Downsides to Preventing Use of Scripting Elements

- Harder debugging
  - <% System.out.println("...."); %>
- No redirects
  - <% response.sendRedirect("welcome.jsp"); %>
- Some techniques hard to do with MVC
  - <%
    if (outputShouldBeExcel()) {
    response.setContentType("application/vnd.ms-excel");
    }
    %>
- Just because scripting is usually bad does not mean it is *always* bad

EL Uses: Scoped vars, Bean properties, collections

Accessing Scoped Variables

- **${varName}**
  - Searches the PageContext, the HttpServletRequest, the HttpSession, and the ServletContext, *in that order*, and output the object with that attribute name. PageContext does not apply with MVC.
  - Application: if you just have an error message, you can store the String directly instead of putting it in a bean and storing the bean
- **Equivalent forms**
  - ${name}
  - <%= pageContext.findAttribute("name") %>
  - [redacted]

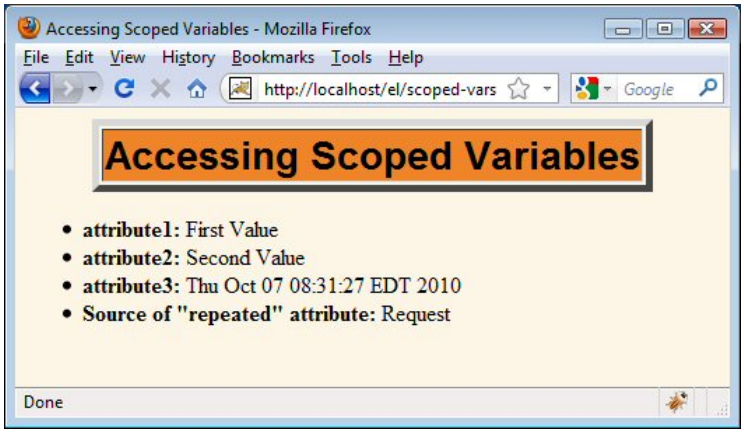
Example: Accessing Scoped Variables

```
@WebServlet("/scoped-vars")
public class ScopedVars extends HttpServlet {
  public void doGet(HttpServletRequest request,
                    HttpServletResponse response)
      throws ServletException, IOException {
    request.setAttribute("attribute1", "First Value");
    HttpSession session = request.getSession();
    session.setAttribute("attribute2", "Second Value");
    ServletContext application = getServletContext();
    application.setAttribute("attribute3",
                             new java.util.Date());
    request.setAttribute("repeated", "Request");
    session.setAttribute("repeated", "Session");
    application.setAttribute("repeated", "ServletContext");
    RequestDispatcher dispatcher =
      request.getRequestDispatcher
                    ("/WEB-INF/results/scoped-vars.jsp");
    dispatcher.forward(request, response);
  }
}
```

Example: Accessing Scoped Variables (Continued)

```
<!DOCTYPE …>
…
<TABLE BORDER=5 ALIGN="CENTER">
  <TR><TH CLASS="TITLE">
  Accessing Scoped Variables
</TABLE>
<P>
<UL>
  <LI><B>attribute1:</B> ${attribute1}
  <LI><B>attribute2:</B> ${attribute2}
  <LI><B>attribute3:</B> ${attribute3}
  <LI><B>Source of "repeated" attribute:</B>
      ${repeated}
</UL>
</BODY></HTML>
```


Example: Accessing Scoped Variables (Result)



Accessing Bean Properties

- **${varName.propertyName}**
  - Means to find scoped variable of given name and output the specified bean property
    - Remember from MVC lecture that bean property corresponds to getter method name, not instance var.
- **Equivalent forms**
  - ${customer.firstName}
  - <%@ page import="coreservlets.NameBean" %>
    <%
    NameBean person =
    (NameBean)pageContext.findAttribute("customer");
    %>
    <%= person.getFirstName() %>

Accessing Bean Properties (Continued)

- **Equivalent forms**
  - ${customer.firstName}
  - <jsp:useBean id="customer"
    type="coreservlets.NameBean"
    scope="*request, session, or application*" />
    <jsp:getProperty name="customer"
    property="firstName" />
- **This is better than script on previous slide.**
  - But, requires you to know the scope
  - And fails for subproperties.
    - No non-Java equivalent to ${customer.address.zipCode}

