

---

# Internet Applications

---

## Chapter 16 - Lecture Slides

---

# Internet Terminology

**URL:** <http://www.colostate.edu/~boese/Research/index.html>

**URL :**

**Protocol :**

**Domain name :**

**IP address :**

**Internet address :**

**Hyperlinks :**

---

# JEditorPane

---

---

# JEditorPane creation

- Declare a JEditorPane component.

```
JEditorPane pane;
```

- Check to see if it is null  
(first time we run the applet).  
If it is null, we instantiate it:

# JEditorPane

- Set the web page in to the JEditorPane.  
*NOTE: The URL MUST begin with http://*
- *try...catch* blocks are Java's way of handling errors which are thrown  
We need to put our code within a *try...catch* block in case there is a problem accessing the web page.

```
try
{
    pane.setPage( url );
}
catch( IOException ioe )
{
    pane.setText( "Error accessing web page: " + url );
}
```

# JEditorPane – setupURL method

- Create a method **setupURL** to make it easier for us to change the page being displayed:

```
public void setupURL( String url )
{
    if( pane == null )
        pane = new JEditorPane( );
    try
    {
        pane.setPage( url );
    } catch( IOException ioe )
    {
        pane.setText( "Error accessing web page: " + url );
    }
}
```

---

# JEditorPane

- Store the JEditorPane inside a JScrollPane, to ensure users can access the whole page being displayed

---

# JEditorPane

- Handle clicks on links inside the web page that is displayed.  
To do this, there are four things we need to do (similar to other events we've worked with).
  1. These events are inside the javax.swing.event package
  2. Specify that we are listening for these type of events –
  3. Specify that we want to listen for HyperLinkEvents on our JEditorPane component:
  4. Implement the required method for implementing the HyperlinkListener,

```
public void hyperlinkUpdate( HyperlinkEvent event )
{
    if( event.getEventType() == HyperlinkEvent.EventType.ACTIVATED )
    {
        setupURL( String.valueOf( event.getURL( ) ) );
    }
}
```

# Simple example

- see JEditorPaneEx.java



# Example with buttons

- see JEditorPaneExBtn.java



---

# Hosting applets on the Internet

---

---

# *Hosting your applet on the Web*

- Get your applet up and running on a host somewhere.
- First you need to get your applet ready without using an IDE such as Eclipse.
  
- **Setting up the HTML file**
  - Create a file (in your text editor).
  - Enter the following line, changing the XYZ to the name of your class.
  
  - This needs to be saved a file named exactly: "**index.html**" in lower case letters.
  - Modify the width and height to what you want.

# *Hosting your applet on the Web*

- **Hosting for free** (or buy a domain – see below)
  - University accounts
  - Geocities (<http://geocities.yahoo.com>) Free (but they put ads up on your pages),
  - <http://www.doteasy.com/>
  - <http://members.freewebs.com/index.jsp>
  
- **Another option: Buying a Domain Name** (FYI, not necessary)
  - Register.com
  - Registar.com
  - Aplus.net
  - others...
  
- **Uploading files**
  - Upload the **index.html**, all the **.class** files, all the **images** and **sound** files necessary to make your applet work
  - Make sure your code referencing the image file names matches the actual file name (case sensitive!! This is important as Eclipse in Windows has been lazy, allowing any case)
  - **NOTE:** if your applet reads in from a local file (not a URL) or writes to a local file (not using CGI), you need to remove that code before it will work on the Internet (see Security section).

---

# Applet Parameters

---

# Applet Parameters

- Customize applet without changing the code
- Send preferences in HTML document as
- Give a

```
<HTML>
```

```
  <BODY>
```

