

# XPages Development 2

XPages, Java, EL, and Source Control

A Legacy Notes Developer's Journey into Madness

Devin S. Olson

## narcissism – Devin S. Olson



- Grand Rapids, MI (USA)
- Married with 2 children
- Developing software for over 25 years (have actually coded using punch cards)
- CNA, MCP, PCLP (SA&AD)  
R4 – R7, lots of other TLAs

## narcissism – Devin S. Olson



- Instructor, Consultant, IBM Business Partner, Customer, Author
- Czarnowski Display Services
- Installing & Configuring Domino 9 on CentOS 6
- Anheuser-Busch Certified Brewmaster

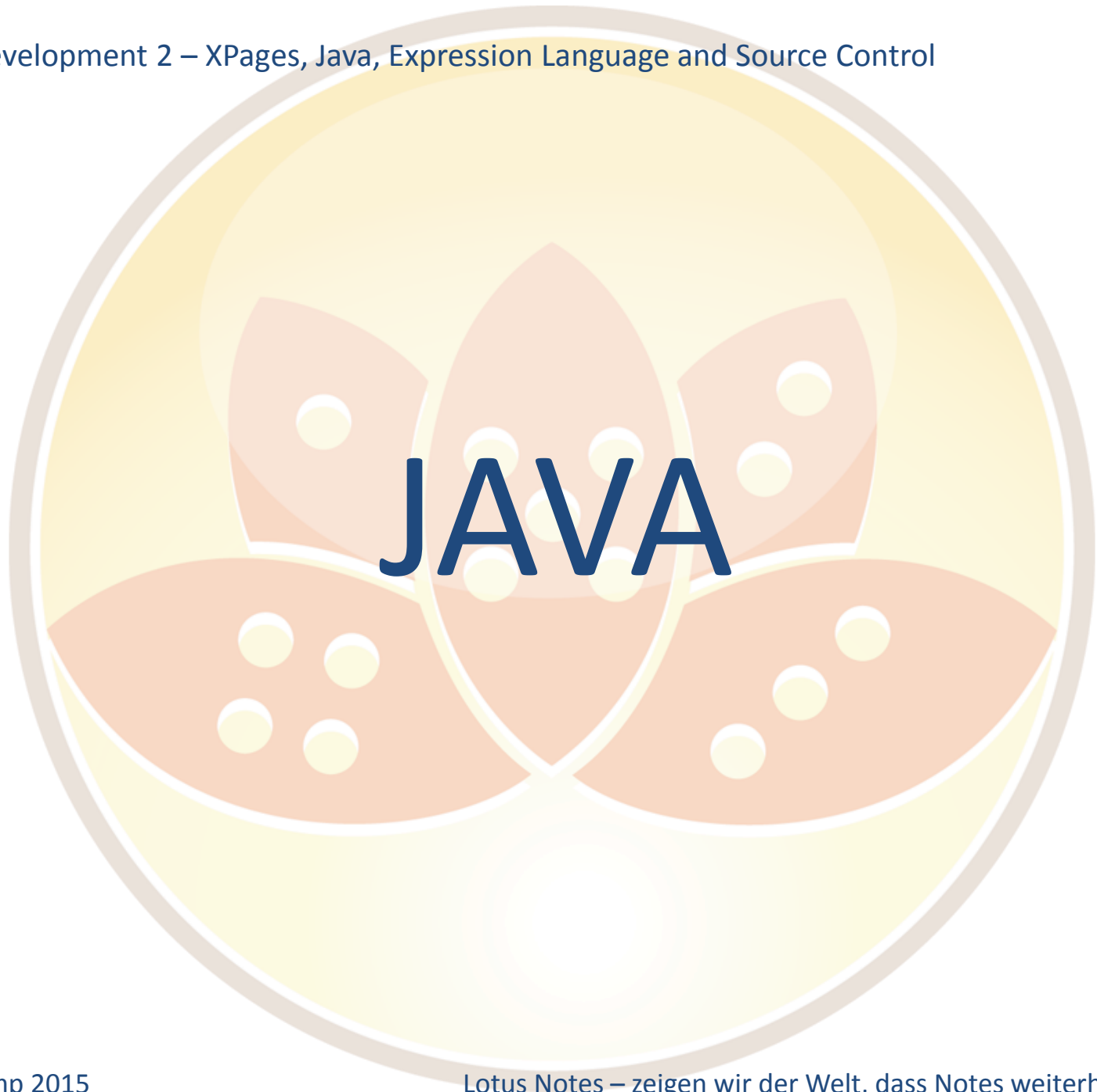
## Target Audience

- Experienced Notes & Domino Developers
- Some familiarity with web development
- Some familiarity with XPages
- Attended Yesterday's session
- Masochists?

