```
      _jspx_page_context = pageContext;
      application = pageContext.getServletContext();
      config = pageContext.getServletConfig();
      session = pageContext.getSession();
      out = pageContext.getOut();
      _jspx_out = out;

      out.write("<html><head>\r\n");
      out.write("<title>Simple</title>\r\n");
      out.write("</head>\r\n");
      out.write("<body>\r\n");
      out.write("Hello, Bender\r\n");
      out.write("</body></html>");
    } catch (Throwable t) {
      if (!(t instanceof SkipPageException)){
        out = _jspx_out;
        if (out != null && out.getBufferSize() != 0)
          out.clearBuffer();
        if (_jspx_page_context != null)
_jspx_page_context.handlePageException(t);
      }
    } finally {
      if (_jspxFactory != null)
_jspxFactory.releasePageContext(_jspx_page_context);
    }
  }
}
```

JSP-код заключается в специальные теги, которые указывают контейнеру, чтобы он использовал этот код для генерации сервлета или его части. Таким образом поддерживается документ, который одновременно содержит и статическую страницу, и теги Java, которые управляют этой страницей. Статические части HTML-страниц посылаются в виде строк в метод **`write()`**. Динамические части включаются прямо в код сервлета. С этого момента страница ведет себя как обычная HTML-страница с ассоциированным сервлетом.

Взаимодействие сервлета и JSP

Страницы JSP и сервлеты никогда не используются в информационных системах друг без друга. Причиной являются принципиально различные роли, которые играют данные компоненты в приложении. Страница JSP ответственна за формирование пользовательского интерфейса и отображение информации, переданной с сервера. Сервлет выполняет роль контроллера запросов и ответов, то есть принимает запросы от всех связанных с ним JSP-страниц, вызывает соответствующую бизнес-логику для их (запросов) обработки и в зависимости от результата выполнения решает, какую JSP поставить этому результату в соответствие.

Ниже приведен пример вызова сервлета из JSP с последующим вызовом другой JSP.

```
<!--пример # 4 : страница JSP с вызовом сервлета : index.jsp -->
<%@ page language="java" contentType="text/html; char-
set=ISO-8859-5" pageEncoding="ISO-8859-5"%>
<html><body>
<jsp:useBean id="gc" class="java.util.GregorianCalendar"/>
<jsp:getProperty name="gc" property="time"/>
<FORM action="serv" method="POST">
          <INPUT type="submit" value="Вызвать сервлет">
</FORM>
</body></html>
```

В результате запуска проекта в браузер будет выведено:

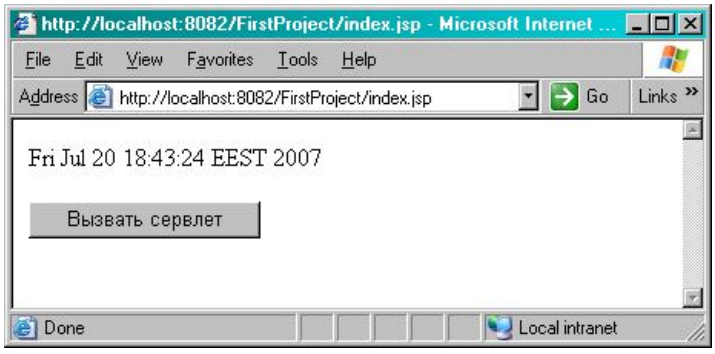

Рис. 17.4. Запуск index.jsp

Кодировка для символов кириллицы задана с помощью директивы **page**. Action-теги **useBean** и **getProperty** используются для создания объекта класса **GregorianCalendar** в области видимости JSP и вывода его значения. Сервлет **ContServlet** вызывается методом **POST**.

```
// пример # 5 : простой контроллер : ContServlet.java
package chapt17;
import java.io.IOException;
import java.util.Calendar;
import java.util.Locale;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

public class ContServlet
          extends javax.servlet.http.HttpServlet {

     protected void doPost(HttpServletRequest request,
                    HttpServletResponse response)
               throws ServletException, IOException {
     //добавление атрибута к запросу
     request.setAttribute("loc", Locale.getDefault());
     //добавление атрибута к сессии
     request.getSession().setAttribute("calend",
                         Calendar.getInstance());
```

```
        //получение объекта RequestDispatcher и вызов JSP
request.getRequestDispatcher("/main.jsp").forward(request,
                                                    response);
        }
}
```