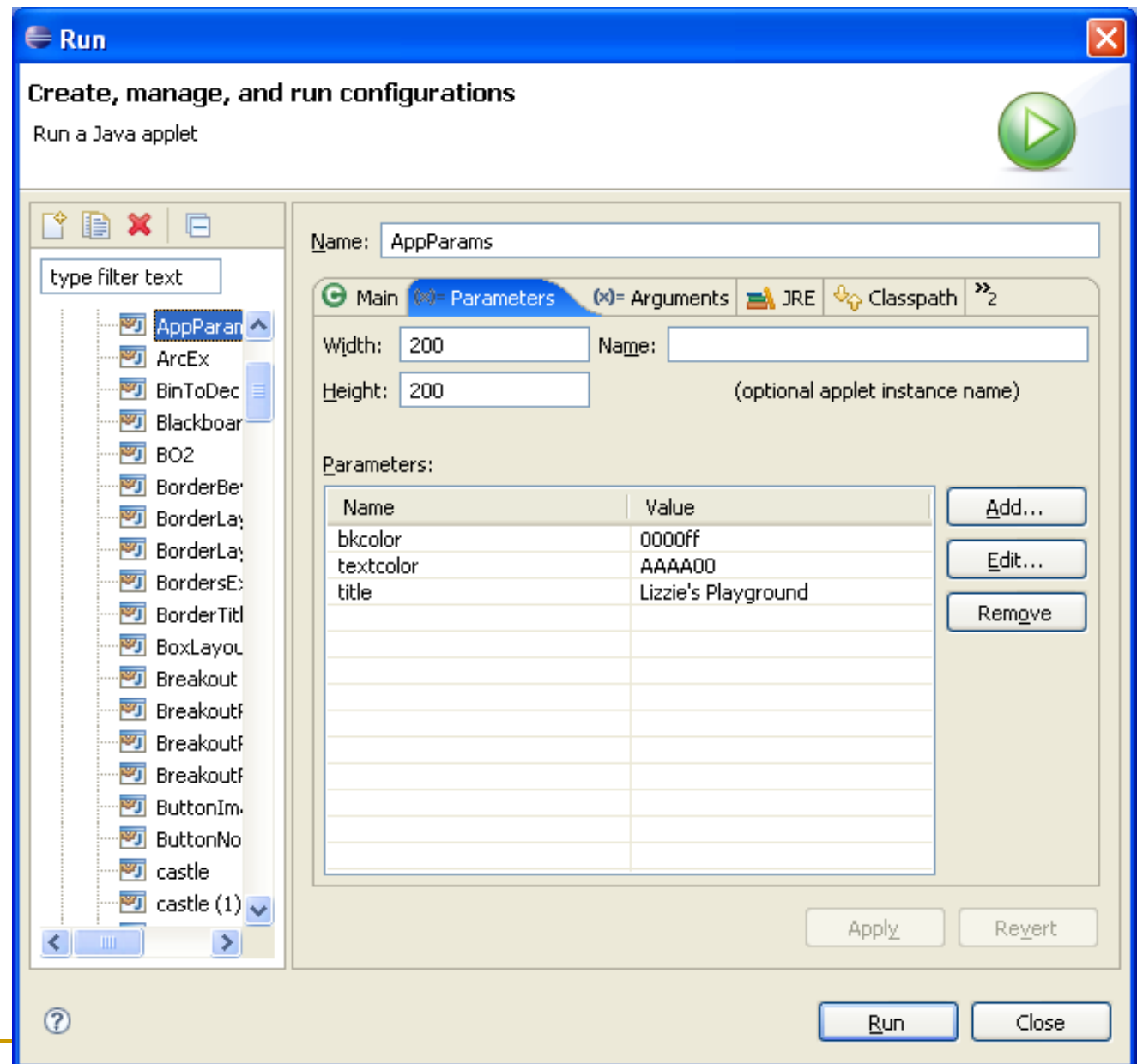
```
    <APPLET CODE=XYZ.class WIDTH=500 HEIGHT=500>
```

```
  </APPLET>Use double-quotes for text with spaces in it.
```

```
</BODY></HTML>
```

# Applet Parameters

## ■ In Eclipse



---

# Applet Parameters

- Inside your Java Code, call **getParameter(keyName )**
- Example:
  - For HTML  
**<PARAM NAME=title VALUE="Lizzie's Playground">**
  - Java Code

# Applet Parameters

- Read in values within the applet
  - `getParameter( name )`

```
import java.awt.*;
import javax.swing.*;
public class AppParams extends JApplet
{
    JPanel bkgrnd;
    JLabel heading;
    public void init( )
    {
        bkgrnd = new JPanel( );
        bkgrnd.setBackground( new Color( Integer.parseInt( getParameter( "bkcolor" ), 16 ) ) );
        heading = new JLabel( getParameter( "title" ) );
        heading.setForeground( new Color( Integer.parseInt( getParameter( "textcolor" ), 16 ) ) );
        bkgrnd.add( heading );
        setLayout( new BorderLayout( ) );
        add( bkgrnd, BorderLayout.CENTER );
    }
}
```