Example: Accessing Bean Properties

```
@WebServlet("/bean-properties")
public class BeanProperties extends HttpServlet {
  public void doGet(HttpServletRequest request,
                    HttpServletResponse response)
      throws ServletException, IOException {
    Name name = new Name("Marty", "Hall");
    Company company =
      new Company("coreservlets.com",
                  "Customized Java EE and Ajax Training");
    Employee employee =
      new Employee(name, company);
    request.setAttribute("employee", employee);
    RequestDispatcher dispatcher =
      request.getRequestDispatcher
        ("/WEB-INF/results/bean-properties.jsp");
    dispatcher.forward(request, response);
  }
}
```


Example: Accessing Bean Properties (Continued)

```
public class Employee {
  private Name name;
  private Company company;

  public Employee(Name name, Company company) {
    setName(name);
    setCompany(company);
  }

  public Name getName() { return(name); }

  public void setName(Name name) {
    this.name = name;
  }

  public CompanyBean getCompany() { return(company); }

  public void setCompany(Company company) {
    this.company = company;
  }
}
```

Example: Accessing Bean Properties (Continued)

```
public class Name {
  private String firstName;
  private String lastName;

  public Name(String firstName, String lastName) {
    setFirstName(firstName);
    setLastName(lastName);
  }

  public String getFirstName() {
    return (firstName);
  }
  public void setFirstName(String firstName) {
    this.firstName = firstName;
  }
  public String getLastName() {
    return (lastName);
  }
  public void setLastName(String lastName) {
    this.lastName = lastName;
  }
}
```

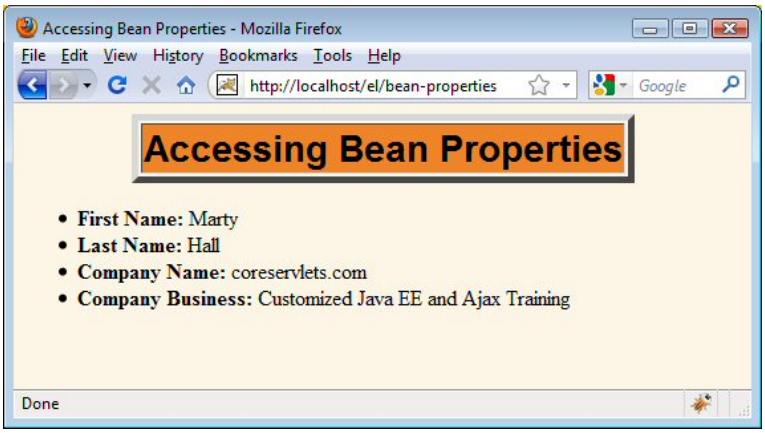

Example: Accessing Bean Properties (Continued)

```
public class Company {
  private String companyName;
  private String business;

  public Company(String companyName, String business) {
    setCompanyName(companyName);
    setBusiness(business);
  }

  public String getCompanyName() { return(companyName); }

  public void setCompanyName(String companyName) {
    this.companyName = companyName;
  }

  public String getBusiness() { return(business); }

  public void setBusiness(String business) {
    this.business = business;
  }
}
```

Example: Accessing Bean Properties (Continued)

```
<!DOCTYPE …>
…
<UL>
  <LI><B>First Name:</B>
      ${employee.name.firstName}
  <LI><B>Last Name:</B>
      ${employee.name.lastName}
  <LI><B>Company Name:</B>
      ${employee.company.companyName}
  <LI><B>Company Business:</B>
      ${employee.company.business}
</UL>
</BODY></HTML>
```


Example: Accessing Bean Properties (Result)



Equivalence of Dot and Array Notations

- **Equivalent forms**
  - ${name.property}
  - ${name["property"]}
- **Reasons for using array notation**
  - To access arrays, lists, and other collections
    - See upcoming slides
  - To calculate the property name at request time.
    - {name1[name2]} (no quotes around name2)
  - To use names that are illegal as Java variable names
    - {foo["bar-baz"]}
    - {foo["bar.baz"]}

