Cookie

Для хранения информации на компьютере клиента используются возможности класса **Cookie**.

Cookie – это небольшие блоки текстовой информации, которые сервер посылает клиенту для сохранения в файлах cookies. Клиент может запретить браузеру прием файлов cookies. Браузер возвращает информацию обратно на сервер как часть заголовка HTTP, когда клиент повторно заходит на тот же Web-ресурс. Cookies могут быть ассоциированы не только с сервером, но и также с доменом – в этом случае браузер посылает их на все серверы указанного домена. Этот принцип лежит в основе одного из протоколов обеспечения единой идентификации пользователя (Single Signon), где серверы одного домена обмениваются идентификационными маркерами (token) с помощью общих cookies.

Cookie были созданы в компании Netscape как средства отладки, но теперь используются повсеместно. Файл cookie – это файл небольшого размера для хранения информации, который создается серверным приложением и размещается на компьютере пользователя. Браузеры накладывают ограничения на размер файла cookie и общее количество cookie, которые могут быть установлены на пользовательском компьютере приложениями одного Web-сервера.

Чтобы послать cookie клиенту, сервлет должен создать объект класса **Cookie**, указав конструктору имя и значение блока, и добавить их в объект-response. Конструктор использует имя блока в качестве первого параметра, а его значение – в качестве второго.

```
Cookie cookie = new Cookie("myid", "007");
response.addCookie(cookie);
```

Извлечь информацию cookie из запроса можно с помощью метода **getCookies()** объекта **HttpServletRequest**, который возвращает массив объектов, составляющих этот файл.

```
Cookie[] cookies = request.getCookies();
```

После этого для каждого объекта класса **Cookie** можно вызвать метод **getValue()**, который возвращает строку **String** с содержимым блока cookie. В данном случае этот метод вернет значение "007".

Объект **Cookie** имеет целый ряд параметров: путь, домен, номер версии, время жизни, комментарий. Одним из важнейших является срок жизни в секундах от момента первой отправки клиенту. Если параметр не указан, то cookie существует только до момента первого закрытия браузера. Для запуска следующего приложения можно использовать сервлет из примера # 1 этой главы, вставив в метод **performTask()** следующий код:

```
CookieWork.setCookie(resp); // добавление cookie
CookieWork.printToBrowser(resp, req); // извлечение cookie
```

Класс **CookieWork** имеет вид:

```
/* пример # 3 : создание и чтение cookie : CookieWork.java */
package chapt16;
import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.http.Cookie;
```

```
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

public class CookieWork {

public static void setCookie(HttpServletResponse resp) {
            String name = "Spiridonov";
            String role = "MegaAdmin";
            Cookie c = new Cookie(name, role);
            c.setMaxAge(3600);// время жизни файла
            resp.addCookie(c);
      }
public static void printToBrowser(
HttpServletResponse response, HttpServletRequest request) {
            try {
                  response.setContentType("text/html");
                  PrintWriter out = response.getWriter();

                  Cookie[] cookies = request.getCookies();
            if (cookies != null) {
                  out.print("Number cookies :"
                              + cookies.length + "<BR>");

            for (int i = 0; i < cookies.length; i++) {
            Cookie c = cookies[i];
            out.print("Secure :" + c.getSecure() + "<br>");
            out.print(c.getName() + " = " + c.getValue()
                                                + "<br>");
                                          }// end for
            }// end if
                  out.close();
            } catch (IOException e) {
                  e.printStackTrace();
                  throw new RuntimeException(e.toString());
            }
      }
}
```

В результате в файле cookie будет содержаться следующая информация:

Number cookies :1
Secure :false
Spiridonov = MegaAdmin

Файл cookie можно изменять. Для этого следует воспользоваться сервлетом из примера #1 и в метод **performTask()** добавить следующий код:

```
CookieCounter.printToBrowser(resp, req);
```

В классе **CookieCounter** производится модификация файла cookie, хранимого на компьютере клиента.

```
/* пример # 4 : создание cookie и чтение количества вызовов сервлета из cookie :
CookieCounter.java */
package chapt16;
import java.io.IOException;
import java.io.Writer;
import javax.servlet.http.Cookie;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

public class CookieCounter {
/* константа, которая будет использована для установки максимального
времени жизни cookie (здесь указано 30 дней) */
public static final int MAX_AGE_COOKIE = 3600 * 24 * 30;

      public static void printToBrowser(
HttpServletResponse response, HttpServletRequest request) {
            try {
                  Writer out = response.getWriter();
                  out.write("My Cookie counter: ");
/* устанавливает счетчик количества вызовов сервлета пользователем    */
out.write(String.valueOf(prepareCookieCounter(
                                    request,response)));
                  out.flush();
                  out.close();
            } catch (IOException e) {
                  e.printStackTrace();
            throw new RuntimeException("Failed: " + e);
            }
      }
      // обновляет в cookie счетчик обращений пользователя к сервлету
      private static int prepareCookieCounter(
HttpServletRequest request, HttpServletResponse response) {
            Cookie[] cookies = request.getCookies();
            Cookie counterCookie;
if (cookies != null) {
   for (int i = 0; i < cookies.length; i++) {
      if ("counter".equals(cookies[i].getName())) {
            String counterStr = cookies[i].getValue();
                  int counterValue;
                  try {
            counterValue = Integer.parseInt(counterStr);
                  } catch (NumberFormatException e) {
                        counterValue = 0;
                              }
      counterValue++;
      counterCookie = new Cookie("counter",
                  String.valueOf(counterValue));
      counterCookie.setMaxAge(MAX_AGE_COOKIE);
      response.addCookie(counterCookie);
```

```
                                return counterValue;
                            }//end if
                        }//end for
                    } //end if
                    counterCookie = new Cookie("counter", "1");
                    counterCookie.setMaxAge(MAX_AGE_COOKIE);
                    response.addCookie(counterCookie);
                    return 1;
            }
        }
```

В результате в файле cookie будет содержаться следующая информация:

```
counter
1
localhost/FirstProject/
1024
939371136
29793584
1067152496
29787549
*
```

В браузер будет выведено следующее сообщение:
My session counter: 1

Обработка событий

Существует несколько интерфейсов, которые позволяют следить за событиями, связанными с сеансом, контекстом и запросом сервлета, генерируемыми во время жизненного цикла Web-приложения:

- **javax.servlet.ServletContextListener** – обрабатывает события создания/удаления контекста сервлета;
- **javax.servlet.http.HttpSessionListener** – обрабатывает события создания/удаления HTTP-сессии;
- **javax.servlet.ServletContextAttributeListener** – обрабатывает события создания/удаления/модификации атрибутов контекста сервлета;
- **javax.servlet.http.HttpSessionAttributeListener** – обрабатывает события создания/удаления/модификации атрибутов HTTP-сессии;
- **javax.servlet.http.HttpSessionBindingListener** – обрабатывает события привязывания/разъединения объекта с атрибутом HTTP-сессии;
- **javax.servlet.http.HttpSessionActivationListener** – обрабатывает события связанные с активацией/дезактивацией HTTP-сессии;
- **javax.servlet.ServletRequestListener** – обрабатывает события создания/удаления запроса;
- **javax.servlet.ServletRequestAttributeListener** – обрабатывает события создания/удаления/модификации атрибутов запроса сервлета.