---

# Configuration Files

---

---

# Configuration Files

- Another way to customize the applet without changing the source code
- Store name/value pairs in a text file
- Example file:

```
title=My Favorite Playground  
bkcolor=0000FF  
textcolor=AAAA00
```

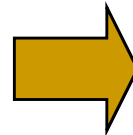
# Configuration Files

## ■ Algorithm

- Read in all the lines in the file and store in a String
- Use **split** method on **\n** to separate each line into a separate String stored in an array

```
String[ ] lines;  
lines = configFile.split( "\n" );
```

```
title=My Favorite Playground  
bkcolor=0000FF  
textcolor=AAAA00
```

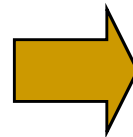


**lines**

0	title=My Favorite Playground\n
1	bkcolor=0000FF\n
2	textcolor=AAAA00\n

- Use **split** method again on **=** for each line to separate the **name** from the **value** **lines[0].split("=")**

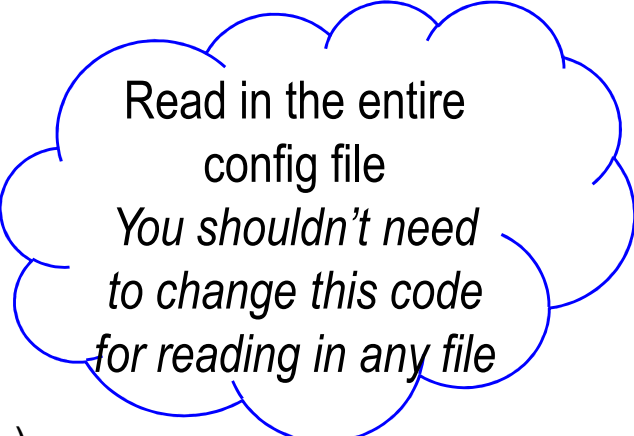
0	title=My Favorite Playground\n
1	bkcolor=0000FF\n
2	textcolor=AAAA00\n



0	title=
1	My Favorite Playground\n

# Configuration Files

```
public String readConfigFile( )
{
    String content;
    try {
        URL target = new URL( getCodeBase( ), configFilename );
        URLConnection con = target.openConnection( );
        con.connect( );
        byte b[ ] = new byte[ 1024 ];           // byte array
        int nbytes;                             // num bytes read in
        String retVal = new String( );
        BufferedInputStream in = new BufferedInputStream( con.getInputStream( ), 2048 );
        while( (nbytes = in.read( b, 0, 1024 )) != -1 ) // while there's more data to read
        {
            content = new String( b, 0, nbytes ); // get 1024 bytes of data fr file
            retVal += content;
        }
        in.close( ); // close connection
        return retVal;
    } catch ( Exception e ) { return "Error reading config file"; }
}
```



Read in the entire  
config file  
*You shouldn't need  
to change this code  
for reading in any file*

# Configuration Files

```
public void parseConfig( String cfg )
```

```
{
```

```
    String[ ] lines, tokenkey;
```

```
    lines = cfg.split( "\n" );
```

```
    for( int i=0; i<lines.length; i++ )
```

```
    {
```

```
        tokenkey = lines[i].split( "=" );
```

```
                                // remove newline at end
```

```
        String value = tokenkey[1].substring( 0, tokenkey[1].length( ) -1 );
```

```
        if ( tokenkey[0].equals( "bkcolor" ) )
```

```
            bgcolor = new Color( Integer.parseInt( value, 16 ) );
```

```
        else if ( tokenkey[0].equals( "textcolor" ) )
```

