service

- **In JAX-WS 2.0**
  - **Client must indicate whether it wants to allow the web service to maintain session information**
  - **Cast the proxy object to interface type `BindingProvider`**
    - **Enables the client to manipulate the request information that will be sent to the server**
    - **Information is stored in an object that implements interface RequestContext**
    - **`BindingProvider` and `RequestContext` are created by the IDE when you add a web service client to the application**
  - **Invoke the `BindingProvider`'s `getRequestContext` method to obtain the `RequestContext` object**
  - **Call the `RequestContext`'s `put` method to set the property `BindingProvider.SESSION_MAINTAIN_PROPERTY` to `true`**
    - **Enables session tracking from the client side so that the web service knows which client is invoking the service's web methods**

Outline

BlackjackGameJFrame.java

(1 of 18)

```
1  // Fig. 28.14: BlackjackGameJFrame.java
2  // Blackjack game that uses the Blackjack Web Service
3  package com.deitel.iw3htp4.ch28.blackjackclient;
4
5  import java.awt.Color;
6  import java.util.ArrayList;
7  import javax.swing.ImageIcon;
8  import javax.swing.JLabel;
9  import javax.swing.JOptionPane;
10 import javax.xml.ws.BindingProvider;
11 import com.deitel.iw3htp4.ch28.blackjackclien
12 import com.deitel.iw3htp4.ch28.blackjackclien
13
14 public class BlackjackGameJFrame extends javax.swing.JFrame
15 {
16    private String playerCards;
17    private String dealerCards;
18    private ArrayList< JLabel > cardboxes; // list of card image JLabels
19    private int currentPlayerCard; // player's current card number
20    private int currentDealerCard; // blackjackProxy's current card number
21    private BlackjackService blackjackService; // used to obtain proxy
22    private Blackjack blackjackProxy; // used to access the web service
23
```

Used to enable session tracking from the client application

Outline

BlackjackGameJFrame.java

(2 of 18)

```
24    // enumeration of game states
25    private enum GameStatus
26    {
27       PUSH, // game ends in a tie
28       LOSE, // player loses
29       WIN, // player wins
30       BLACKJACK // player has blackjack
31    } // end enum GameStatus
32
33    // no-argument constructor
34    public BlackjackGameJFrame()
35    {
36       initComponents();
37
38       // due to a bug in Netbeans, we must change the JFrame's background
39       // color here rather than in the designer
40       getContentPane().setBackground( new Color( 0, 180, 0 ) );
41
42       // initialize the blackjack proxy
43       try
44       {
45          // create the objects for accessing the Blackjack web service
46          blackjackService = new BlackjackService();
47          blackjackProxy = blackjackService.getBlackjackPort();
48
49          // enable session tracking
50          ( ( BindingProvider ) blackjackProxy ).getRequestContext().put(
51             BindingProvider.SESSION_MAINTAIN_PROPERTY, true );
52       } // end try
53       catch ( Exception e )
```

Enable session tracking for the client

```
54       {
55          e.printStackTrace();
56       } // end catch
57
58       // add JLabels to cardBoxes ArrayList for programmatic manipulation
59       cardboxes = new ArrayList< JLabel >();
60
61       cardboxes.add( 0, dealerCard1JLabel );
62       cardboxes.add( dealerCard2JLabel );
63       cardboxes.add( dealerCard3JLabel );
64       cardboxes.add( dealerCard4JLabel );
65       cardboxes.add( dealerCard5JLabel );
66       cardboxes.add( dealerCard6JLabel );
67       cardboxes.add( dealerCard7JLabel );
68       cardboxes.add( dealerCard8JLabel );
69       cardboxes.add( dealerCard9JLabel );
70       cardboxes.add( dealerCard10JLabel );
71       cardboxes.add( dealerCard11JLabel );
72       cardboxes.add( playerCard1JLabel );
73       cardboxes.add( playerCard2JLabel );
74       cardboxes.add( playerCard3JLabel );
75       cardboxes.add( playerCard4JLabel );
76       cardboxes.add( playerCard5JLabel );
77       cardboxes.add( playerCard6JLabel );
78       cardboxes.add( playerCard7JLabel );
79       cardboxes.add( playerCard8JLabel );
80       cardboxes.add( playerCard9JLabel );
81       cardboxes.add( playerCard10JLabel );
82       cardboxes.add( playerCard11JLabel );
83    } // end no-argument constructor
```

Outline

BlackjackGameJFrame.java

(3 of 18)

Outline

BlackjackGameJFrame.java

(4 of 18)

```
84
85    // play the dealer's hand
86    private void dealerPlay()
87    {
88       try
89       {
90          // while the value of the dealers's hand is below 17
91          // the dealer must continue to take cards
92          String[] cards = dealerCards.split( "\t" );
93
94          // display dealers's cards
95          for ( int i = 0; i < cards.length; i++ )
96             displayCard( i, cards[ i ]);
97
98          while ( blackjackProxy.getHandValue( dealerCards ) < 17 )
99          {
100            String newCard = blackjackProxy.dealCard();
101            dealerCards += "\t" + newCard; // deal new card
102            displayCard( currentDealerCard, newCard );
103            ++currentDealerCard;
104            JOptionPane.showMessageDialog( this, "Dealer takes a card",
105               "Dealer's turn", JOptionPane.PLAIN_MESSAGE );
106         } // end while
```

Outline

BlackjackGameJFrame.java

(5 of 18)

```
107
108       int dealersTotal = blackjackProxy.getHandValue( dealerCards );
109       int playersTotal = blackjackProxy.getHandValue( playerCards );
110
111       // if dealer busted, player wins
112       if ( dealersTotal > 21 )
113       {
114          gameOver( GameStatus.WIN );
115          return;
116       } // end if
117
118       // if dealer and player are below 21
119       // higher score wins, equal scores is a push
120       if ( dealersTotal > playersTotal )
121          gameOver( GameStatus.LOSE );
122       else if ( dealersTotal < playersTotal )
123          gameOver( GameStatus.WIN );
124       else
125          gameOver( GameStatus.PUSH );
126    } // end try
127    catch ( Exception e )
128    {
129       e.printStackTrace();
130    } // end catch
131 } // end method dealerPlay
132
```

```
133   // displays the card represented by cardValue in specified JLabel
134   public void displayCard( int card, String cardValue )
135   {
136      try
137      {
138         // retrieve correct JLabel from cardBoxes
139         JLabel displayLabel = cardboxes.get( card );
140
141         // if string representing card is empty, display back of card
142         if ( cardValue.equals( "" ) )
143         {
144             displayLabel.setIcon( new ImageIcon( getClass().getResource(
145                "/com/deitel/jhtp7/ch28/blackjackclient/" +
146                "blackjack_images/cardback.png" ) ) ) ;
147             return;
148         } // end if
149
150         // retrieve the face value of the card
151         String face = cardValue.substring( 0, cardValue.indexOf( " " ) );
152
```

Outline

BlackjackGameJFrame.java

(6 of 18)

Outline

BlackjackGameJFrame.java

(7 of 18)

```
153        // retrieve the suit of the card
154        String suit =
155           cardValue.substring( cardValue. indexOf( " " ) + 1 );
156
157        char suitLetter; // suit letter used to form image file
158
159        switch ( Integer.parseInt( suit ) )
160        {
161           case 0: // hearts
162              suitLetter = 'h';
163              break;
164           case 1: // diamonds
165              suitLetter = 'd';
166              break;
167           case 2: // clubs
168              suitLetter = 'c';
169              break;
170           default: // spades
171              suitLetter = 's';
172              break;
173        } // end switch
174
175        // set image for displayLabel
176        displayLabel.setIcon( new ImageIcon( getClass().getResource(
177           "/com/deitel/jhtp7/ch28/blackjackclient/blackjack_images/" +
178           face + suitLetter + ".png" ) ) );
179     } // end try
```

Outline

BlackjackGameJFrame.java

(8 of 18)

```
180      catch ( Exception e )
181      {
182         e.printStackTrace();
183      } // end catch
184   } // end method displayCard
185
186   // displays all player cards and shows appropriate message
187   public void gameOver( GameStatus winner )
188   {
189      String[] cards = dealerCards.split( "\t" );
190
191      // display blackjackProxy's cards
192      for ( int i = 0; i < cards.length; i++ )
193         displayCard( i, cards[ i ]);
194
195      // display appropriate status image
196      if ( winner == GameStatus.WIN )
197         statusJLabel.setText( "You win!" );
198      else if ( winner == GameStatus.LOSE )
199         statusJLabel.setText( "You lose." );
200      else if ( winner == GameStatus.PUSH )
201         statusJLabel.setText( "It's a push." );
202      else // blackjack
203         statusJLabel.setText( "Blackjack!" );
204
205      // display final scores
206      int dealersTotal = blackjackProxy.getHandValue( dealerCards );
207      int playersTotal = blackjackProxy.getHandValue( playerCards );
208      dealerTotalJLabel.setText( "Dealer: " +  dealersTotal );
209      playerTotalJLabel.setText( "Player: " + playersTotal );
```

Outline

BlackjackGameJFrame.java

(9 of 18)

```
210
211       // reset for new game
212       standJButton.setEnabled( false );
213       hitJButton.setEnabled( false );
214       dealJButton.setEnabled( true );
215    } // end method gameOver
216
217    // The initComponents method is autogenerated by Netbeans and is called
218    // from the constructor to initialize the GUI. This method is not shown
219    // here to save space. Open BlackjackGameJFrame.java in this
220    // example's folder to view the complete generated code (lines 221-531)
221
532    // handles standJButton click
533    private void standJButtonActionPerformed(
534       java.awt.event.ActionEvent evt )
535    {
536       standJButton.setEnabled( false );
537       hitJButton.setEnabled( false );
538       dealJButton.setEnabled( true );
539       dealerPlay();
540    } // end method standJButtonActionPerformed
541
542    // handles hitJButton click
543    private void hitJButtonActionPerformed(
544       java.awt.event.ActionEvent evt )
545    {
546       // get player another card
547       String card = blackjackProxy.dealCard(); // deal new card
548       playerCards += "\t" + card; // add card to hand
549
```