# Agenda

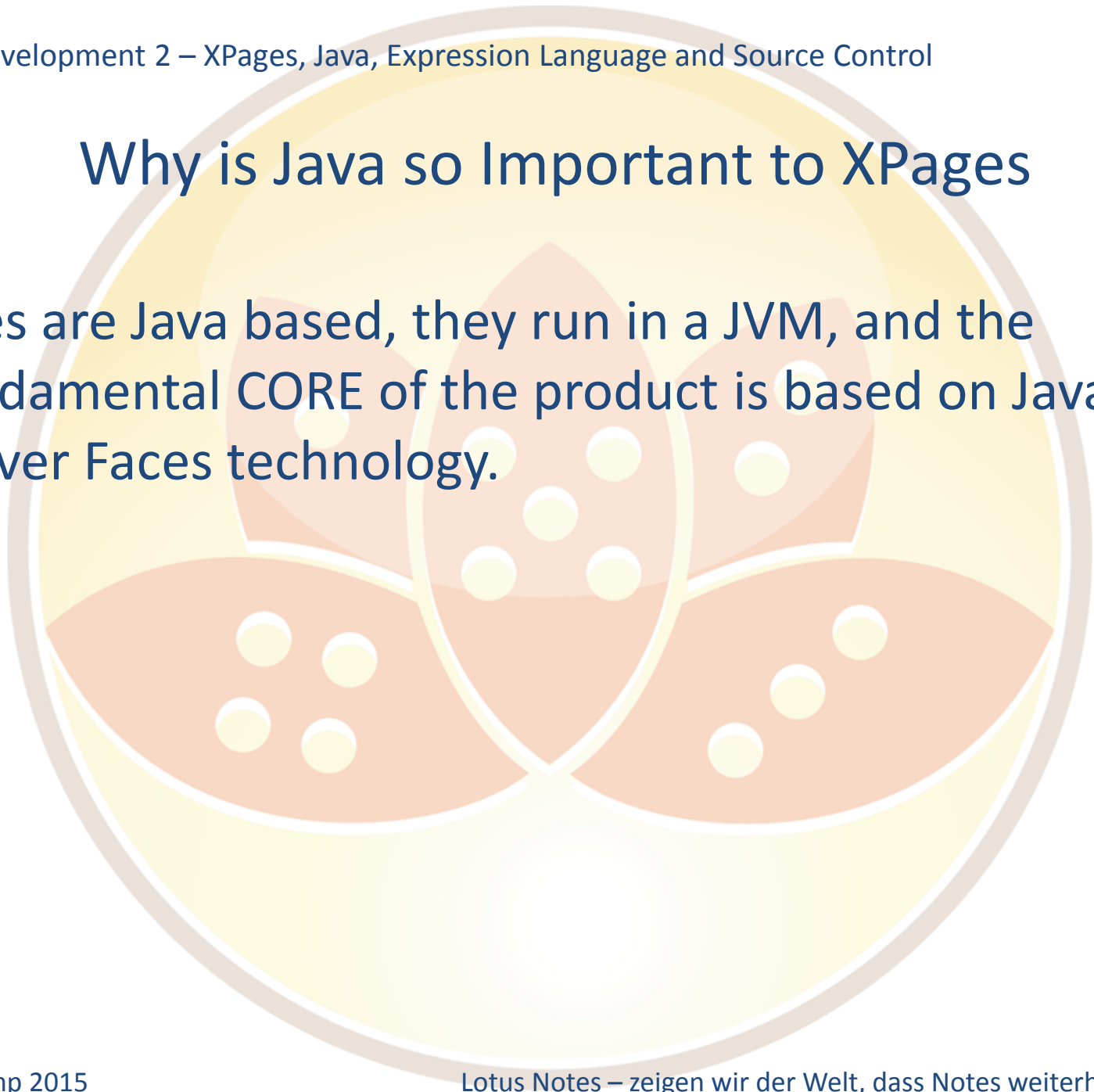
- Why Java is so Important to XPages
- Source Control
- Custom Controls
- Implementing Java in an XPages Application
- The Most Important Control You Will Ever Use
- The Power of EL (Expression Language)
- Putting it All Together





## Why is Java so Important to XPages

XPages are Java based, they run in a JVM, and the fundamental CORE of the product is based on Java Server Faces technology.



