COM222 DESENVOLVIMENTO DE SISTEMAS WEB

Aula 08: Cookies

Conteúdo

- Cookies – como funcionam
- Enviando cookies
- Recebendo cookies
- Fazendo tracking de visitantes
- Especificando atributos em cookies
- Cookies de sessão vs. cookies persistentes
- Validação de campos de formulário

Como funciona um cookie

- Conceito básico
  - Servlet envia um par nome/valor para o cliente
  - Cliente retorna esse mesmo par nome/valor quando retorna ao mesmo site
- Uso típico
  - Identificar um usuário durante uma sessão de e-commerce
    - Servlets provêem uma API de alto nível para essa tarefa
  - Evitar o uso de username e password
  - Customizar um site
  - Facilitar o uso de propagandas direcionadas nos sites

Propaganda direcionada com cookies

Amazon.com home page for new visitor or visitor with cookies disabled.

Amazon.com home page for repeat visitor. Books shown are based on prior history.

Enviando cookies para um cliente

- Chamar o construtor da classe Cookie com um nome e um valor, ambos Strings

```
Cookie c = new Cookie("userID", "1234");
```

- Fazer o ajuste de "maximum age"
  - Informa ao browser para manter o cookie no disco e o tempo em que o mesmo deve ser mantido

```
c.setMaxAge(60*60*24*7);   // Uma semana
```

- Colocar o cookie na resposta HTTP

```
response.addCookie(c);
```

Lendo cookies de um cliente

- Chamar request.getCookies
  - Produz um array de objetos tipo cookie
- Varrer o array chamando o método getName até obter o cookie procurado

```
String cookieName = "userID";
Cookie[] cookies = request.getCookies();
if (cookies != null) {
    for(int i=0; i<cookies.length; i++) {
        Cookie cookie = cookies[i];
        if (cookieName.equals(cookie.getName())) {
            doSomethingWith(cookie.getValue());
        }
    }
}
```

Usando cookies para detectar primeira visita a um site


```
import javax.servlet.*;
import javax.servlet.http.*;
import javax.servlet.annotation.*;
import java.io.*;
import java.util.*;
@WebServlet("/repeat-visitor")
public class RepeatVisitor extends HttpServlet {
   public void doGet(HttpServletRequest request, HttpServletResponse response)
         throws ServletException, IOException {
      boolean newbie = true;
      Cookie[] cookies = request.getCookies();
      if (cookies != null) {
         for (Cookie c : cookies) {
            if ((c.getName().equals("repeatVisitor")) && (c.getValue().equals("yes"))) {
               newbie = false;
               break;
            }
         }
      }
```

Usando cookies para detectar primeira visita a um site


```
String msg;
if (newbie) {
    Cookie returnVisitorCookie = new Cookie("repeatVisitor", "yes");
    returnVisitorCookie.setMaxAge(60 * 60 * 24 * 365);
    response.addCookie(returnVisitorCookie);
    msg = "Bem-vindo a bordo";
} else {
    msg = "Bem-vindo de volta";
}
response.setContentType("text/html");
PrintWriter out = response.getWriter();
out.println("<HTML>");
out.println("<BODY>");
out.println("<H2>" + msg + "</H2>");
out.println("</BODY>");
out.println("</HTML>");
out.close();
}}
```

Métodos get/set

- getDomain/setDomain
  - Permite ler/especificar o domínio ao qual um cookie se aplica.
- getMaxAge/setMaxAge
  - Permite ler/especificar a validade de um cookie (em segundos). Se não especificado, a duração de um cookie é a mesma da seção
- getName
  - Obtém o nome de um cookie. Não há método set corresponde
    - Nome do cookie é “setado” no construtor

Métodos get/set

- getPath/setPath
  - Permite ler/especificar o caminho ao qual o cookie se aplica. Se não for especificado, o cookie é aplicado a URLs que estão dentro do diretório que contém o cookie
- getSecure/setSecure
  - Permite ler/especificar uma flag indicando se o cookie se aplica a apenas conexões SSL ou a todas conexões
- getValue/setValue
  - Permite ler/especificar o valor associado a um cookie

Cookies de sessão vs. Cookies persistentes

- Quando não se usa setMaxAge, temos, por default um cookie de sesssão
  - Cookies de sessão tem a duração da sessão
  - Quando o browser é fechado, cookies de sessão morrem
- Quando setMaxAge é usado, o cookie criado é persistente
  - Persiste pelo tempo de duração especificado