Outline

BlackjackGameJFrame.java

(10 of 18)

```
550      // update GUI to display new card
551      displayCard( currentPlayerCard, card );
552      ++currentPlayerCard;
553
554      // determine new value of player's hand
555      int total = blackjackProxy.getHandValue( playerCards );
556
557      if ( total > 21 ) // player busts
558         gameOver( GameStatus.LOSE );
559      if ( total == 21 ) // player cannot take any more cards
560      {
561         hitJButton.setEnabled( false );
562         dealerPlay();
563      } // end if
564   } // end method hitJButtonActionPerformed
565
566   // handles dealJButton click
567   private void dealJButtonActionPerformed(
568      java.awt.event.ActionEvent evt )
569   {
570      String card; // stores a card temporarily until it's added to a hand
571
572      // clear card images
573      for ( int i = 0; i < cardboxes.size(); i++ )
574         cardboxes.get( i ).setIcon( null );
575
576      statusJLabel.setText( "" );
577      dealerTotalJLabel.setText( "" );
578      playerTotalJLabel.setText( "" );
579
```

Outline

BlackjackGameJFrame.java

(11 of 18)

```
580      // create a new, shuffled deck on remote machine
581      blackjackProxy.shuffle();
582
583      // deal two cards to player
584      playerCards = blackjackProxy.dealCard(); // add first card to hand
585      displayCard( 11, playerCards ); // display first card
586      card = blackjackProxy.dealCard(); // deal second card
587      displayCard( 12, card ); // display second card
588      playerCards += "\t" + card; // add second card to hand
589
590      // deal two cards to blackjackProxy, but only show first
591      dealerCards = blackjackProxy.dealCard(); // add first card to hand
592      displayCard( 0, dealerCards ); // display first card
593      card = blackjackProxy.dealCard(); // deal second card
594      displayCard( 1, "" ); // display back of card
595      dealerCards += "\t" + card; // add second card to hand
596
597      standJButton.setEnabled( true );
598      hitJButton.setEnabled( true );
599      dealJButton.setEnabled( false );
600
601      // determine the value of the two hands
602      int dealersTotal = blackjackProxy.getHandValue( dealerCards );
603      int playersTotal = blackjackProxy.getHandValue( playerCards );
604
605      // if hands both equal 21, it is a push
606      if ( playersTotal == dealersTotal && playersTotal == 21 )
607         gameOver( GameStatus.PUSH );
608      else if ( dealersTotal == 21 ) // blackjackProxy has blackjack
609         gameOver( GameStatus.LOSE );
```

Outline

BlackjackGameJFrame.java

(12 of 18)

```
610      else if ( playersTotal == 21 ) // blackjack
611         gameOver( GameStatus.BLACKJACK );
612
613      // next card for blackjackProxy has index 2
614      currentDealerCard = 2;
615
616      // next card for player has index 13
617      currentPlayerCard = 13;
618   } // end method dealJButtonActionPerformed
619
620   // begins application execution
621   public static void main( String args[] )
622   {
623      java.awt.EventQueue.invokeLater(
624         new Runnable()
625         {
626            public void run()
627            {
628               new BlackjackGameJFrame().setVisible(true);
629            }
630         }
631      ); // end call to java.awt.EventQueue.invokeLater
632   } // end method main
633
634   // Variables declaration - do not modify
635   private javax.swing.JButton dealJButton;
636   private javax.swing.JLabel dealerCard10JLabel;
637   private javax.swing.JLabel dealerCard11JLabel;
638   private javax.swing.JLabel dealerCard1JLabel;
639   private javax.swing.JLabel dealerCard2JLabel;
```

Outline

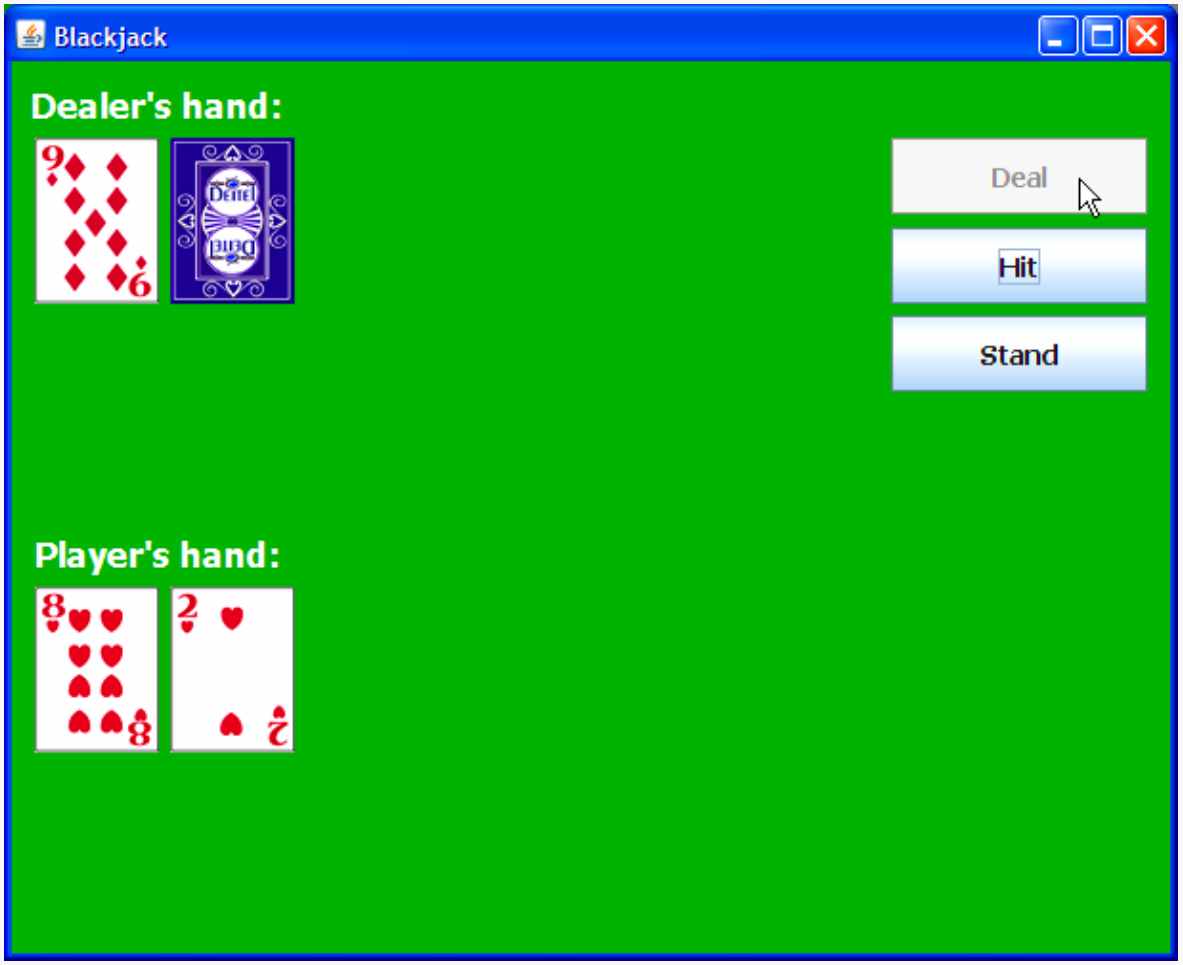
BlackjackGameJFrame.java

(13 of 18)

```
640   private javax.swing.JLabel dealerCard3JLabel;
641   private javax.swing.JLabel dealerCard4JLabel;
642   private javax.swing.JLabel dealerCard5JLabel;
643   private javax.swing.JLabel dealerCard6JLabel;
644   private javax.swing.JLabel dealerCard7JLabel;
645   private javax.swing.JLabel dealerCard8JLabel;
646   private javax.swing.JLabel dealerCard9JLabel;
647   private javax.swing.JLabel dealerJLabel;
648   private javax.swing.JLabel dealerTotalJLabel;
649   private javax.swing.JButton hitJButton;
650   private javax.swing.JLabel playerCard10JLabel;
651   private javax.swing.JLabel playerCard11JLabel;
652   private javax.swing.JLabel playerCard1JLabel;
653   private javax.swing.JLabel playerCard2JLabel;
654   private javax.swing.JLabel playerCard3JLabel;
655   private javax.swing.JLabel playerCard4JLabel;
656   private javax.swing.JLabel playerCard5JLabel;
657   private javax.swing.JLabel playerCard6JLabel;
658   private javax.swing.JLabel playerCard7JLabel;
659   private javax.swing.JLabel playerCard8JLabel;
660   private javax.swing.JLabel playerCard9JLabel;
661   private javax.swing.JLabel playerJLabel;
662   private javax.swing.JLabel playerTotalJLabel;
663   private javax.swing.JButton standJButton;
664   private javax.swing.JLabel statusJLabel;
665    // End of variables declaration
666} // end class BlackjackGameJFrame
```

Outline

a) Dealer and player hands after the user clicks the **Deal** `JButton`.

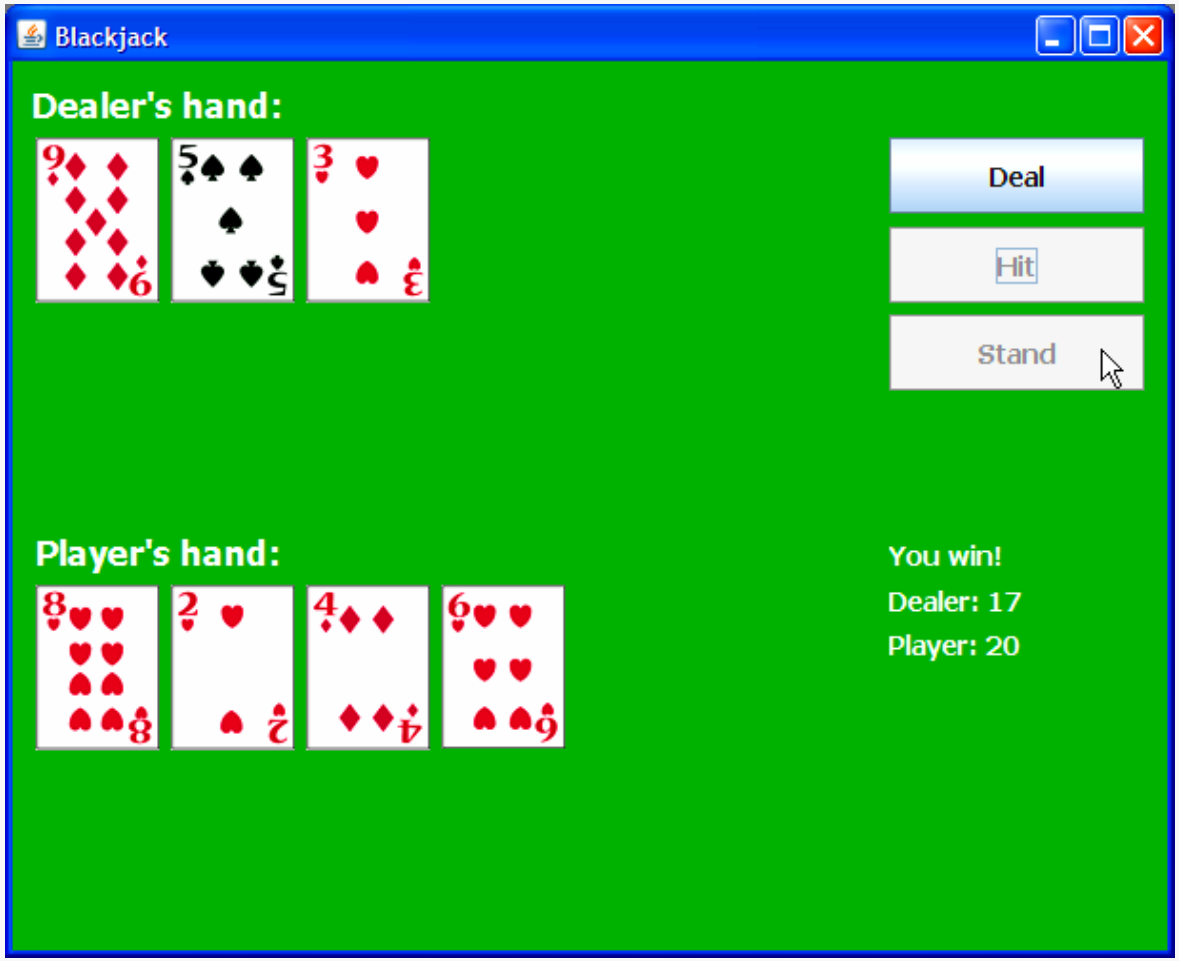


`BlackjackGameJFrame.java`

(14 of 18)