## Why is Java so Important to XPages

Because XPages run in a JVM, EVERYTHING is Java.  
The Java you write is compiled into byte code.

EVERYTHING else you write is CONVERTED into Java, and  
then compiled into byte code.



## Why is Java so Important to XPages

All that cool SSJS and XML you write, all those neat controls you drag onto your XPage, **EVERYTHING** you do must be converted into Java.

IBM did an awesome job putting all this black magic real-time interpretation / language translation / object mapping uber cool mojo together, and yes it works extremely well, but it still takes up time and resources.



# SOURCE CONTROL



SOURCE  
**HAVE YOU EVER**  
CONTROL

- Lost any code?
- Made a code change and later realized it was a mistake?
- Wanted to see the difference between different versions of your code?
- Wanted to verify that any particular change either broke or fixed something?
- Hade more than 1 version of a template?
- Lost code or had a backup that was too old?



SOURCE  
**HAVE YOU EVER**  
CONTROL

- Wanted to change somebody else's code?
- Wanted to share your code?
- Wanted to let others work on your code?
- Wanted to know where the code “hot spots” were at any time during a project?
- Wanted to sandbox a project so you can test out potential features?
- Wanted to work with anybody else?
- Lost code?



# SOURCE CONTROL

A large, stylized Lotus logo is centered on the slide. It features a yellow circle with a brown border. Inside the circle is a stylized lotus flower with five petals in a light orange color. Each petal contains several small yellow circles, resembling seeds or bubbles. The text 'SOURCE CONTROL' is written in a bold, dark blue, sans-serif font across the center of the lotus flower.



**SOURCE  
CONTROL  
BECAUSE YOU  
MUST**

## Source Control

DEMONSTRATION

The screenshot displays the Atlassian Source Control interface. The top section shows a commit history table with columns for Graph, Description, and Date. The bottom section shows the details of a specific revision (754) and a code diff for the file `nsf.emp.main\Code\Java\com\czarnowski\emp\Currer`.

Graph	Description	Date
	flow: Merged <feature> 'Portal Update Feb 23 2015' to <develop> ('develop').	23 Feb 2015 21:17
	feature/Portal Update Feb 23 2015 flow: Closed <feature> 'Portal Update Feb 23 2015'.	23 Feb 2015 21:17
	Merge	23 Feb 2015 21:14
	Merge	23 Feb 2015 20:54
	flow: Created branch 'feature/Portal Update Feb 23 2015'.	23 Feb 2015 18:55
	flow: Merged <feature> 'Portal update Feb 18 2015' to <develop> ('develop').	23 Feb 2015 18:51
	feature/Portal update Feb 18 2015 flow: Closed <feature> 'Portal update Feb 18 2015'.	23 Feb 2015 18:51
	Error message text option added. Also removed required option for Status and Submitted Date fields.	23 Feb 2015 18:51
	cc_DisplayErrors custom control added to XPage. Also, status label and field added to XPage without using	23 Feb 2015 18:50
	Field for error message added to form.	23 Feb 2015 18:48
	No longer needed. Replaced with cc_DisplayErrors custom control.	23 Feb 2015 18:48
	No longer needed. Link to original added to cc_DisplayErrors custom control.	23 Feb 2015 18:47
	Displays single message and highlights empty required fields.	23 Feb 2015 18:46
	Renamed custom control that lists the error messages. Not used as of Feb 23 2015 but maybe use later.	23 Feb 2015 18:45
	Added custom control to configurable fields without a required option.	23 Feb 2015 18:45
	Added switch to show/hide pricing.	23 Feb 2015 18:42
	flow: Created branch 'feature/Portal update Feb 18 2015'.	18 Feb 2015 8:50
	Debugging czarcache issues.	23 Feb 2015 20:58

**Revision:** 754  
**Changeset:** ef5c3bb582d482f5863eb8f8c7729cb39a6c1a5b [ef5c3bb582d4]  
**Parents:** 747, 753  
**Author:** Devin S. Olson <dolson@czarnowski.com>  
**Date:** Thursday, February 26, 2015 11:03:20 PM  
**Branch:** develop  
**Labels:** tip

