Глава 21
СЕССИИ, СОБЫТИЯ И ФИЛЬТРЫ

Сеанс (сессия)

При посещении клиентом Web-ресурса и выполнении вариантов запросов, контекстная информация о клиенте не хранится. В протоколе HTTP нет возможностей для сохранения и изменения информации о предыдущих посещениях клиента. При этом возникают проблемы в распределенных системах с различными уровнями доступа для разных пользователей. Действия, которые может делать администратор системы, не может выполнять гость. В данном случае необходима проверка прав пользователя при переходе с одной страницы на другую. В иных случаях необходима информация о предыдущих запросах клиента. Существует несколько способов хранения текущей информации о клиенте или о нескольких соединениях клиента с сервером.

Сеанс (сессия) – соединение между клиентом и сервером, устанавливаемое на определенное время, за которое клиент может отправить на сервер сколько угодно запросов. Сеанс устанавливается непосредственно между клиентом и Web-сервером. Каждый клиент устанавливает с сервером свой собственный сеанс.

Сеансы используются для обеспечения хранения данных во время нескольких запросов Web-страницы или на обработку информации, введенной в пользовательскую форму в результате нескольких HTTP-соединений (например, клиент совершает несколько покупок в интернет-магазине; студент отвечает на несколько тестов в системе дистанционного обучения). Как правило, при работе с сессией возникают следующие проблемы:

- поддержка распределенной сессии (синхронизация/репликация данных, уникальность идентификаторов и т.д.);
- обеспечение безопасности;
- проблема инвалидации сессии (expiration), предупреждение пользователя об уничтожении сессии и возможность ее продления (watchdog).

Чтобы открыть новый сеанс, используется метод **`getSession()`** интерфейса **`HttpServletRequest`**. Метод извлекает из переданного в сервлет запроса объект сессии класса **`HttpSession`**, соответствующий данному пользователю. Сессия содержит информацию о дате и времени создания последнего обращения к сессии, которая может быть извлечена с помощью методов **`getCreationTime()`** и **`getLastAccessedTime()`**.

Если для метода **`getSession(boolean param)`** входной параметр равен **`true`**, то сервлет-контейнер проверяет наличие активного сеанса, установленного с данным клиентом. В случае успеха метод возвращает дескриптор этого сеанса. В противном случае метод устанавливает новый сеанс:

```
HttpSession se = request.getSession(true);
```

после чего начинается сбор информации о клиенте.

Чтобы сохранить значения переменной в текущем сеансе, используется метод **setAttribute()** класса **HttpSession**, прочесть – **getAttribute()**, удалить – **removeAttribute()**. Список имен всех переменных, сохраненных в текущем сеансе, можно получить, используя метод **Enumeration getAttributeNames()**, работающий так же, как и соответствующий метод интерфейса **HttpServletRequest**.

Метод **String getId()** возвращает уникальный идентификатор, который получает каждый сеанс при создании. Метод **isNew()** возвращает **false** для уже существующего сеанса и **true** – для только что созданного.

Если требуется сохранить для использования одну из переменных сеанса, представляющего собой целое число, то:

```
se.setAttribute("teacherId", new Integer(71));
```

После этого любой подключившийся к текущему сеансу сервлет сможет прочесть значение переменной **teacherId** следующим образом:

```
Integer testId = (Integer)se.getAttribute("teacherID");
```

Завершить сеанс можно методом **invalidate()**. Сеанс уничтожает все связи с объектами, и данные, сохраненные в старом сеансе, будут потеряны для всех приложений.