Outline

b) Dealer and player hands after the user clicks **Hit** twice, then clicks **Stand**. In this case, the player wins.



`BlackjackGameJFrame.java`

(15 of 18)

Outline

c) Dealer and player hands after the user clicks **Stand** based on the initial hand. In this case, the player loses.

`BlackjackGameJFrame.java`

(16 of 18)

Outline

d) Dealer and player hands after the user is dealt blackjack.

`BlackjackGameJFrame.java`

(17 of 18)

Outline

e) Dealer and player hands after the dealer is dealt blackjack.

`BlackjackGameJFrame.java`

(18 of 18)

28.7 Consuming a Database-Driven Web Service from a Web Application

- **Because web-based businesses are becoming increasingly prevalent, it is common for web applications to consume web services**

28.7.1 Configuring Java DB in Netbeans and Creating the Reservation Database

- To add a Java DB database server in Netbeans
  - Select **Tools > Options…** to display the Netbeans **Options** dialog
  - Click the **Advanced Options** button to display the **Advanced Options** dialog
  - Under **IDE Configuration**, expand the **Server and External Tool Settings** node and select **Java DB Database**
  - If the Java DB properties are not already configured, set the **Java DB Location** property to the location of Java DB on your system
  - Set the **Database Location** property to the location where you'd like the Java DB databases to be stored

28.7.1 Configuring Java DB in Netbeans and Creating the Reservation Database (Cont.)

- To create a new database
  - Select **Tools > Java DB Database > Create Java DB Database…**
  - Enter the name of the database to create, a username and a password
  - Click **OK** to create the database
- Can use the Netbeans **Runtime** tab to create tables and to execute SQL statements that populate the database with data
  - Click the Netbeans **Runtime** tab and expand the **Databases** node.
  - Netbeans must be connected to the database to execute SQL statements
  - If not connected, right click the icon next to the databse and click **Connect**

Number	Location	Class	Taken
1	Aisle	Economy	0
2	Aisle	Economy	0
3	Aisle	First	0
4	Middle	Economy	0
5	Middle	Economy	0
6	Middle	First	0
7	Window	Economy	0
8	Window	Economy	0
9	Window	First	0
10	Window	First	0

Fig. 28.15 | `Seats` table's data.

Software Engineering Observation 28.1

Using `PreparedStatements` to create SQL statements is highly recommended to secure against so-called SQL injection attacks in which executable code is inserted SQL code. The site `www.owasp.org/index.php/Preventing_SQL_Injection_in_Java` provides a summary of SQL injection attacks and ways to mitigate against them..

Outline

Reservation.java

(1 of 3)

```
1  // Fig. 28.16: Reservation.java
2  // Airline reservation web service.
3  package com.deitel.iw3htp4.ch28.reservation;
4
5  import java.sql.Connection;
6  import java.sql.PreparedStatement;
7  import java.sql.DriverManager;
8  import java.sql.ResultSet;
9  import java.sql.SQLException;
10 import javax.jws.WebService;
11 import javax.jws.WebMethod;
12 import javax.jws.WebParam;
13
14 @WebService( name = "Reservation", serviceName = "ReservationService" )
15 public class Reservation
16 {
17    private static final String DATABASE_URL =
18       "jdbc:derby://localhost:1527/Reservation";
19    private static final String USERNAME = "iw3htp4";
20    private static final String PASSWORD = "iw3htp4";
21    private Connection connection;
22    private PreparedStatement lookupSeat;
23    private PreparedStatement reserveSeat;
24
25    // a WebMethod that can reserve a seat
26    @WebMethod( operationName = "reserve" )
27    public boolean reserve( @WebParam( name = "seatType" ) String seatType,
28       @WebParam( name = "classType" ) String classType )
```

Import classes and interfaces used for database processing

Strings that represent the database URL, username and password

Outline

Reservation.java

(2 of 3)

```
29    {
30       try
31       {
32          connection = DriverManager.getConnection(
33             DATABASE_URL, USERNAME, PASSWORD );
34          lookupSeat = connection.prepareStatement(
35             "SELECT \"NUMBER\" FROM \"SEATS\" WHERE (\"TAK
36             "AND (\"LOCATION\" = ?) AND (\"CLASS\" = ?)" )
37          lookupSeat.setString( 1, seatType );
38          lookupSeat.setString( 2, classType );
39          ResultSet resultSet = lookupSeat.executeQuery();
40
41          // if requested seat is available, reserve it
42          if ( resultSet.next() )
43          {
44             int seat = resultSet.getInt( 1 );
45             reserveSeat = connection.prepareStatement(
46                "UPDATE \"SEATS\" SET \"TAKEN\"=1 WHERE \"NUMBER\"=?"
47             reserveSeat.setInt( 1, seat );
48             reserveSeat.executeUpdate();
49             return true;
50          } // end if
51
52          return false;
53       } // end try
```

Get a connection to the database

Create a Statement for executing queries

Execute a query that selects seats that are not taken and match the specified criteria

Update a seat as taken

Outline

Reservation.java

(3 of 3)

```
54       catch ( SQLException e )
55       {
56          e.printStackTrace();
57          return false;
58       } // end catch
59       catch ( Exception e )
60       {
61          e.printStackTrace();
62          return false;
63       } // end catch
64       finally
65       {
66          try
67          {
68             lookupSeat.close();
69             reserveSeat.close();
70             connection.close();
71          } // end try
72          catch ( Exception e )
73          {
74             e.printStackTrace();
75             return false;
76          } // end catch
77       } // end finally
78    } // end WebMethod reserve
79 } // end class Reservation
```

28.7.2 Creating a Web Application to Interact with the Reservation Web Service

Outline

Reserve.jsp

(1 of 4)

```
1  <?xml version="1.0" encoding="UTF-8"?>
2
3  <!-- Fig. 28.17 Reserve.jsp -->
4  <!-- JSP that allows a user to select a seat -->
5  <jsp:root version="1.2"
6     xmlns:f="http://java.sun.com/jsf/core"
7     xmlns:h="http://java.sun.com/jsf/html"
8     xmlns:jsp="http://java.sun.com/JSP/Page"
9     xmlns:webuijsf="http://www.sun.com/webui/webuijsf">
10    <jsp:directive.page contentType="text/html;charset=UTF-8"
11       pageEncoding="UTF-8"/>
12    <f:view>
13    <webuijsf:page binding="#{Reserve.page1}" id="page1">
14       <webuijsf:html binding="#{Reserve.html1}" id="html1">
15          <webuijsf:head binding="#{Reserve.head1}" id="head1">
16             <webuijsf:link binding="#{Reserve.link1}" id="link1"
17                url="/resources/stylesheet.css"/>
18          </webuijsf:head>
19          <webuijsf:body binding="#{Reserve.body1}" id="body1"
20             style="-rave-layout: grid">
21             <webuijsf:form binding="#{Reserve.form1}" id="form1">
22                <webuijsf:label binding="#{Reserve.instructionLabel}"
23                   id="instructionLabel" style="left: 24px; top: 24px;
24                   position: absolute" text="Please select the seat type
25                   and class to reserve:"/>
26                <webuijsf:dropDown binding="#{Reserve.seatTypeDropDown}"
27                   id="seatTypeDropDown" items=
28                   "#{Reserve.seatTypeDropDownDefaultOptions.options}"
```

Outline

Reserve.jsp

(2 of 4)

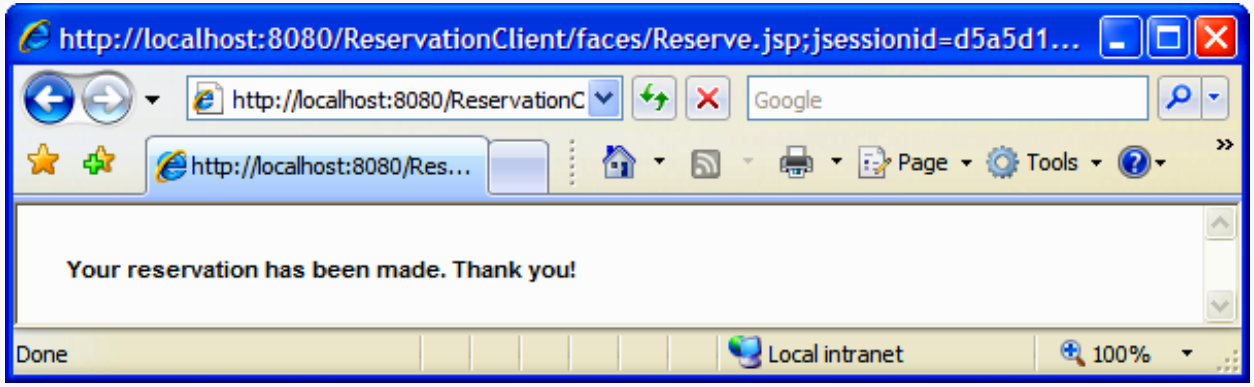
```
29                style="left: 310px; top: 21px; position: absolute"
30                valueChangeListenerExpression=
31                "#{Reserve.seatTypeDropDown_processValueChange}"/>
32             <webuijsf:dropDown binding="#{Reserve.classTypeDropDown}"
33                id="classTypeDropDown" items=
34                "#{Reserve.classTypeDropDownDefaultOptions.options}"
35                style="left: 385px; top: 21px; position: absolute"
36                valueChangeListenerExpression=
37                "#{Reserve.classTypeDropDown_processValueChange}"/>
38             <webuijsf:button actionExpression=
39                "#{Reserve.reserveButton_action}" binding=
40                "#{Reserve.reserveButton}" id="reserveButton" style=
41                "height: 20px; left: 460px; top: 21px; position:
42                absolute; width: 100px" text="Reserve"/>
43             <webuijsf:label binding="#{Reserve.errorLabel}"
44                id="errorLabel" rendered="false" style="color: red;
45                left: 24px; top: 48px; position: absolute" text="This
46                type of seat is not available. Please modify your
47                request and try again."/>
48             <webuijsf:label binding="#{Reserve.successLabel}"
49                id="successLabel" rendered="false" style="left: 24px;
50                top: 24px; position: absolute"
51                text="Your reservation has been made. Thank you!"/>
52          </webuijsf:form>
53       </webuijsf:body>
54    </webuijsf:html>
55  </webuijsf:page>
56  </f:view>
57 </jsp:root>
```