**File Status** | **Log / History** | **Search**

**Code Diff:** nsf.emp.main\Code\Java\com\czarnowski\emp\Currer

Hunk 1: Lines 20-26

```
20 20 import org.openntf.domino.utils.DominoUtils;  
21 21 import org.openntf.domino.utils.Factory;  
22 22 import org.openntf.domino.utils.Strings;  
23 + import org.openntf.domino.utils.XSPUtil;  
23 24  
24 24 import com.czarnowski.base.util.CzarDebug;  
25 25 import com.czarnowski.base.util.CzarUtils;  
25 26
```

Hunk 2: Lines 584-634

```
583 584 }  
584 585
```

Atlassian



# CUSTOM CONTROLS

## Custom Controls

- Similar in THEORY to Subforms in that they are contained within an XPage as a Subform is contained within a Form.
- Break up our XPage objects into more manageable units.
- Create self-contained nodes that can be added to multiple XPages (or Custom Controls)
- Can be repeated multiple times (even on same XPage)
- Genericize our code for better re-use.

# Custom Controls

```
<xp:panel
  style="margin:1em;">
  <xp:panel style="font-size:16pt;font-family:sans-serif;font-weight:bold;margin-bottom:1.0em"> Photos </xp:panel>
  <xp:panel style="margin:1em;">
    <xp:button value="Add Photo" id="button1">
      <xp:eventHandler event="onclick" submit="true" refreshMode="complete">
        <xp:this.action>
          <xp:openPage name="/photo.xsp" target="newDocument" />
        </xp:this.action>
      </xp:eventHandler>
    </xp:button>
  </xp:panel>
  <xp:viewPanel value="#{view1}" id="viewPanel1" pageName="/photo.xsp">
    <xp:this.facets>
      <xp:pager partialRefresh="true" layout="Previous Group Next" p:key="headerPager" id="pager1" />
    </xp:this.facets>
    <xp:viewColumn columnName="Subject" id="viewColumn1" displayAs="link" openDocAsReadonly="true">
      <xp:this.facets>
        <xp:viewColumnHeader value="Subject" xp:key="header" id="viewColumnHeader1" />
      </xp:this.facets>
    </xp:viewColumn>
    <xp:viewColumn columnName="Description" id="viewColumn2">
      <xp:this.facets>
        <xp:viewColumnHeader value="Description" xp:key="header" id="viewColumnHeader2" />
      </xp:this.facets>
    </xp:viewColumn>
  </xp:viewPanel>
</xp:panel>

<xc:photos_1 />
```



## Custom Controls

```
<xp:panel
  style="margin:1em;"
  <xp:panel style="font-size:16pt;font-family:sans-serif;font-weight:bold;margin-bottom:1.0em"> Photos </xp:panel>
  <xp:panel style="margin:1em;"
    <xp:button value="Add Photo" id="button1">
      <xp:eventHandler event="onclick" submit="true" refreshMode="complete">
        <xp:this.action>
          <xp:openPage name="/photo.xsp" target="newDocument" />
        </xp:this.action>
      </xp:eventHandler>
    </xp:button>
  </xp:panel>
  <xp:viewPanel value="#{view1}" id="viewPanel1" pageName="/photo.xsp">
    <xp:this.facets>
      <xp:pager partialRefresh="true" layout="Previous Group Next" p:key="headerPager" id="pager1" />
    </xp:this.facets>
    <xp:viewColumn columnName="Subject" id="viewColumn1" displayAs="link" openDocAsReadOnly="true">
      <xp:this.facets>
        <xp:viewColumnHeader value "Subject" xp:key "header" id "viewColumnHeader1" />
      </xp:this.facets>
    </xp:viewColumn>
    <xp:viewColumn columnName="Description" id="viewColumn2">
      <xp:this.facets>
        <xp:viewColumnHeader value "Description" xp:key "header" id "viewColumnHeader2" />
      </xp:this.facets>
    </xp:viewColumn>
  </xp:viewPanel>
</xp:panel>

<xc:photos 1 />
```