Передачу информации между JSP и сервлетом можно осуществлять, в частности, с помощью добавления атрибутов к объектам **HttpServletRequest**, **HttpSession**, **HttpServletContext**. Вызов **main.jsp** из сервлета в данном случае производится методом **forward()** интерфейса **RequestDispatcher**.

```
<!--пример # 6 : страница, вызванная сервлетом : main.jsp -->
<%@ page language="java"
contentType="text/html; charset=utf-8"
pageEncoding="utf-8"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core"
prefix="c"%>
<html><body>
<h3>Региональные установки и Время</h3>
<c:out value="Locale from request: ${loc}"/><br>
<c:out value="Time from Servlet: ${calend.time}"/>
</body></html>
```

После вызова сервлета и последующего вызова **main.jsp** будет выведено:

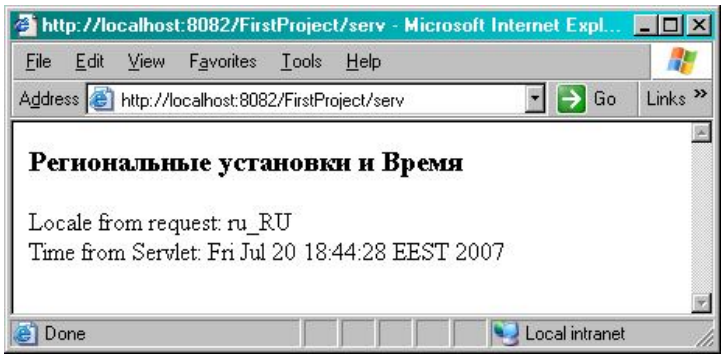

Рис. 17.5. Вывод информации страницей main.jsp

В данном коде директива **taglib** подключает JSP Standard Tag Library (JSTL), и становится возможным вызов тега **<c:out>**, а также использование Expression Language (EL) в виде **${loc}**.

Конфигурационный файл **web.xml** для данной задачи должен содержать следующую информацию:

```
<servlet>
  <display-name>Controller</display-name>
  <servlet-name>controller</servlet-name>
  <servlet-class>chapt17.ContServlet</servlet-class>
 </servlet>
  <servlet-mapping>
  <servlet-name>controller</servlet-name>
```

```
  <url-pattern>/serv</url-pattern>
 </servlet-mapping>
```

В этой главе была дана общая информация о взаимодействии различных компонентов Web-приложения.

Задания к главе 17

Вариант А

Создать сервлет и взаимодействующие с ним пакеты Java-классов и JSP-страницы, способные выполнить следующие действия:

1. Подсчет суммы случайным образом выбранных элементов массива.
2. Вывести полное название страны и языка.
3. Подсчитать время между выполнением сервлета и JSP в наносекундах.
4. Создать массив дат и вывести самую позднюю дату.
5. Задать температуру. Если она меньше нуля, вывести значение температуры синим цветом, если больше, то красным.
6. Создать приложение, выводящее фамилию разработчика, дату и время получения задания, а также дату и время его выполнения.

Вариант В

Задания варианта В главы 1 выполнить на основе сервлетов. Число **n** генерировать с помощью методов класса **java.util.Random**.

Тестовые задания к главе 17

Вопрос 17.1.

Укажите стандартный путь к сервлету `com.example.MyServlet`, чтобы Web-приложение могло к нему обратиться.

1) /lib/MyServlet.class
2) /com/example/MyServlet.class
3) /WEB-INF/lib/MyServlet.class
4) classes/com/example/MyServlet.class
5) /servlets/com/example/MyServlet.class
6) /WEB-INF/classes/com/example/MyServlet.class

Вопрос 17.2.

Дано:

```
public void service(ServletRequest request,
                    ServletResponse response) {
    ServletInputStream sis =
        //1
    }
```

Какой код инициализирует ссылку на байтовый поток в строке 1?

1) `request.getWriter();`
2) `request.getReader();`
3) `request.getInputStream();`
4) `request.getResourceAsStream();`
5) `request.getResourceAsStream(ServletRequest.REQUEST);`