Outline

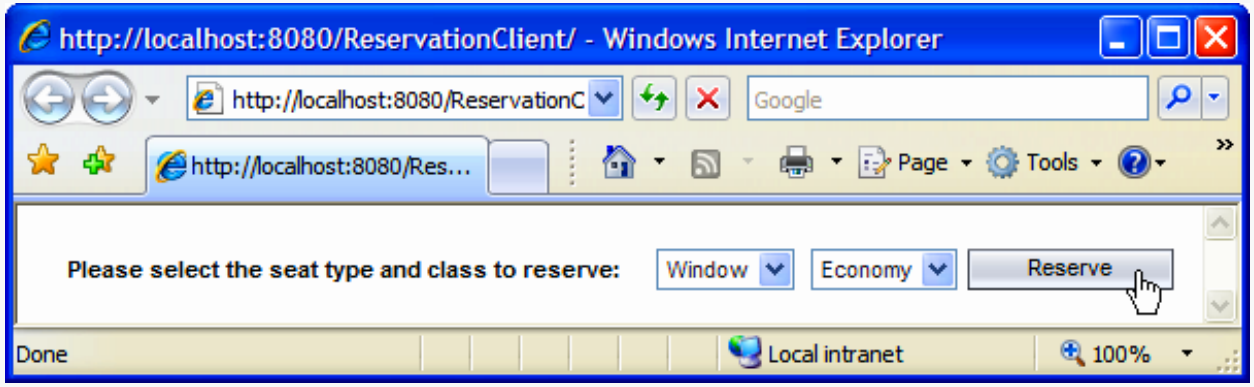
`Reserve.jsp`

(3 of 4)

a) Selecting a seat:



b) Seat reserved successfully:

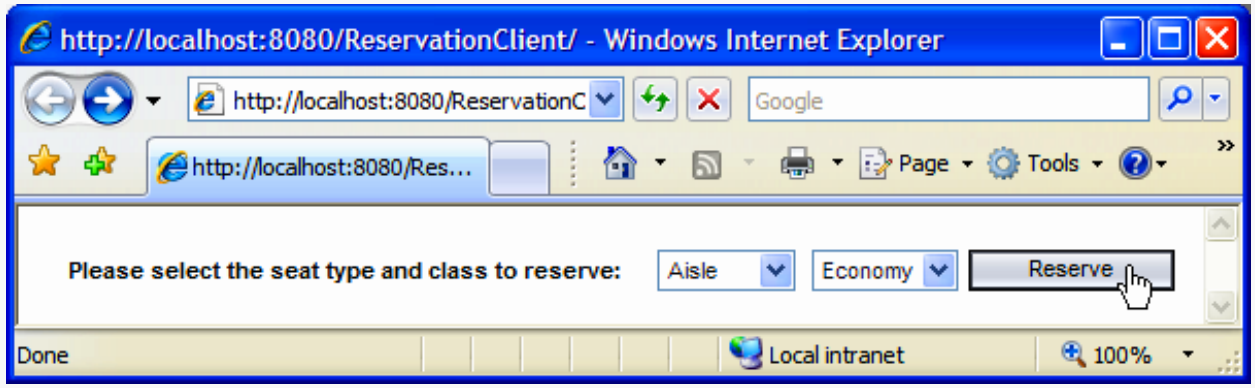


Outline

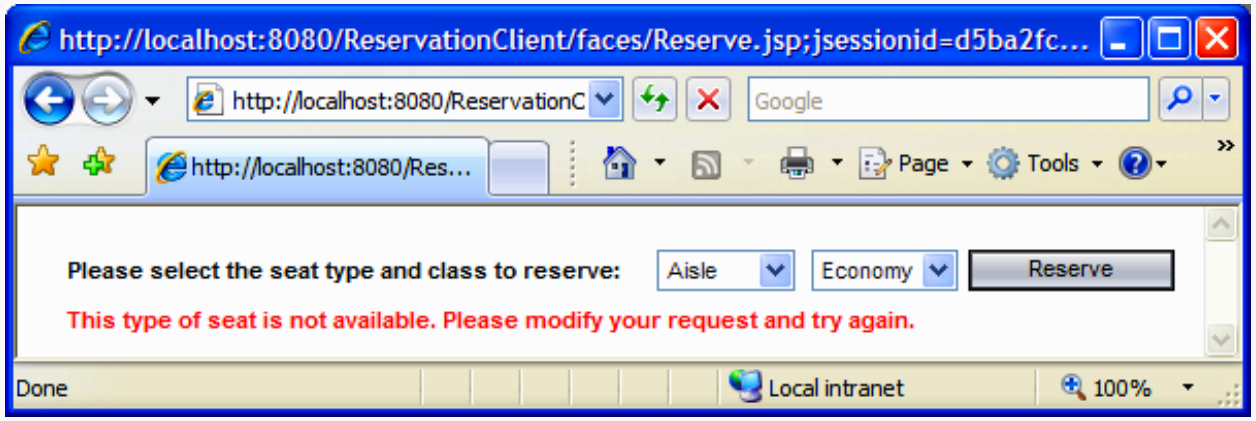
`Reserve.jsp`

(4 of 4)

c) Attempting to reserve another window seat in economy when there are no such seats available:



d) No seats match the requested seat type and class:



Outline

Reserve.java

(1 of 4)

```
1  // Fig. 28.18: Reserve.java
2  // Page scope backing bean class for seat reservation client
3  package com.deitel.iw3htp4.ch28.reservationclient;
4
5  import com.sun.rave.web.ui.appbase.AbstractPageBean;
6  import com.sun.webui.jsf.component.Body;
7  import com.sun.webui.jsf.component.Button;
8  import com.sun.webui.jsf.component.DropDown;
9  import com.sun.webui.jsf.component.Form;
10 import com.sun.webui.jsf.component.Head;
11 import com.sun.webui.jsf.component.Html;
12 import com.sun.webui.jsf.component.Label;
13 import com.sun.webui.jsf.component.Link;
14 import com.sun.webui.jsf.component.Page;
15 import com.sun.webui.jsf.model.SingleSelectOptionsList;
16 import javax.faces.FacesException;
17 import javax.faces.event.ValueChangeEvent;
18 import reservationservice.ReservationService;
19 import reservationservice.Reservation;
20
21 public class Reserve extends AbstractPageBean
22 {
23    private int __placeholder;
24    private ReservationService reservationService; // reference to service
25    private Reservation reservationServiceProxy; // reference to proxy
26
```

Required to access the `ReservationService`

Outline

Reserve.java

(2 of 4)

```
27    private void _init() throws Exception
28    {
29       seatTypeDropDownDefaultOptions.setOptions(
30          new com.sun.webui.jsf.model.Option[] {
31             new com.sun.webui.jsf.model.Option( "Aisle", "Aisle" ),
32             new com.sun.webui.jsf.model.Option( "Middle", "Middle" ),
33             new com.sun.webui.jsf.model.Option( "Window", "Window" ) } );
34       classTypeDropDownDefaultOptions.setOptions(
35          new com.sun.webui.jsf.model.Option[] {
36             new com.sun.webui.jsf.model.Option( "Economy", "Economy" ),
37             new com.sun.webui.jsf.model.Option( "First", "First" ) } );
38       reservationService = new ReservationService();
39       reservationServiceProxy = reservationService.getReservationPort();
40    } // end method
41
42    // Lines 42-260 of the autogenerated code have been removed to save
43    // space. The complete code is available in this example's folder.
44
261   // store selected class in session bean
262   public void classTypeDropDown_processValueChange(
263      ValueChangeEvent event )
264   {
265      getSessionBean1().setClassType(
266         ( String ) classTypeDropDown.getSelected() );
267   } // end method classTypeDropDown_processValueChange
268
```

Store selected class type in the session bean

Outline

```
269   // store selected seat type in session bean
270   public void seatTypeDropDown_processValueChange(
271      ValueChangeEvent event )
272   {
273      getSessionBean1().setSeatType(
274         ( String ) seatTypeDropDown.getSelected() );
275   } // end method seatTypeDropDown_processValueChange
276
277   // invoke the web service when the user clicks Reserve button
278   public String reserveButton_action()
279   {
280      try
281      {
282         boolean reserved = reservationServiceProxy.reserve(
283            getSessionBean1().getSeatType(),
284            getSessionBean1().getClassType() );
285
286         if ( reserved ) // display successLabel; hide all others
287         {
288            instructionLabel.setRendered( false );
289            seatTypeDropDown.setRendered( false );
290            classTypeDropDown.setRendered( false );
291            reserveButton.setRendered( false );
292            successLabel.setRendered( true );
293            errorLabel.setRendered( false );
294         } // end if
295         else // display all but successLabel
```

Store selected seat type in the session bean

rve.java

```
296         {
297            instructionLabel.setRendered( true );
298            seatTypeDropDown.setRendered( true );
299            classTypeDropDown.setRendered( true );
300            reserveButton.setRendered( true );
301            successLabel.setRendered( false );
302            errorLabel.setRendered( true );
303         } // end else
304      } // end try
305      catch ( Exception e )
306      {
307         e.printStackTrace();
308      } // end catch
309
310      return null;
311   } // end method reserveButton_action
312 } // end class Reserve
```

Outline

Reserve.java

(4 of 4)

28.8 Passing an Object of a User-Defined Type to a Web Service

- **Web services can receive and return objects of user-defined types—known as custom types.**
- **Custom types that are sent to or from a web service using SOAP are serialized into XML format**
  - **This process is referred to as XML serialization**
  - **Handled for you automatically**
- **Custom type**
  - **If used to specify parameter or return types in web methods, must provide a public default or no-argument constructor**
  - **Any instance variables that should be serialized must have public *set* and *get* methods or the instance variables must be declared public**
  - **Any instance variable that is not serialized simply receives its default value when an object of the class is deserialized**

Common Programming Error 28.3

A runtime error occurs if an attempt is made to deserialize an object of a class that does not have a default or no-argument constructor.