Accessing Collections

- **${attributeName[entryName]}**
- **Works for**
  - Array. Equivalent to
    - theArray[index]
  - List. Equivalent to
    - theList.get(index)
  - Map. Equivalent to
    - theMap.get(keyName)
- **Equivalent forms (for HashMap)**
  - ${stateCapitals["maryland"]}
  - ${stateCapitals.maryland}
  - But the following is illegal since 2 is not a legal var name
    - ${listVar.2}

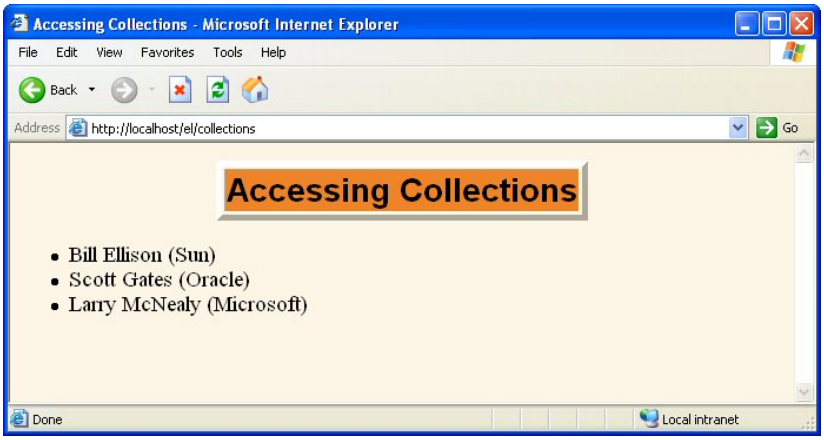
Example: Accessing Collections

```
public class Collections extends HttpServlet {
  public void doGet(HttpServletRequest request,
                    HttpServletResponse response)
      throws ServletException, IOException {
    String[] firstNames = { "Bill", "Scott", "Larry" };
    List<String> lastNames = new ArrayList<String>();
    lastNames.add("Ellison");
    lastNames.add("Gates");
    lastNames.add("McNealy");
    Map<String,String> companyNames =
      new HashMap<String,String>();
    companyNames.put("Ellison", "Sun");
    companyNames.put("Gates", "Oracle");
    companyNames.put("McNealy", "Microsoft");
    request.setAttribute("first", firstNames);
    request.setAttribute("last", lastNames);
    request.setAttribute("company", companyNames);
    RequestDispatcher dispatcher =
      request.getRequestDispatcher
                        ("/WEB-INF/results/collections.jsp");
    dispatcher.forward(request, response);
  }
```


Example: Accessing Collections (Continued)

```
<!DOCTYPE …>
…
<BODY>
<TABLE BORDER=5 ALIGN="CENTER">
  <TR><TH CLASS="TITLE">
  Accessing Collections
</TABLE>
<P>
<UL>
  <LI>${first[0]} ${last[0]} (${company["Ellison"]})
  <LI>${first[1]} ${last[1]} (${company["Gates"]})
  <LI>${first[2]} ${last[2]} (${company["McNealy"]})
</UL>
</BODY></HTML>
```

Example: Accessing Collections (Result)



Implicit Objects and Operators

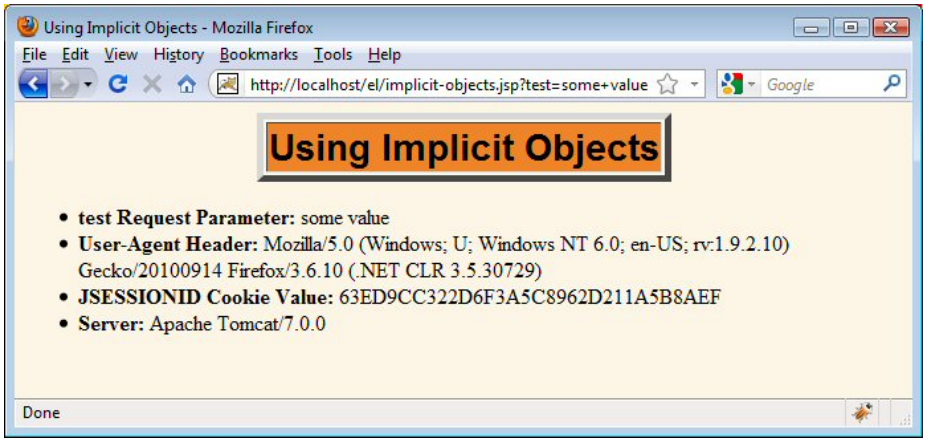
Referencing Implicit Objects (Predefined Variable Names)