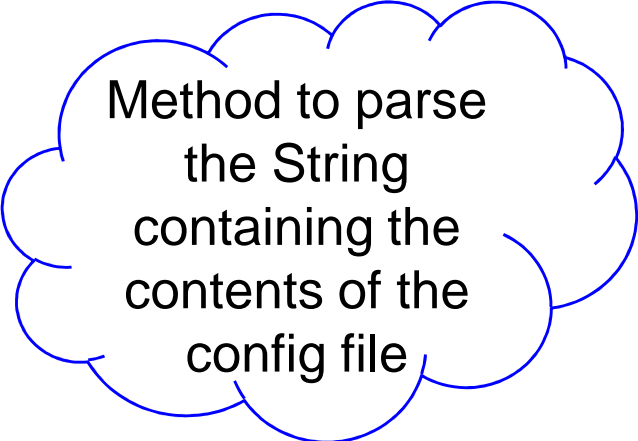
```
            textcolor = new Color( Integer.parseInt( value, 16 ) );
```

```
        else if ( tokenkey[0].equals( "title" ) )
```


```
            titletext = value;
```

```
    }
```

```
}
```

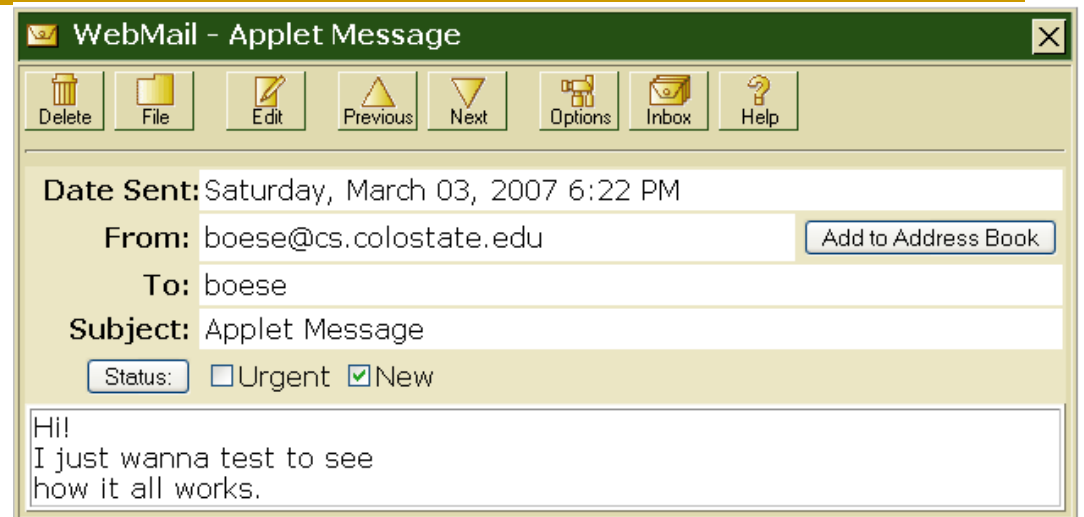
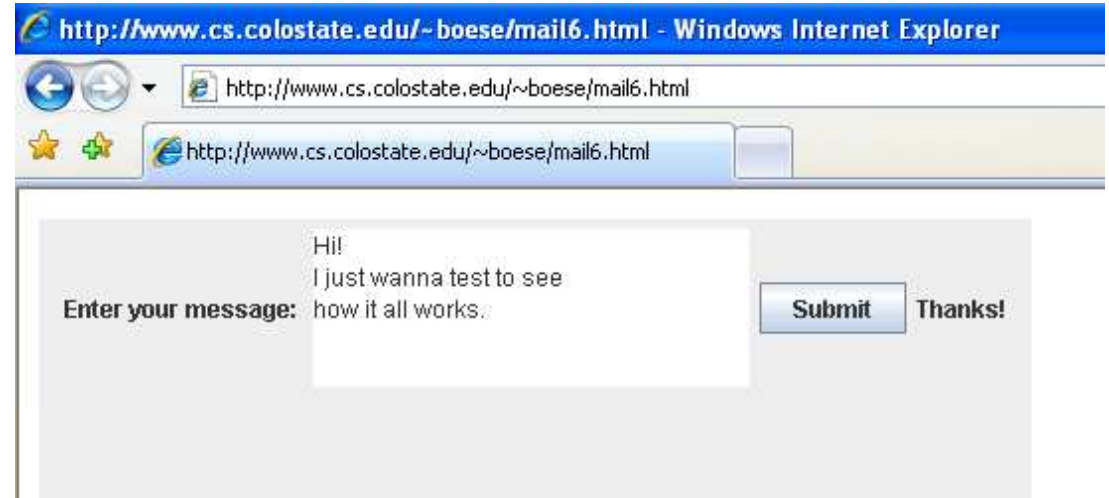