Outline

Equation.java

(1 of 4)

```
1  // Fig. 28.19: Equation.java
2  // Class Equation that contains information about an equation
3  package com.deitel.iw3htp4.generator;
4
5  public class Equation
6  {
7     private int leftOperand;
8     private int rightOperand;
9     private int resultValue;
10    private String operationType;
11
12    // required no-argument
13    public Equation()
14    {
15       this( 0, 0, "+" );
16    } // end no-argument con
17
18    public Equation( int leftValue, int rightValue, String type )
19    {
20       leftOperand = leftValue;
21       rightOperand = rightValue;
22       operationType = type;
23
24       //determine resultValue
25       if ( operationType.equals( "+" ) ) // addition
26          resultValue = leftOperand + rightOperand;
27       else if ( operationType.equals( "-" ) ) // subtraction
28          resultValue = leftOperand - rightOperand;
29       else // multiplication
```

Set and *get* methods are provided for these instance variables so they can IL serialized

Required constructor because objects of this class are passed to or returned from web methods

Outline

Equation.java

(2 of 4)

```
30          resultValue = leftOperand * rightOperand;
31       } // end three argument constructor
32
33       // method that overrides Object.toString()
34       public String toString()
35       {
36          return leftOperand + " " + operationType + " " +
37             rightOperand + " = " + resultValue;
38       } // end method toString
39
40       // returns the left hand side of the equation as a String
41       public String getLeftHandSide()
42       {
43          return leftOperand + " " + operationType + " " + rightOperand;
44       } // end method getLeftHandSide
45
46       // returns the right hand side of the equation as a String
47       public String getRightHandSide()
48       {
49          return "" + resultValue;
50       } // end method getRightHandSide
51
52       // gets the leftOperand
53       public int getLeftOperand()
54       {
55          return leftOperand;
56       } // end method getLeftOperand
57
```

Outline

Equation.java

(3 of 4)

```
58    // gets the rightOperand
59    public int getRightOperand()
60    {
61       return rightOperand;
62    } // end method getRightOperand
63
64    // gets the resultValue
65    public int getReturnValue()
66    {
67       return resultValue;
68    } // end method getResultValue
69
70    // gets the operationType
71    public String getOperationType()
72    {
73       return operationType;
74    } // end method getOperationType
75
76    // required setter
77    public void setLeftHandSide( String value )
78    {
79       // empty body
80    } // end setLeftHandSide
81
82    // required setter
83    public void setRightHandSide( String value )
84    {
85       // empty body
86    } // end setRightHandSide
```

Outline

Equation.java

(4 of 4)

```
87
88    // required setter
89    public void setLeftOperand( int value )
90    {
91       // empty body
92    } // end method setLeftOperand
93
94    // required setter
95    public void setRightOperand( int value )
96    {
97       // empty body
98    } // end method setRightOperand
99
100   // required setter
101   public void setReturnValue( int value )
102   {
103      // empty body
104   } // end method setResultOperand
105
106   // required setter
107   public void setOperationType( String value )
108   {
109      // empty body
110   } // end method setOperationType
111} // end class Equation
```

Outline

Generator.java

(1 of 2)

```
1  // Fig. 28.20: Generator.java
2  // Web service that generates random equations
3  package com.deitel.iw3htp4.ch28.equationgenerator;
4
5  import java.util.Random;
6  import javax.jws.WebService;
7  import javax.jws.WebMethod;
8  import javax.jws.WebParam;
9
10 @WebService( name = "EquationGenerator",
11    serviceName = "EquationGeneratorService" )
12 public class EquationGenerator
13 {
14    private int minimum;
15    private int maximum;
16
```

Outline

Generator.java

(2 of 2)

```
17    // generates a math equation and returns it as an Equation object
18    @WebMethod( operationName = "generateEquation" )
19    public Equation generateEquation(
20       @WebParam( name = "operation" ) String operation,
21       @WebParam( name = "difficulty" ) int difficulty )
22    {
23       minimum = ( int ) Math.pow( 10, difficulty - 1 );
24       maximum = ( int ) Math.pow( 10, difficulty );
25
26       Random randomObject = new Random();
27
28       return new Equation(
29          randomObject.nextInt( maximum - minimum ) + minimum,
30          randomObject.nextInt( maximum - minimum ) + minimum, operation
31    } // end method generateEquation
32 } // end class EquationGenerator
```

Note that no special code is required to return an object of our user-defined class from the web method

a) Using the `EquationGenerator` web service's `Tester` web page to generate an `Equation`.

Fig. 28.21 | Testing a web method that returns an XML serialized **`Equation`** object. (Part 1 of 2.)

b) Result of generating an `Equation`.

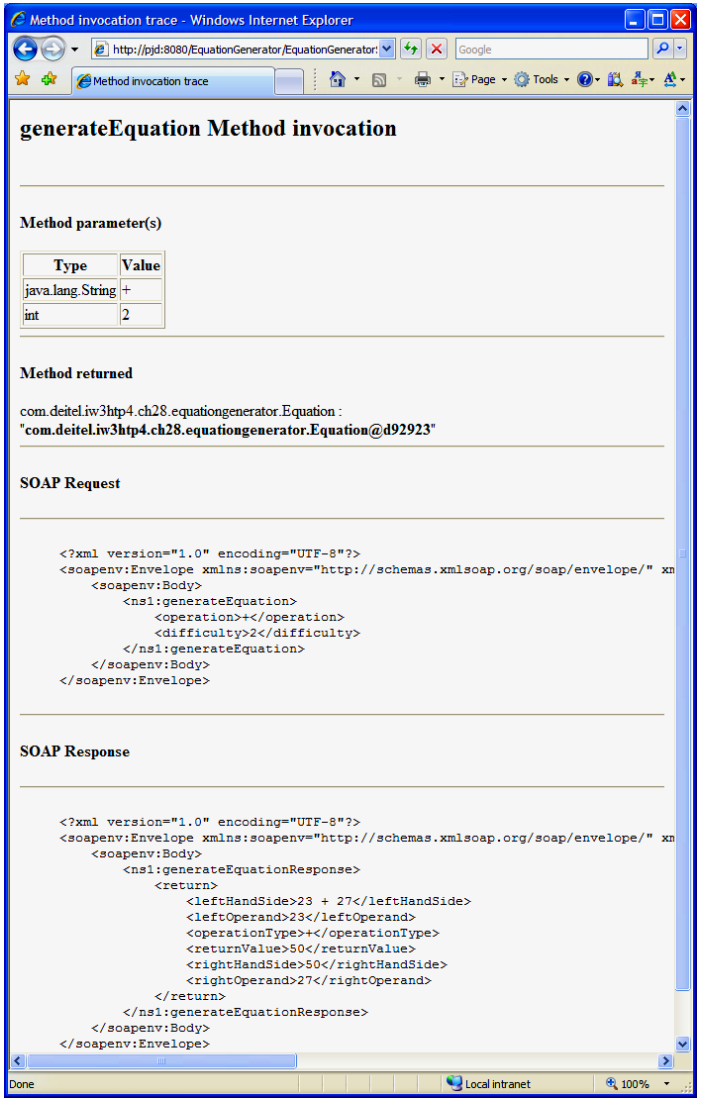


Fig. 28.21 | Testing a web method that returns an XML serialized **`Equation`** object. (Part 2 of 2.)

Outline

```
1  // Fig. 28.22: EquationGeneratorClientJFrame.java
2  // Math tutoring program using web services to generate equations
3  package com.deitel.iw3htp4.ch28.equationgeneratorclient;
4
5  import javax.swing.JOptionPane;
6
7  public class EquationGeneratorClientJFrame extends javax.swing.JFrame
8  {
9     private EquationGeneratorService service; // used to obtain proxy
10    private EquationGenerator proxy; // used to access the web service
11    private Equation equation; // represents an equation
12    private int answer; // the user's answer to the question
13    private String operation = "+"; // mathematical operation +, - or *
14    private int difficulty = 1; // 1, 2 or 3 digits in each number
15
16    // no-argument constructor
17    public EquationGeneratorClientJFrame()
18    {
19       initComponents();
20
21       try
22       {
23          // create the objects for accessing the EquationGenerator service
24          service = new EquationGeneratorService();
25          proxy = service.getEquationGeneratorPort();
26       } // end try
```

Equation GeneratorClientJ Frame.java

(1 of 6)

Outline

Equation GeneratorClientJ Frame.java

(2 of 6)

```
27       catch ( Exception ex )
28       {
29          ex.printStackTrace();
30       } // end catch
31    } // end no-argument constructors
32
33    // The initComponents method is autogenerated by Netbeans and is called
34    // from the constructor to initialize the GUI. This method is not shown
35    // here to save space. Open EquationGeneratorClientJFrame.java in this
36    // example's folder to view the complete generated code (lines 37-156).
37
157   // obtains the difficulty level selected by the user
158   private void levelJComboBoxItemStateChanged(
159      java.awt.event.ItemEvent evt )
160   {
161      // indices start at 0, so add 1 to get the difficulty level
162      difficulty = levelJComboBox.getSelectedIndex() + 1;
163   } // end method levelJComboBoxItemStateChanged
164
165   // obtains the mathematical operation selected by the user
166   private void operationJComboBoxItemStateChanged(
167      java.awt.event.ItemEvent evt )
168   {
169      String item = ( String ) operationJComboBox.getSelectedItem();
170
```

Outline

Equation GeneratorClientJ Frame.java

(3 of 6)

```
171      if ( item.equals( "Addition" ) )
172         operation = "+"; // user selected addition
173      else if ( item.equals( "Subtraction" ) )
174         operation = "-"; // user selected subtraction
175      else
176         operation = "*"; // user selected multiplication
177   } // end method operationJComboBoxItemStateChanged
178
179   // checks the user's answer
180   private void checkAnswerJButtonActionPerformed(
181      java.awt.event.ActionEvent evt )
182   {
183      if ( answerJTextField.getText().equals( "" ) )
184      {
185         JOptionPane.showMessageDialog(
186            this, "Please enter your answer." );
187      } // end if
188
189      int userAnswer = Integer.parseInt( answerJTextField.getText() );
190
191      if ( userAnswer == answer )
192      {
193         equationJLabel.setText( "" );
194         answerJTextField.setText( "" );
195         checkAnswerJButton.setEnabled( false );
196         JOptionPane.showMessageDialog( this, "Correct! Good Job!",
197            "Correct", JOptionPane.PLAIN_MESSAGE );
198      } // end if
```