Cookies de sessão vs. Cookies persistentes


```
import javax.servlet.*;
import javax.servlet.http.*;
import javax.servlet.annotation.*;
import java.io.*;
import java.util.*;
@WebServlet("/cookie-test")
public class CookieTest extends HttpServlet {
  public void doGet(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
    for(int i=0; i<3; i++) {
      Cookie cookie = new Cookie("Session-Cookie-" + i, "Cookie-Value-S" + i);
      // No maxAge (ie maxAge = -1)
      response.addCookie(cookie);
      cookie = new Cookie("Persistent-Cookie-" + i, "Cookie-Value-P" + i);
      cookie.setMaxAge(3600);
      response.addCookie(cookie);
    }
```

Cookies de sessão vs. Cookies persistentes


```
Cookie[] cookies = request.getCookies();
    PrintWriter out = response.getWriter();
    out.println("<HTML><BODY><TABLE BORDER=2>");
    if (cookies == null) {
      out.println("<TR><TH COLSPAN=2>No cookies");
    } else {
      for (Cookie cookie : cookies) {
        out.println("<TR>"
            + " <TD>" + cookie.getName()
            + " <TD>" + cookie.getValue());
      }
    }
    out.println("</TABLE></BODY></HTML>");
  }
}
```

Cookies de sessão vs. Cookies persistentes

- Resultado de uma visita inicial a CookieTest
  - Mesmo resultado ocorre quando se fecha o browser, espera-se uma hora e revisita o servlet

Cookies de sessão vs. Cookies persistentes

- Resultado de uma visita dentro de um período de uma hora
  - Mantendo o browser aberto entre a primeira visita e as demais

Classe utilitária

Encontrar cookies a partir de nomes


```
import javax.servlet.http.*;

public class CookieUtilities {

  public static String getCookieValue(HttpServletRequest request,
      String cookieName, String defaultValue) {
    Cookie[] cookies = request.getCookies();
    if (cookies != null) {
      for (Cookie cookie : cookies) {
        if (cookieName.equals(cookie.getName())) {
          return (cookie.getValue());
        }
      }
    }
    return (defaultValue);
  }
  ...
}
```

LongLivedCookie

Para criar um cookie com duração de 1 ano


```
public class LongLivedCookie extends Cookie {

    public static final int SECONDS_PER_YEAR = 60 * 60 * 24 * 365;

    public LongLivedCookie(String name, String value) {
       super(name, value);
       setMaxAge(SECONDS_PER_YEAR);
    }
}
```

Reescrevendo RepeatVisitor
Para utilizar classes utilitárias


```
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;
import java.util.*;
import javax.servlet.annotation.*;

@WebServlet("/repeat-visitor2")
public class RepeatVisitor2 extends HttpServlet {

  public void doGet(HttpServletRequest request,
      HttpServletResponse response)
      throws ServletException, IOException {
    boolean newbie = true;
    String value = CookieUtilities.getCookieValue(request, "repeatVisitor2", "no");
    if (value.equals("yes")) {
      newbie = false;
    }
```

Reescrevendo RepeatVisitor

Para utilizar classes utilitárias


```
      String msg;
      if (newbie) {
         LongLivedCookie returnVisitorCookie
               = new LongLivedCookie("repeatVisitor2", "yes");
         response.addCookie(returnVisitorCookie);
         msg = "Bem-vindo a bordo";
      } else {
         msg = "Bem-vindo de volta";
      }
      response.setContentType("text/html");
      PrintWriter out = response.getWriter();
      out.println("<HTML>");
      out.println("<BODY>");
      out.println("<H2>" + msg + "</H2>");
      out.println("</BODY>");
      out.println("</HTML>");
      out.close();
   }
}
```

Modificando o valor de cookies

- Reusar cookies recuperados é possível, mas trabalhoso
- Melhor abordagem é, simplesmente, criar novo cookie com o mesmo nome e novo valor
  - Cookie antigo é, então, substituído pelo novo

Contagem de acessos por usuário (browser)

- Objetivo
  - Fazer tracking de usuários utilizando cookies
    - Ideia: guardar contador como valor de um cookie
    - A cada nova visita, valor do cookie é atualizado (incrementado)

