CS520 Web Programming
Bits and Pieces of Web Programming

Chengyu Sun
California State University, Los Angeles

Logging

- Use print statements to assist debugging
  - Why do we want to do that when we have GUI debugger??

```
public void foo()
{
   System.out.println( "loop started" );
   // some code that might get into infinite loop
   ...
   System.out.println( "loop finished" );
}
```

Requirements of Good Logging Tools

- Minimize performance penalty
- Support different log output
  - Console, file, database, …
- Support different message levels
  - Fatal, error, warn, info, debug, trace
- Easy configuration

Java Logging Libraries

- Logging implementations
  - Log4j - http://logging.apache.org/log4j/
  - java.util.logging in JDK
- Logging API
  - Apache Commons Logging (JCL) - http://commons.apache.org/logging/
  - Simple Logging Façade for Java (SLF4J) - http://www.slf4j.org/

Choose Your Logging Libraries

- Log4j
  - Widely used
  - Good performance
  - Easy configuration
- java.util.logging
  - Part of JDK, i.e. no extra library dependency
- Commons Logging
  - Determines logging implementation at runtime
- SLF4j
  - Statically linked to a logging implementation
    - Cleaner design
    - Better performance
    - Less problem

Using Log4j and SLF4j

- Library dependencies
- Coding
  - Creating a `Logger`
  - Logging statements
- Configuration
- Output format

Log4j Configuration File

- `log4j.xml` or `log4j.properties`
- Appender
  - Output type
  - Output format
- Logger
  - Package or class selection
  - Message level

Log4j PatternLayout

- http://logging.apache.org/log4j/1.2/apidocs/org/apache/log4j/PatternLayout.html

Testing Basics

- Unit Testing
- System Testing
- Integration Testing
- User Acceptance Testing (Beta Testing)

Java Testing Frameworks

- JUnit
  - http://www.junit.org/
  - Widely used and supported
- TestNG
  - http://testng.org/
  - Technically superior to JUnit (for a while) but not as widely used or supported

What We Want From a Testing Framework

- Automation
  - Including setup and tear-down
- Grouping and selection
- Test Dependency
  - Skip tests that depend on tests that already failed
  - Run independent test in parallel
- Report
- Other, e.g. tool support, API, dependency injection, and so on

Maven Support for JUnit/TestNG

- Library dependency
- Directory structure
  - `src/test/java`
  - `src/test/resources`
- The *surefire* plugin

Basic TestNG Annotations

- `@Test`
  - Method
  - Class
- Annotations for various before/after methods

Test Dependency

- @Test
  - `dependsOnMethods`
  - `dependsOnGroups`

Test Suite

testng.xml

```
<suite name="cs520">
   <test name="all">
      <packages>
         <package name="cs520.testing" />
      </packages>
   </test>
</suite>
```

TestNG and Spring

- Test classes inherit from Spring TestNG support classes
- Specify Spring configuration file using `@ContextConfiguration`

Exercise: SpringMVC

- Use a separate test database
- Use a *Before* method to populate the test database
- Use a *After* method to clear the test databse

More About TestNG

- TestNG Documenation – http://testng.org/doc/documentation-main.html
- *Next Generation Java Testing* by Cédric Beust and Hani Suleiman

File Upload – The Form

```
<form action="FileUploadHandler"
        method="post"
        enctype="multipart/form-data">

   First file: <input type="file" name="file1" /> <br />
   Second file: <input type="file" name="file2" /> <br />

   <input type="submit" name="upload" value="Upload" />

</form>
```

File Upload – The Request

```
POST / HTTP/1.1
Host: cs.calstatela.edu:4040
[…]
Cookie: SITESERVER=ID=289f7e73912343a2d7d1e6e44f931195
Content-Type: multipart/form-data; boundary=-----------------------------146043902153
Content-Length: 509

-----------------------------146043902153
Content-Disposition: form-data; name="file1"; filename="test.txt"
Content-Type: text/plain

this is a test file.

-----------------------------146043902153
Content-Disposition: form-data; name="file2"; filename="test2.txt.gz"
Content-Type: application/x-gzip

???□□?????UC
```

Apache commons-fileupload

- http://jakarta.apache.org/commons/fileupload/using.html

```
FileItemFactory fileItemFactory = DiskFileItemFactory();
ServletFileUpload fileUpload = new ServletFileUpload( fileItemFactory );

List items = fileUpload.parseRequest( request );
for( Object o : items )
{
   FileItem item = (FileItem) items;
   if( ! item.isFormFiled() ) {...}
}
```

Spring File Upload Support

- `multipartResolver` bean
  - Support multiple request parser libraries
- Handle uploaded files
  - Add an `MultipartFile` argument to the controller method

```
@RequestMapping(value="/upload", method=RequestMethod.POST)
Public String upload( MultipartFile uploadedFile )
{
   // Process the uploaded file
   …
}
```

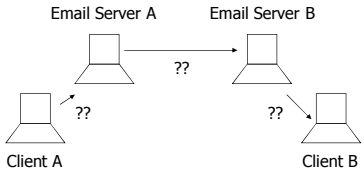
Store Uploaded Files

- In database
  - BLOB, CLOB
  - BINARY VARCAR, VARCHAR
- On disk

- *Pros and Cons??*

How Email Works

- SMTP, IMAP, POP



JavaMail

- http://java.sun.com/products/javamail/

```
Properties props = System.getProperties();
props.put("mail.smtp.host", mailhost);
Session session = Session.getInstance( props );

Message msg = new MimeMessage(session);
...
Transport.send( msg );
```

Spring Email Support

- Declare a `mailSender` bean
- Mail message classes
  - `SimpleMailMessage`
    - http://static.springsource.org/spring/docs/current/spring-framework-reference/html/mail.html#mail-usage
    - No attachment, no special character encoding
  - `MimeMailMessage`