Outline

Equation GeneratorClientJ Frame.java

(4 of 6)

```
199       else
200       {
201          JOptionPane.showMessageDialog( this, "Incorrect. Try again.",
202             "Incorrect", JOptionPane.PLAIN_MESSAGE );
203       } // end else
204    } // end method checkAnswerJButtonActionPerformed
205
206    // generates a new Equation based on user's selections
207    private void generateJButtonActionPerformed(
208       java.awt.event.ActionEvent evt )
209    {
210       try
211       {
212          equation = proxy.generateEquation( operation, difficulty );
213          answer = equation.getReturnValue();
214          equationJLabel.setText( equation.getLeftHandSide() + " =" );
215          checkAnswerJButton.setEnabled( true );
216       } // end try
217       catch ( Exception e )
218       {
219          e.printStackTrace();
220       } // end catch
221    } // end method generateJButtonActionPerformed
222
223    // begins program execution
```

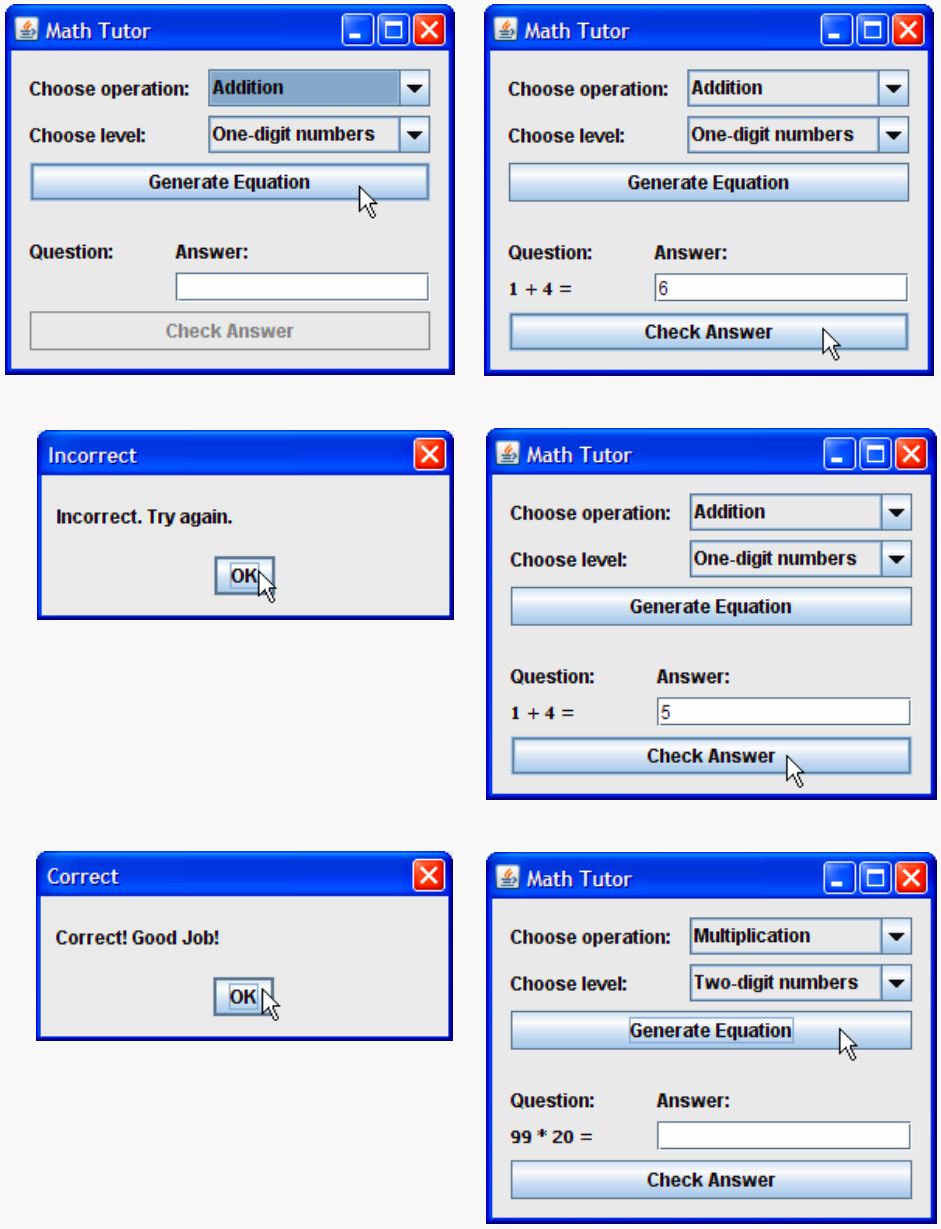
Note that no special code is required to receive an object of our user-defined class from the web method

Outline

Equation GeneratorClientJ Frame.java

(5 of 6)

```
224   public static void main( String args[] )
225   {
226      java.awt.EventQueue.invokeLater(
227         new Runnable()
228         {
229            public void run()
230            {
231               new EquationGeneratorClientJFrame().setVisible( true );
232            } // end method run
233         } // end anonymous inner class
234      ); // end call to java.awt.EventQueue.invokeLater
235   } // end method main
236
237   // Variables declaration - do not modify
238   private javax.swing.JLabel answerJLabel;
239   private javax.swing.JTextField answerJTextField;
240   private javax.swing.JButton checkAnswerJButton;
241   private javax.swing.JLabel equationJLabel;
242   private javax.swing.JButton generateJButton;
243   private javax.swing.JComboBox levelJComboBox;
244   private javax.swing.JLabel levelJLabel;
245   private javax.swing.JComboBox operationJComboBox;
246   private javax.swing.JLabel operationJLabel;
247   private javax.swing.JLabel questionJLabel;
248   // End of variables declaration
249} // end class EquationGeneratorClientJFrame
```



Outline

`Equation GeneratorClientJ Frame.java`

(6 of 6)

28.9 REST-Based Web Services in ASP.NET

- **Representational State Transfer (REST)**
  - **An architectural style for implementing web services**
  - **Not a standard, but RESTful web services are implemented using web standards, such as HTTP, XML and JSON**
- **Each operation in a RESTful web service is easily identified by a unique URL**
- **REST web services can be invoked from a program or directly from a web browser by entering the URL in the browser's address field**
- **Many Web 2.0 web services provide RESTful interfaces**
- **Microsoft's Visual Web Developer 2005 Express provides a simple way to build REST-based web services**

28.9.1 REST-Based Web Service Functionality

- When creating a web service in Visual Web Developer, you work almost exclusively in the code-behind file.
- `System.Web.Script.Serialization` namespace
  - tools to convert .NET objects into JSON strings
- `WebService` attribute
  - indicates that a class implements a web service
  - allows you to specify the web service's namespace
- Each new web service class created in Visual Web Developer inherits from class `System.Web.Services.WebService`
- `WebMethod` attribute
  - exposes a method so that it can be called remotely (similar to Java's `@WebMethod` annotation)

Outline

CalendarService.vb

```
1  ' Fig. 28.23 CalendarService.vb
2  ' REST-based event web service.
3  Imports System.Web
4  Imports System.Web.Services
5  Imports System.Web.Services.Protocols
6  Imports System.Data ' Used to access a database
7  Imports System.Web.Script.Serialization ' Used to return JSON
8
9  <WebService(Namespace:="http://www.deitel.com/")> _
10 <WebServiceBinding(ConformsTo:=WsiProfiles.BasicProfile1_1)> _
11 <Global.Microsoft.VisualBasic.CompilerServices.DesignerGenerated()> _
12 Public Class CalendarService
13    Inherits System.Web.Services.WebService
14
15    ' variables used to access the database
16    Private calendarDataSet As New CalendarDataSet(
17    Private eventsTableAdapter As _
18       New CalendarDataSetTableAdapters.EventsTableAdapter()
19
20    ' retrieve the event from the database given an id
21    <WebMethod(Description:="Gets a list of events for a given id.")> _
22    Public Sub getItemById(ByVal id As Integer)
23      ' set up the data set
24       eventsTableAdapter.FillById(calendarDataSet.Events, id)
25
```

Begins the definition of a web service class in ASP.NET (with sual Basic).

Web service classes in ASP.NET should inherit from this class.

Begins the definition of a web method in ASP.NET.

Outline

```
26       ' insert the data into an Item object.
27       Dim identification As String = calendarDataSet.Events(0).ID
28       Dim description As String = calendarDataSet.Events(0).Description
29       Dim itemObject As New Item(identification, description)
30
31       ' convert the data to JSON and send it back to the client
32       Dim serializer As JavaScriptSerializer = New JavaScriptSerializer()
33       Dim response As String = serializer.Serialize(itemObject)
34       HttpContext.Current.Response.Write(response) ' send to client
35    End Sub ' getItemById
36
37    ' retrieve the list of events that occur on a specific date
38    <WebMethod(Description:="Gets a list of events for a given date.")> _
39    Public Sub getItemsByDate(ByVal eventDate As String)
40       eventsTableAdapter.FillByDate(calendarDataSet.Events, eventDate)
41       Dim identification As String ' string used to store the id
42       Dim description As String ' string used to store the description
43       Dim eventRow As DataRow ' used to iterate over the DataSet
44       Dim length As Integer = calendarDataSet.Events.Rows.Count
45       Dim itemObject As = New Item(0 To length - 1) {} ' initialize array
46       Dim count As Integer = 0
47
48       ' insert the data into an array of Item objects
```

Used to serialize an object into JSON format.

Outline

CalendarService.vb

```
49       For Each eventRow In calendarDataSet.Events.Rows
50          identification = eventRow.Item("ID")
51          description = eventRow.Item("Description")
52          itemObject(count) = New Item(identification, description)
53          count += 1
54       Next
55
56       ' convert the data to JSON and send it back to the client
57       Dim serializer As New JavaScriptSerializer()
58       Dim response As String = serializer.Serialize(itemObject)
59       HttpContext.Current.Response.Write(response)
60    End Sub ' getItemsByDate
61
62    ' modify the description of the event with the given id
63    <WebMethod(Description:="Updates an event's description.")> _
64    Public Sub Save(ByVal id As String, ByVal descr As String)
65       eventsTableAdapter.UpdateDescription(descr, id)
66       getItemById(id)
67    End Sub ' Save
68 End Class ' CalendarService
```

Used to serialize an object into JSON format.