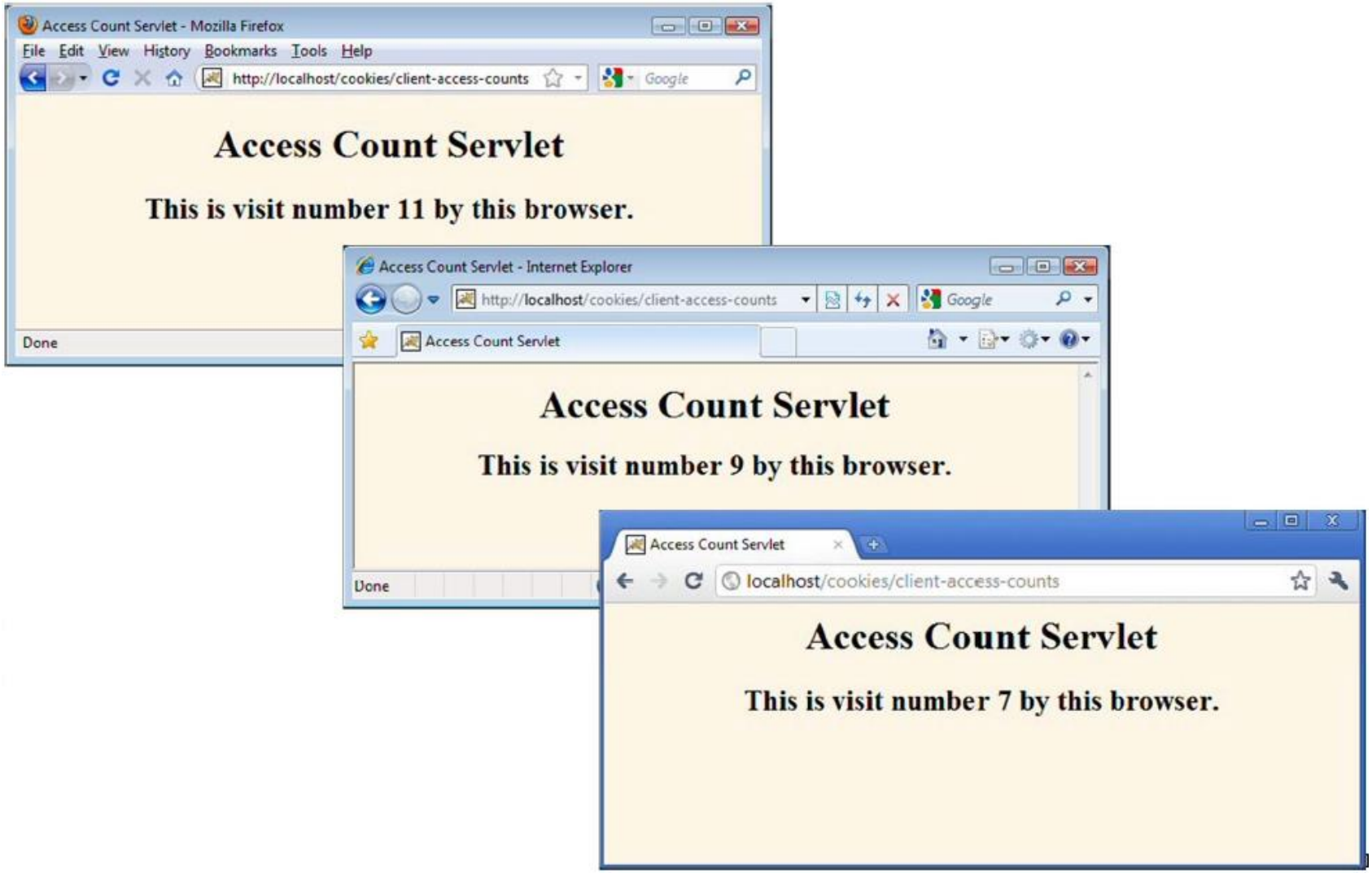
Contagem de acessos por usuário (browser)


```
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;
import java.util.*;
import javax.servlet.annotation.*;

@WebServlet("/client-access-counts")
public class ClientAccessCounts extends HttpServlet {

  public void doGet(HttpServletRequest request, HttpServletResponse response)
      throws ServletException, IOException {
    String countString = CookieUtilities.getCookieValue(request, "accessCount", "1");
    int count = 1;
    try {
      count = Integer.parseInt(countString);
    } catch (NumberFormatException nfe) {
    }
```

Contagem de acessos por usuário (browser)


```
        LongLivedCookie c = new LongLivedCookie("accessCount", String.valueOf(count + 1));
        response.addCookie(c);

        String title = new String("Servlet Contador");
        response.setContentType("text/html");
        PrintWriter out = response.getWriter();
        out.println("<HTML>"
              + "<HEAD><TITLE>" + title
              + "</TITLE></HEAD>\n"
              + "<BODY BGCOLOR=\"#FDF5E6\">\n"
              + "<CENTER>\n"
              + "<H1>" + title + "</H1>\n"
              + "<H2>Essa é a visita número  "
              + count + " desse browser.</H2>\n"
              + "</CENTER></BODY></HTML>");
    }
}
```