/ пример # 1 : добавление информации в сессию : SessionServlet.java */*

```
package chapt21;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

public class SessionServlet extends HttpServlet {
      protected void doGet(
            HttpServletRequest req,
            HttpServletResponse resp)
            throws ServletException {
            performTask(req, resp);
      }
      private void performTask(
            HttpServletRequest req,
            HttpServletResponse resp)
            throws ServletException {
            SessionLogic.printToBrowser(resp, req);
      }
}
```

```
package chapt21;
import java.io.IOException;
import java.io.PrintWriter;
import java.util.Date;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;
```

```
public class SessionLogic {

      public static void printToBrowser(
      HttpServletResponse resp, HttpServletRequest req) {
            try {
/* возвращается ссылка на сессию для текущего пользователя (если сессия еще не
существует, то она при этом создается) */
            HttpSession session = req.getSession(true);

                  PrintWriter out = resp.getWriter();

                  StringBuffer url = req.getRequestURL();
                  session.setAttribute("URL", url);

                  out.write("My session counter: ");
/* количество запросов, которые были сделаны к данному сервлету текущим
пользователем в рамках текущей пользовательской сессии (следует приводить
значение к строковому виду для корректного отображения в результате) */
out.write(String.valueOf(prepareSessionCounter(session)));
                  out.write("<br> Creation Time : "
            + new Date(session.getCreationTime()));
                  out.write("<br> Time of last access : "
            + new Date(session.getLastAccessedTime()));
                  out.write("<br> session ID : "
            + session.getId());
                  out.write("<br> Your URL: " + url);
                  int timeLive = 60 * 30;
                  session.setMaxInactiveInterval(timeLive);
            out.write("<br>Set max inactive interval : "
            + timeLive + "sec");

                  out.flush();
                  out.close();
            } catch (IOException e) {
                  e.printStackTrace();
            throw new RuntimeException("Failed : " + e);
            }
      }
/* увеличивает счетчик обращений к текущему сервлету и кладет его в сессию */
      private static int prepareSessionCounter(
                        HttpSession session) {
            Integer counter =
            (Integer)session.getAttribute("counter");

            if (counter == null) {
               session.setAttribute("counter", 1);
                  return 1;
            } else {
                  counter++;
```

```
            session.setAttribute("counter", counter);
                  return counter;
            }
      }
}
```

В результате в браузер будет выведено:

My session counter: 3
Creation Time : Sun Jan 29 16:02:30 EET 2006
Time of last access : Sun Jan 29 16:02:38 EET 2006
session ID : 314A546CD9270A840E0BDA3286636B20
Your URL: http://localhost:8080/FirstProject/SessionServlet
Set max inactive interval : 1800sec

В качестве данных сеанса выступают: счетчик кликов – объект типа **Integer** и URL запроса, сохраненный в объекте **StringBuffer**. В ответ на пользовательский запрос сервлет **SessionServlet** возвращает страницу HTML, на которой отображаются все атрибуты сессии, время создания и последнего доступа, идентификационный номер сессии и время инвалидации (жизни) сессии. Это время можно задать с помощью тега **session-config** в **web.xml** в виде:

```
<session-config>
    <session-timeout>30</session-timeout>
</session-config>
```

где время ожидания задается в минутах.

В следующем примере рассмотрен процесс ликвидации сессии при отсутствии активности за определенный промежуток времени.

```
/* пример # 2 : инвалидация и ликвидация сессии : TimeSessionServlet.java */
package chapt21;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;

public class TimeSessionServlet extends HttpServlet {
      boolean flag = true;

      protected void doGet(HttpServletRequest req,
HttpServletResponse resp)
                  throws ServletException {
            performTask(req, resp);
      }
      private void performTask(HttpServletRequest req,
 HttpServletResponse resp) throws ServletException {

            HttpSession session = null;
            if (flag) {
            //создание сессии и установка времени инвалидации
                  session = req.getSession();
```

```
                int timeLive = 10; //десять секунд!
                session.setMaxInactiveInterval(timeLive);
                flag = false;
            } else {
            //если сессия не существует, то ссылка на нее не будет получена
                session = req.getSession(false);
                    }
            TimeSession.go(resp, req, session);
      }
}
package chapt21;
import java.io.IOException;
import java.io.PrintWriter;
import java.util.Date;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;

public class TimeSession {

      public static void go(HttpServletResponse resp,
HttpServletRequest req, HttpSession session ) {
            PrintWriter out = null;
            try {
                out = resp.getWriter();
                out.write("<br> Creation Time : "
            + new Date(session.getCreationTime()));
                out.write("<br> Session alive! ");

                out.flush();
                out.close();
            } catch (NullPointerException e) {
            //если сессия не существует, то генерируется исключение
                if (out != null)
                out.print("Session disabled!");
            } catch (IOException e) {
                e.printStackTrace();
            throw new RuntimeException("i/o failed: " + e);
            }
      }
   }
```

При первом запуске в браузер будет выведено:
Creation Time : Tue Aug 14 17:54:23 EEST 2007
Session alive!

Если повторить запрос к сервлету менее чем за 10 секунд, вывод будет повторен. Если же запрос повторить более через десять секунд, сессия будет автоматически уничтожена, и в браузер будет выведено следующее:
Session disabled!