- **pageContext. The PageContext object.**
  - E.g. ${pageContext.session.id}
- **param and paramValues. Request params.**
  - E.g. ${param.custID}
- **header and headerValues. Request headers.**
  - E.g. ${header.Accept} or ${header["Accept"]}
  - ${header["Accept-Encoding"]}
- **cookie. Cookie object (not cookie value).**
  - E.g. ${cookie.userCookie.value} or ${cookie["userCookie"].value}
- **initParam. Context initialization param.**
- **page[redacted], request[redacted], session[redacted], application[redacted]**
  - [redacted]
- **Problem**
  - Using implicit objects usually works poorly with MVC model

Example: Implicit Objects

```
<!DOCTYPE …>
…
<P>
<UL>
  <LI><B>test Request Parameter:</B>
      ${param.test}
  <LI><B>User-Agent Header:</B>
      ${header["User-Agent"]}
  <LI><B>JSESSIONID Cookie Value:</B>
      ${cookie.JSESSIONID.value}
  <LI><B>Server:</B>
      ${pageContext.servletContext.serverInfo}
</UL>
</BODY></HTML>
```

Example: Implicit Objects (Result)



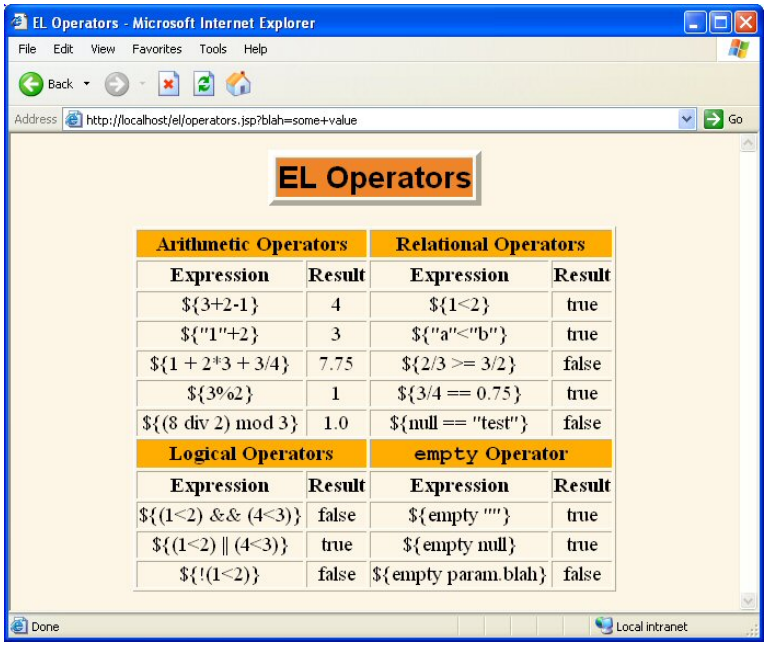
Expression Language Operators

- **Arithmetic**
  - + - * / div % mod
- **Relational**
  - == eq != ne < lt > gt <= le >= ge
- **Logical**
  - && and || or ! Not
- **Empty**
  - Empty
  - True for null, empty string, empty array, empty list, empty map. False otherwise.
- **CAUTION**
  - Use extremely sparingly to preserve MVC model

Example: Operators

```
...
<TABLE BORDER=1 ALIGN="CENTER">
  <TR><TH CLASS="COLORED" COLSPAN=2>Arithmetic Operators
      <TH CLASS="COLORED" COLSPAN=2>Relational Operators
  <TR><TH>Expression<TH>Result<TH>Expression<TH>Result
  <TR ALIGN="CENTER">
    <TD>\${3+2-1}<TD>${3+2-1}
    <TD>\${1&lt;2}<TD>${1<2}
  <TR ALIGN="CENTER">
    <TD>\${"1"+2}<TD>${"1"+2}
    <TD>\${"a"&lt;"b"}<TD>${"a"<"b"}
  <TR ALIGN="CENTER">
    <TD>\${1 + 2*3 + 3/4}<TD>${1 + 2*3 + 3/4}
    <TD>\${2/3 &gt;= 3/2}<TD>${2/3 >= 3/2}
  <TR ALIGN="CENTER">
    <TD>\${3%2}<TD>${3%2}
    <TD>\${3/4 == 0.75}<TD>${3/4 == 0.75}
...
```