Common Programming Error 28.4

Properties and instance variables that are not public will not be serialized as part of an object's JSON representation.

Outline

Item.vb

(1 of 2)

```
1  ' Fig. 28.24 Item.vb
2  ' A simple class to create objects to be converted into JSON format.
3  Imports Microsoft.VisualBasic
4  Public Class Item
5     Private descriptionValue As String ' the item's description
6     Private idValue As String ' the item's id
7
8     ' Default constructor
9     Public Sub New()
10    End Sub 'New
11
12    ' constructor that initializes id and description
13    Public Sub New(ByVal ident As String, ByVal descr As String)
14       id = ident
15       description = descr
16    End Sub 'New
17
18    ' property that encapsulates the description
```

Outline

Item.vb

(2 of 2)

```
19    Public Property description() As String
20       Get
21          Return descriptionValue
22       End Get
23       Set(ByVal value As String)
24          descriptionValue = value
25       End Set
26    End Property ' description
27
28    ' Property that encapsulates the id
29    Public Property id() As String
30       Get
31          Return idValue
32       End Get
33       Set(ByVal value As String)
34          idValue = value
35       End Set
36    End Property ' id.
37 End Class ' Item
```

28.9.2 Creating an ASP.NET REST-Based Web Service

- **Create an ASP.NET web service project in Visual Web Developer**
- **Code for web service is placed in the App_Code directory**
- **Define your web service and supporting classes**
- **Change the `Web.Config` File to allow REST requests**
  - **paste the following code as a new element in the `system.web` element.**

```
<webServices>
   <protocols>
      <add name="HttpGet"/>
      <add name="HttpPost"/>
   </protocols>
</webServices>
```

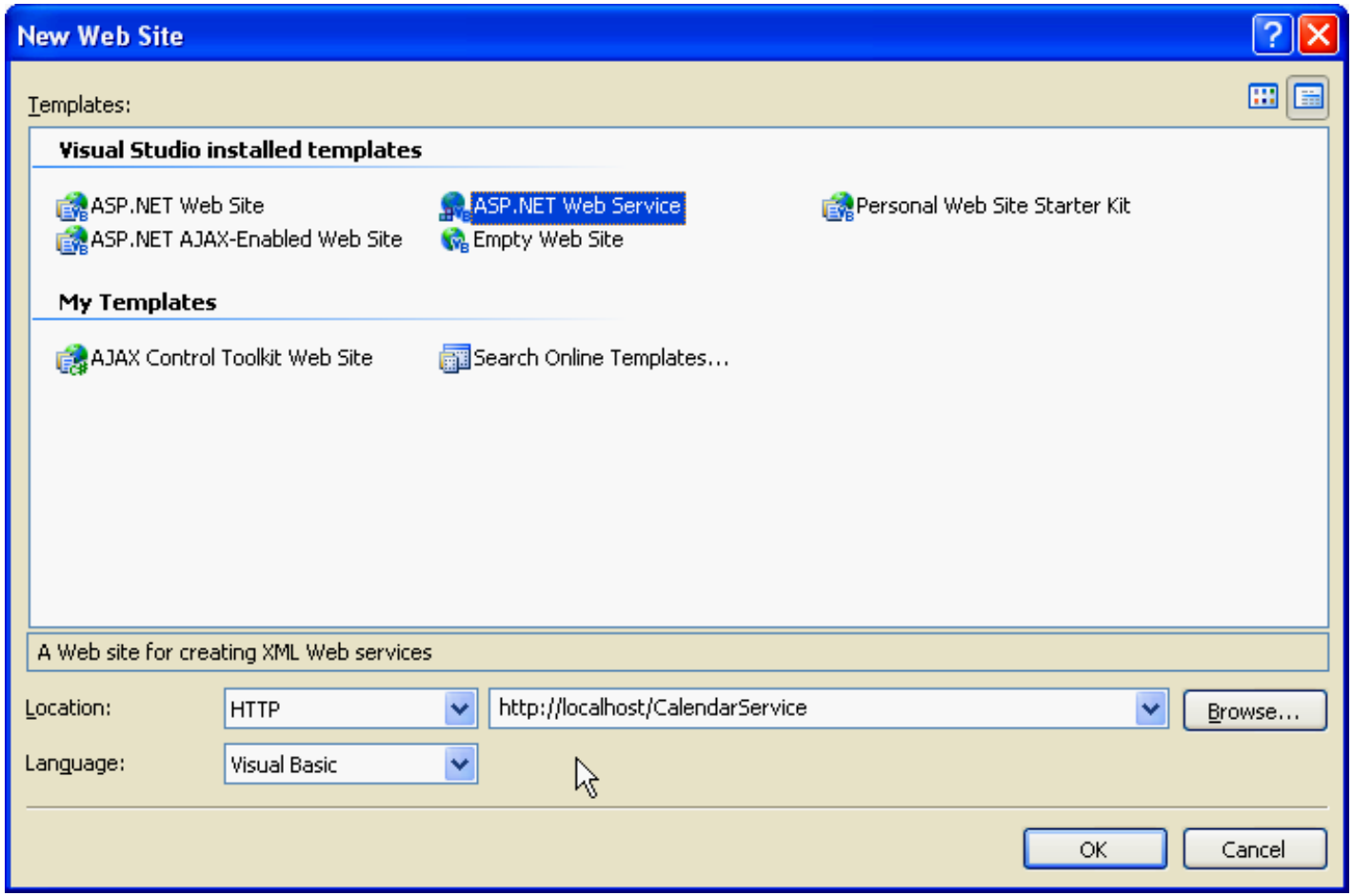


Fig. 28.25 | Creating an **ASP.NET Web Service** in Visual Web Developer.

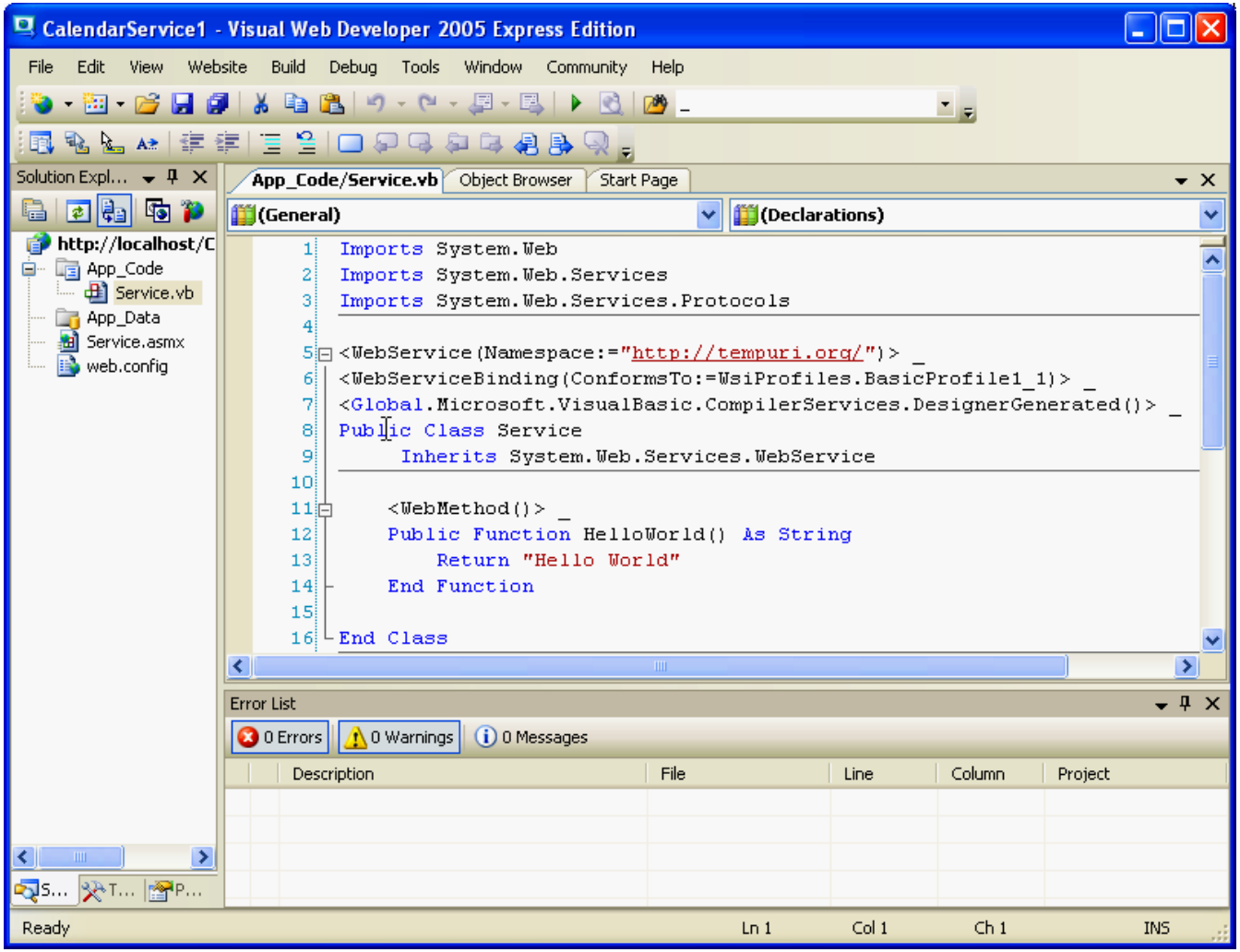


Fig. 28.26 | Code view of a web service.

Error-Prevention Tip 28.1

Update the web service's ASMX file appropriately whenever the name of a web service's code-behind file or the class name changes. Visual Web Developer creates the ASMX file, but does not automatically update it when you make changes to other files in the project.