Contagem de acessos por usuário (browser)



Usando cookies para validar Forms

- RegistrationForm servlet
  - Usa o valor de cookies para popular campos de formulários
  - Usa valor default (string vazia) se nenhum cookie for encontrado

- Registration servlet
  - Cria cookies baseado nos parâmetros de request
  - Mostra os valores se todos os parâmetros estão presentes
  - Redirecionado para o formulário se algum parâmetro estiver faltando

RegistrationForm servlet


```
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;
import java.util.*;
import javax.servlet.annotation.*;

@WebServlet("/registration-form")
public class RegistrationForm extends HttpServlet {

    public void doGet(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        response.setContentType("text/html");
        PrintWriter out = response.getWriter();
        String actionURL = "/FormCookie/registration";
        String firstName = CookieUtilities.getCookieValue(request, "firstName", "");
        String lastName = CookieUtilities.getCookieValue(request, "lastName", "");
        String emailAddress = CookieUtilities.getCookieValue(request, "emailAddress", "");
        String title = new String("Please register");
```

RegistrationForm servlet


```
out.println("<HTML>"
            + "<HEAD><TITLE>" + title + "</TITLE></HEAD>\n"
            + "<BODY BGCOLOR=\"#FDF5E6\">\n"
            + "<CENTER>\n"
            + "<H1>" + title + "</H1>\n"
            + "<FORM METHOD=\"POST\" ACTION=\"" + actionURL + "\">\n"
            + "First Name:\n"
            + " <INPUT TYPE=\"TEXT\" NAME=\"firstName\" "
            + "VALUE=\"" + firstName + "\"><BR>\n"
            + "Last Name:\n"
            + " <INPUT TYPE=\"TEXT\" NAME=\"lastName\" "
            + "VALUE=\"" + lastName + "\"><BR>\n"
            + "Email Address: \n"
            + " <INPUT TYPE=\"TEXT\" NAME=\"emailAddress\" "
            + "VALUE=\"" + emailAddress + "\"><P>\n"
            + "<INPUT TYPE=\"SUBMIT\" VALUE=\"Register\">\n"
            + "</FORM></CENTER></BODY></HTML>");
  }
}
```

Registration servlet


```
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;
import java.util.*;
import javax.servlet.annotation.*;

@WebServlet("/registration")
public class RegistrationServlet extends HttpServlet {

   public void doPost(HttpServletRequest request, HttpServletResponse response)
         throws ServletException, IOException {
      response.setContentType("text/html");
      boolean isMissingValue = false;
      String firstName = request.getParameter("firstName");
      if (isMissing(firstName)) {
         firstName = "Missing first name";
         isMissingValue = true;
      }
```

Registration servlet


```
String lastName = request.getParameter("lastName");
    if (isMissing(lastName)) {
       lastName = "Missing last name";
       isMissingValue = true;
    }
    String emailAddress = request.getParameter("emailAddress");
    if (isMissing(emailAddress)) {
       emailAddress = "Missing eMail address";
       isMissingValue = true;
    }
    Cookie c1 = new LongLivedCookie("firstName", firstName);
    response.addCookie(c1);
    Cookie c2 = new LongLivedCookie("lastName", lastName);
    response.addCookie(c2);
    Cookie c3 = new LongLivedCookie("emailAddress", emailAddress);
    response.addCookie(c3);
    String formAddress = "/FormCookie/registration-form";
```

Registration servlet


```
if (isMissingValue) {
        response.sendRedirect(formAddress);
    } else {
        String title = new String("Thank you for registering");
        response.setContentType("text/html");
        PrintWriter out = response.getWriter();
        out.println("<HTML>"
              + "<HEAD><TITLE>" + title
              + "</TITLE></HEAD>\n"
              + "<BODY BGCOLOR=\"#FDF5E6\">\n"
              + "<CENTER>\n"
              + "<H1>" + title + "</H1>\n"
              + "<H3>\n"
              + "<P>First name: " + firstName + "\n"
              + "<P>Last name: " + lastName + "\n"
              + "<P>Email address: " + emailAddress + "\n"
              + "</CENTER></BODY></HTML>");
    }
}
```

Registration servlet


```
 private boolean isMissing(String str) {
        if (str.equalsIgnoreCase("")) {
            return true;
        }
        return false;
    }
}
```

Exercício

- Implemente o exercício 2 da aula 5 fazendo a validação dos campos do formulário através de cookies
  - Quando o formulário for preenchido corretamente, dados devem ser salvos no banco