Example: Operators (Result)



Evaluating Expressions Conditionally

- **${ test ? expression1 : expression2 }**
  - Evaluates test and outputs either expression1 or expression2
- **Problems**
  - Relatively weak
    - c:if and c:choose from JSTL are much better
  - Tempts you to put business/processing logic in JSP page.
  - Should only be used for presentation logic.
    - Even then, consider alternatives

Example: Conditional Expressions

```
@WebServlet("/conditionals")
public class Conditionals extends HttpServlet {
  public void doGet(HttpServletRequest request,
                    HttpServletResponse response)
      throws ServletException, IOException {
    SalesBean apples =
      new SalesBean(150.25, -75.25, 22.25, -33.57);
    SalesBean oranges =
      new SalesBean(-220.25, -49.57, 138.25, 12.25);
    request.setAttribute("apples", apples);
    request.setAttribute("oranges", oranges);
    RequestDispatcher dispatcher =
      request.getRequestDispatcher
        ("/WEB-INF/results/conditionals.jsp");
    dispatcher.forward(request, response);
  }
}
```

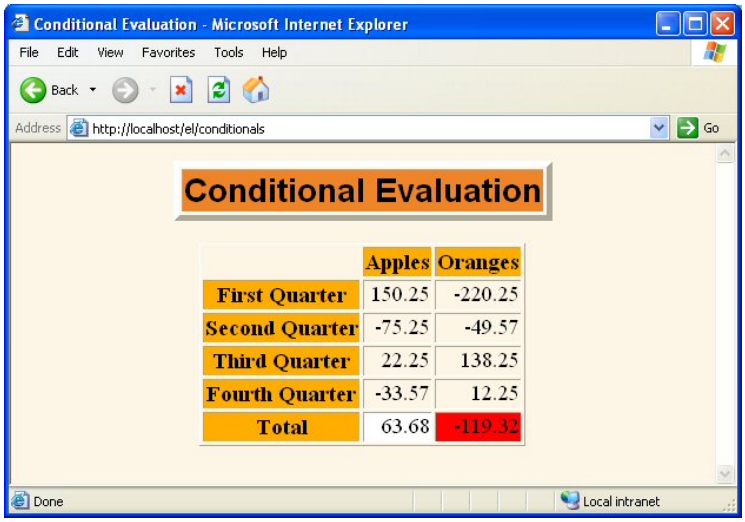
Example: Conditional Expressions (Continued)

```
public class SalesBean {
  private double q1, q2, q3, q4;

  public SalesBean(double q1Sales,
                   double q2Sales,
                   double q3Sales,
                   double q4Sales) {
    q1 = q1Sales; q2 = q2Sales;
    q3 = q3Sales; q4 = q4Sales;
  }

  public double getQ1() { return(q1); }
  public double getQ2() { return(q2); }
  public double getQ3() { return(q3); }
  public double getQ4() { return(q4); }
  public double getTotal() {
    return(q1 + q2 + q3 + q4); }
}
```


Example: Conditional Expressions (Continued)

```
…
<TABLE BORDER=1 ALIGN="CENTER">
  <TR><TH>
      <TH CLASS="COLORED">Apples
      <TH CLASS="COLORED">Oranges
  <TR><TH CLASS="COLORED">First Quarter
      <TD ALIGN="RIGHT">${apples.q1}
      <TD ALIGN="RIGHT">${oranges.q1}
  <TR><TH CLASS="COLORED">Second Quarter
      <TD ALIGN="RIGHT">${apples.q2}
      <TD ALIGN="RIGHT">${oranges.q2}
…
  <TR><TH CLASS="COLORED">Total
      <TD ALIGN="RIGHT"
          BGCOLOR="${(apples.total < 0) ? "RED" : "WHITE" }">
      ${apples.total}
      <TD ALIGN="RIGHT"
          BGCOLOR="${(oranges.total < 0) ? "RED" : "WHITE" }">
      ${oranges.total}
</TABLE>…
```

Example: Conditional Expressions (Result)



Redoing JSP 1.2 MVC Examples in JSP 2