Method to parse  
the String  
containing the  
contents of the  
config file



Customize your  
applet based on  
the token and  
value pairs

# Applets and Email



---

# Applets and Email

- 
- 
- 
- 
- **Example:**
  - Requires access to perl and sendmail on UNIX/Linux system

---

# Applets and Email

- Store perl file in appropriate web site area
- 
- Depending on setup, may need to ask administrator:
  - Full path to perl
  - Full path to sendmail program
  - Location for CGI files (e.g. in public\_html/cgi-bin directory)
  - Required filename postfix
    - End with .pl or .cgi

# Applets and Email

```
#!/usr/bin/perl
```

```
use CGI ':standard';
```

```
my $outfile = "mailing.out";
```

```
my $sendmail = "/usr/sbin/sendmail -t";
```

```
my $reply_to = "Reply-to: boese@cs.colostate.edu\n";
```

```
my $subject = "Subject: Applet Message\n";
```

```
my $to = "To: boese@cs.colostate.edu\n";
```

```
my $content = param('message');
```

```
print header; # required first set of lines
```

```
print "Thanks!";
```

```
open(SENDMAIL, "|$sendmail") or die "Cannot open $sendmail: $!";
```

```
print SENDMAIL $reply_to;
```

```
print SENDMAIL $subject;
```

```
print SENDMAIL $to;
```

```
print SENDMAIL "Content-type: text/plain\n\n";
```

```
print SENDMAIL $content;
```

```
close(SENDMAIL);
```

---

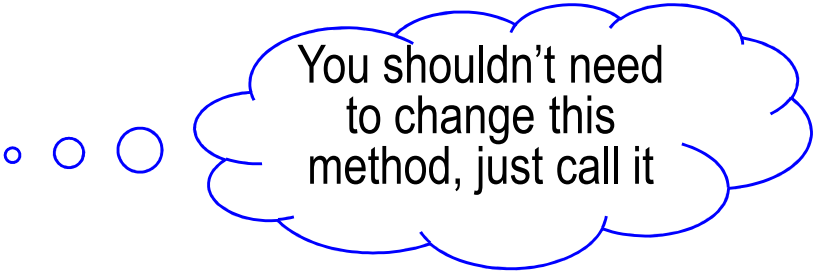
# Applets and Email

- Inside Java applet
- Write a method `sendMail` takes the URL to the cgi program
  - URL ends with **?** then the message string **?message=String**

```
String s = sendMail( "http://www.cs.colostate.edu/~boese/cgi-bin/mailer.cgi?message=",message );
```

# Applets and Email

```
public String sendMail( String fullPath, String msg )
{
    try {
        String enc = URLEncoder.encode(msg, "UTF-8");
        URL target = new URL( fullPath + enc );
        String content;
        URLConnection con = target.openConnection(); // open connection
        con.setUseCaches( false );
        con.setDefaultUseCaches( false );
        byte b[ ] = new byte[ 1024 ]; // byte array
        int nbytes; // num bytes read in
        String retVal = new String();
        BufferedInputStream in = new BufferedInputStream( con.getInputStream(), 2048 );
        while( (nbytes = in.read( b, 0, 1024 )) != -1 ) // while there's more data to read
        {
            content = new String( b, 0, nbytes ); // get 1024 bytes of data fr file
            retVal += content;
        }
        in.close(); // close connection
        return retVal;
    }
    catch( Exception e ) { return "Error " + e.toString(); }
}
```



You shouldn't need to change this method, just call it

---

# Writing Files

---

---

# Writing Applets

- 
- To record information (e.g. results from a poll, user information, etc.) send information to a CGI program

---

# Writing Applets

```
#!/usr/bin/perl
use CGI ':standard';
my $outfile = "mailing.out";
print header;      # required first set of lines

# Write to file
open(OUT, ">>$outfile") or print("Couldn't open $outfile: $!");
if (param('message'))
{
    print OUT param('message'), "\n";
}
close(OUT);
print "Thanks!";
```

---

# Summary

- JEditorPane
- Hosting Applets
- Applet Parameters
- Configuration Files
- Applets and Emails
- Writing to Files