28.9.2 Creating an ASP.NET REST-Based Web Service (Cont.)

- To use JSON and the `JavaScriptSerializer`, you must first install the ASP.NET Ajax extensions (http://ajax.asp.net)
- After you have installed ASP.NET Ajax
  - right click the project name in the solution explorer and select **Add Reference...** to display the **Add Reference** window
  - select `System.Web.Extensions` from the .NET tab and click OK

28.9.3 Adding Data Components to a Web Service Creating an ASP.NET REST-Based Web Service

- Add a `DataSet` to the project
- Select the data source and create a connection
- Open the **Query Builder** and add the table(s) for your query
- Build your query
- Repeat as necessary for each query

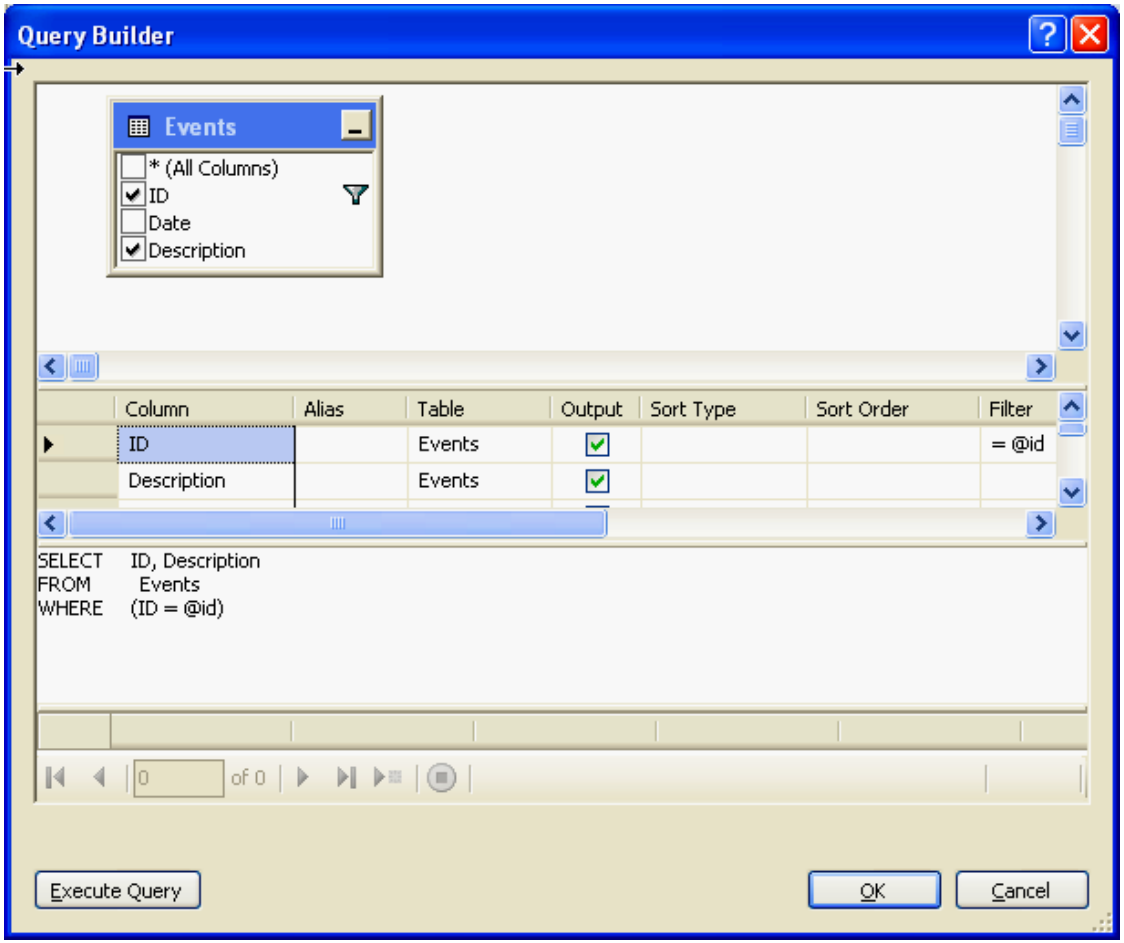


Fig. 28.27 | QueryBuilder dialog specifying a **SELECT** query that selects an event with a specific ID.

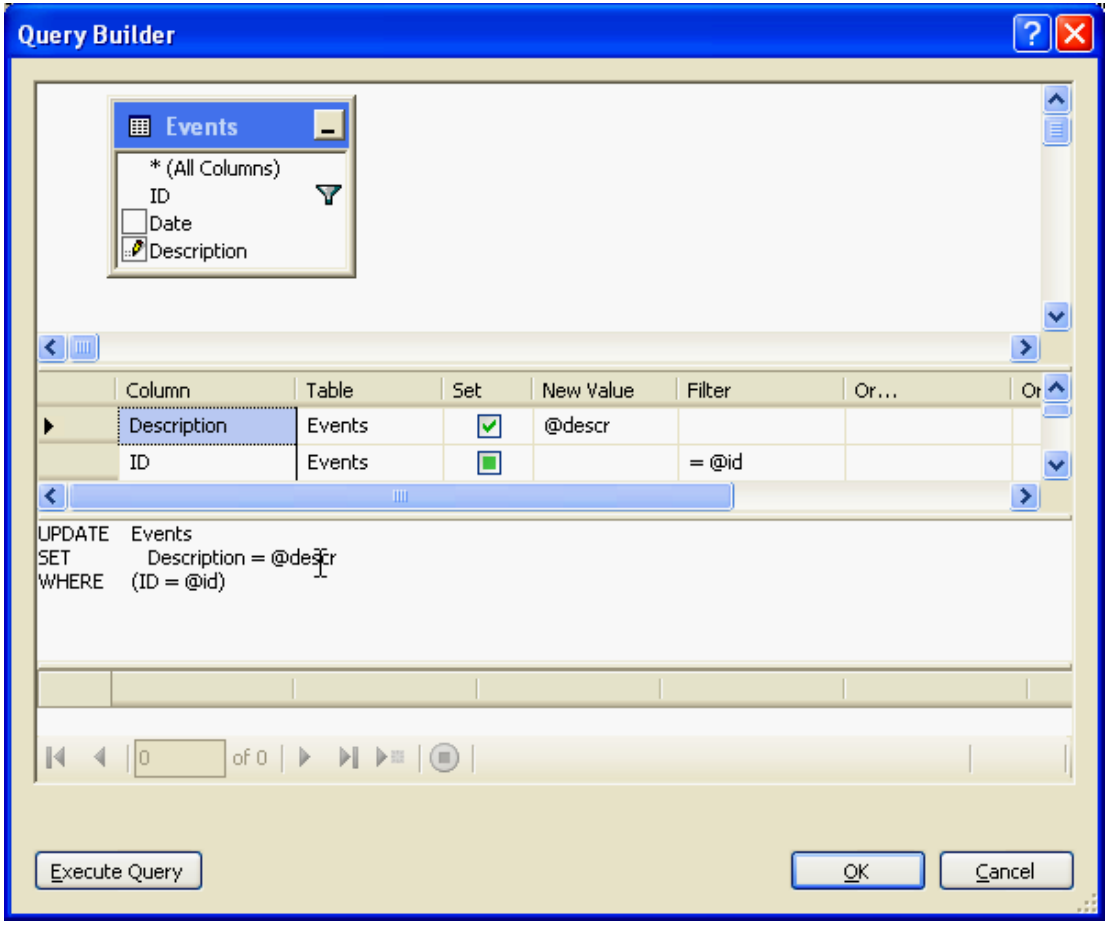


Fig. 28.28 | QueryBuilder specifying an **UPDATE** statement used to modify a description.

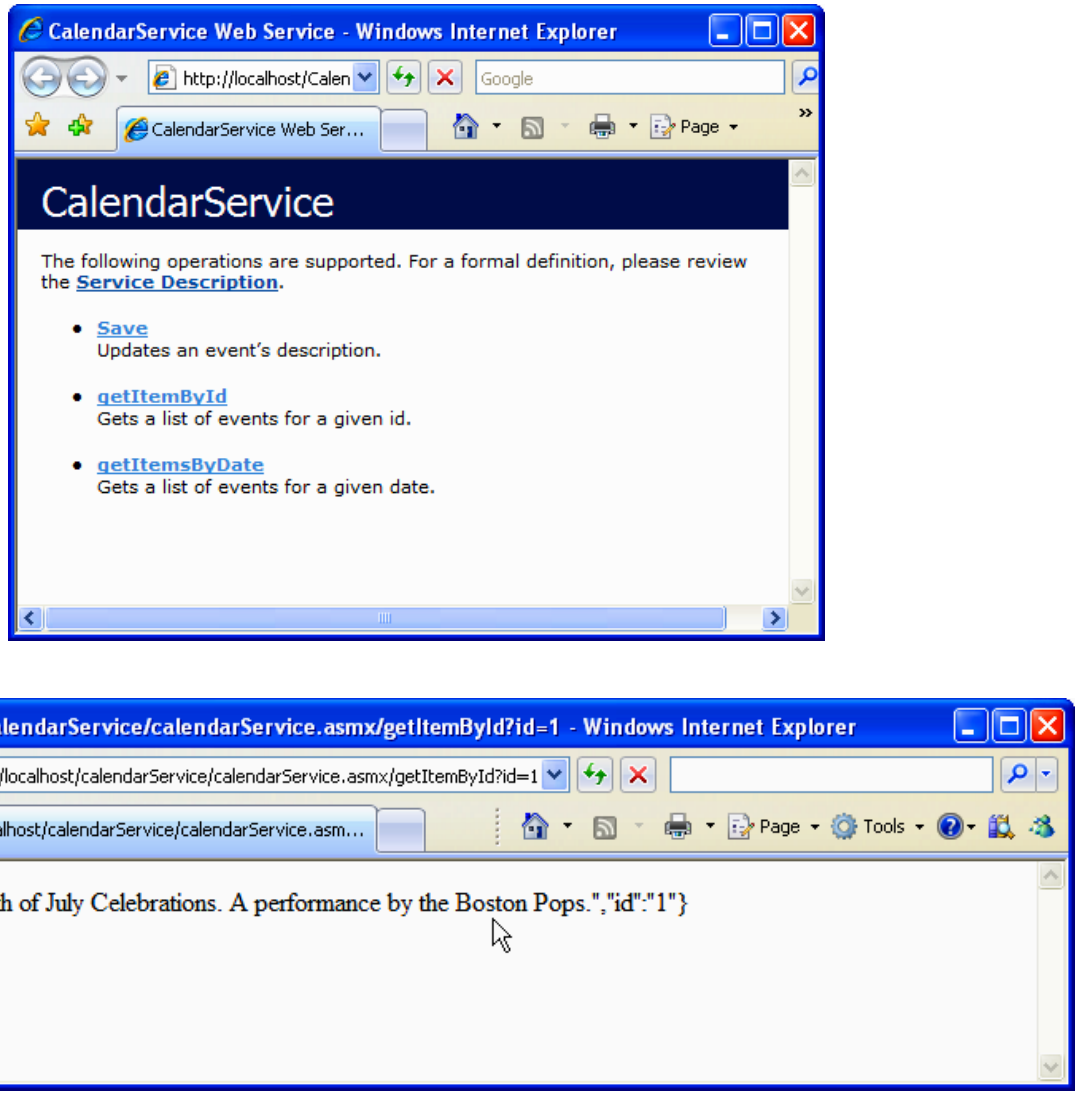


Fig. 28.29 | The test page for the **`CalendarService`** web service.