Introduction to Java Servlets
What are Servlets?

- Designed to provide dynamic functionality to any internet server service
  - HTTP servers
  - SMTP servers
  - FTP servers

- Opposite of Applets
  - Applets extend functionality of a web browser
  - Servlets extend the functionality of an internet service (HTTP server)
What are Servlets? [cont.]

- Server-centric Java objects
  - Developed using standard OO principles
  - Synchronous in nature
  - Conceptually stateless
- Simplify server development
  - Hide protocol details
  - Hide network communication details
  - Hide server implementation
Servlet Specification

- Have own specification outside of Java EE specification
  - Current version is 2.5
  - Updated / matured independently of Java EE spec
- Specification creates WORA for internet services
- Achieved through definition of:
  - Servlet API
  - Servlet lifecycle
  - Servlet development requirements
  - Servlet packaging and configuration
The Servlet API is based on two primary packages:
- javax.servlet - Generic
- javax.servlet.http - HTTP centric

Leverages many Java SE core libraries
<table>
<thead>
<tr>
<th></th>
<th>HttpServlet</th>
<th>GenericServlet</th>
<th>Filter</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Servlet Types</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Protocol</strong></td>
<td>HTTP</td>
<td>Generic</td>
<td>Generic</td>
</tr>
<tr>
<td><strong>Server Managed State</strong></td>
<td>Y</td>
<td>Maybe</td>
<td>Maybe</td>
</tr>
<tr>
<td><strong>Client Managed State</strong></td>
<td>Y</td>
<td>Maybe</td>
<td>Maybe</td>
</tr>
<tr>
<td><strong>Development Effort</strong></td>
<td>Easy</td>
<td>More Difficult</td>
<td>Easy</td>
</tr>
<tr>
<td><strong>Managed Lifecycle</strong></td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td><strong>Error-prone</strong></td>
<td>LTD</td>
<td>Somewhat</td>
<td>LTD</td>
</tr>
<tr>
<td><strong>Run-time Configuration</strong></td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td><strong>Development Style</strong></td>
<td>Extend</td>
<td>Extend</td>
<td>Implement</td>
</tr>
<tr>
<td><strong>Chaining</strong></td>
<td>MAN</td>
<td>MAN</td>
<td>AUTO</td>
</tr>
<tr>
<td><strong>Request Access</strong></td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td><strong>Response Access</strong></td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
</tbody>
</table>
Servlet Lifecycle

- Servlets have an explicitly defined lifecycle
- Goes beyond standard object lifecycle
- Consists of 4 primary phases:
  - Creation
  - Initialization
  - Servicing requests
  - Destruction
Servlet Development Requirements

- Follow semantics of any Java object
  - Can have constructor
  - Can have member variables
  - Can have methods
  - Can use other objects and classes
  - Typically no static methods
  - Limited use of static variables

- Required yet flexible class hierarchy
  - Must be considered a `javax.servlet.Servlet`
  - Can also be specialization of other types
 Servlet Development Cycle

1. Create client; typically HTML form
2. Build servlet
   - Accept request data
   - Perform business logic
   - Return response data
3. Compile servlet
4. Package servlet
5. Deploy servlet
6. Test servlet
7. Repeat 2 - 6 until satisfied
Packaging Servlets

- Servlets are packaged within WAR files
- WAR file must contain application artifacts and configuration information
  - Servlet application artifact is a class
  - Servlet configuration is found in web.xml
- WAR filename becomes context
**Servlet Example**

**Dateservlet.java**

```java
//"include" the APIs
import javax.servlet.http.*;
import java.util.Calendar;
import java.io.*;

public class DateServlet extends HttpServlet
{
    //helper
    public void doGet(HttpServletRequest request, HttpServletResponse response) throws IOException
    {
        PrintWriter out = response.getWriter();
        String date = getDate();
        out.println("<HTML><HEAD><TITLE>Date</TITLE></HEAD>");
        out.println("<BODY>");
        out.println("The date is: " + date);
        out.println("</BODY></HTML>");
    }

    //hidden behavior
    private String getDate() {
        Calendar date = Calendar.getInstance();
        int month = date.get(Calendar.MONTH)+1;
        int day = date.get(Calendar.DAY_OF_MONTH);
        int year = date.get(Calendar.YEAR);
        return (month + "/" + day + "\"/" + year);
    }
}
```

The date is: 2/6/2004
About DevelopIntelligence

- Founded in 2003
- Provides outsourced services to learning organizations in area of software development
- Represents over 35 years of combined experience, enabling software development community through educational and performance services
- Represents over 50 years of combined software development experience
- Delivered training to over 40,000 developers worldwide
Areas of Expertise

Instruction
- Java
- J2EE
- WebServices / SOA
- Web Application Development
- Database Development
- Open Source Frameworks
- Application Servers

Courseware
- Java Application Development
- Java Web App Development
- Enterprise Java Development
- OOAD / UML
- IT Managerial
- Emerging Technologies and Frameworks
Contact Us

For more information about our services, please contact us:

- Kelby Zorgdrager
- Kelby@DevelopIntelligence.com
- 303-395-5340