**replace all of this**

## Custom Controls

```
<xp:panel
  style="margin:1em;"
  <xp:panel style="font-size:16pt;font-family:sans-serif;font-weight:bold;margin-bottom:1.0em"> Photos </xp:panel>
  <xp:panel style="margin:1em;"
    <xp:button value="Add Photo" id="button1">
      <xp:eventHandler event="onclick" submit="true" refreshMode="complete">
        <xp:this.action>
          <xp:openPage name="/photo.xsp" target="newDocument" />
        </xp:this.action>
      </xp:eventHandler>
    </xp:button>
  </xp:panel>
  <xp:viewPanel value="#{view1}" id="viewPanel1" pageName="/photo.xsp">
    <xp:this.facets>
      <xp:pager partialRefresh="true" layout="Previous Group Next" p:key="headerPager" id="pager1" />
    </xp:this.facets>
    <xp:viewColumn columnName="Subject" id="viewColumn1" displayAs="link" openDocAsReadOnly="true">
      <xp:this.facets>
        <xp:viewColumnHeader value "Subject" xp:key "header" id "viewColumnHeader1" />
      </xp:this.facets>
    </xp:viewColumn>
    <xp:viewColumn columnName="Description" id="viewColumn2">
      <xp:this.facets>
        <xp:viewColumnHeader value "Description" xp:key "header" id "viewColumnHeader2" />
      </xp:this.facets>
    </xp:viewColumn>
  </xp:viewPanel>
</xp:panel>
```

**replace all of this**

**<xc: photos\_1 />**

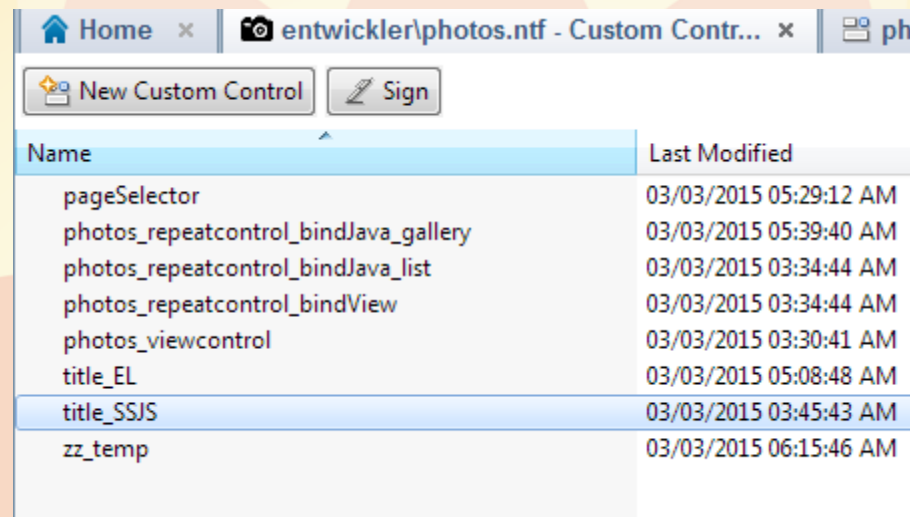
**with this**

# Custom Controls

```
<xp:panel
  style="margin:1em;">
  <xp:panel style="font-size:16pt;font-family:sans-serif;font-weight:bold;margin-bottom:1.0em"> Photos </xp:panel>
  <xp:panel style="margin:1em;">
    <xp:button value="Add Photo" id="button1">
      <xp:eventHandler event="onclick" submit="true" refreshMode="complete">
        <xp:this.action>
          <xp:openPage name="/photo.xsp" target="newDocument" />
        </xp:this.action>
      </xp:eventHandler>
    </xp:button>
    <xp:panel>
      <xp:viewPanel value="#view1" id="viewPanel1" pageName="/photo.xsp">
        <xp:this.facets>
          <xp:pager partialRefresh="true" layout="Previous Group Next" xp:key="header" id="pager1" />
        </xp:this.facets>
        <xp:viewColumn columnName="Subject" id="viewColumn1" displayAs="link" openDocAsReadonly="true">
          <xp:this.facets>
            <xp:viewColumnHeader value="Subject" xp:key="header" id="viewColumnHeader1" />
          </xp:this.facets>
        </xp:viewColumn>
        <xp:viewColumn columnName="Description" id="viewColumn2">
          <xp:this.facets>
            <xp:viewColumnHeader value="Description" xp:key="header" id="viewColumnHeader2" />
          </xp:this.facets>
        </xp:viewColumn>
      </xp:viewPanel>
    </xp:panel>
  </xp:panel>
</xc:photos_1 />
```

DEMONSTRATION

# Custom Controls



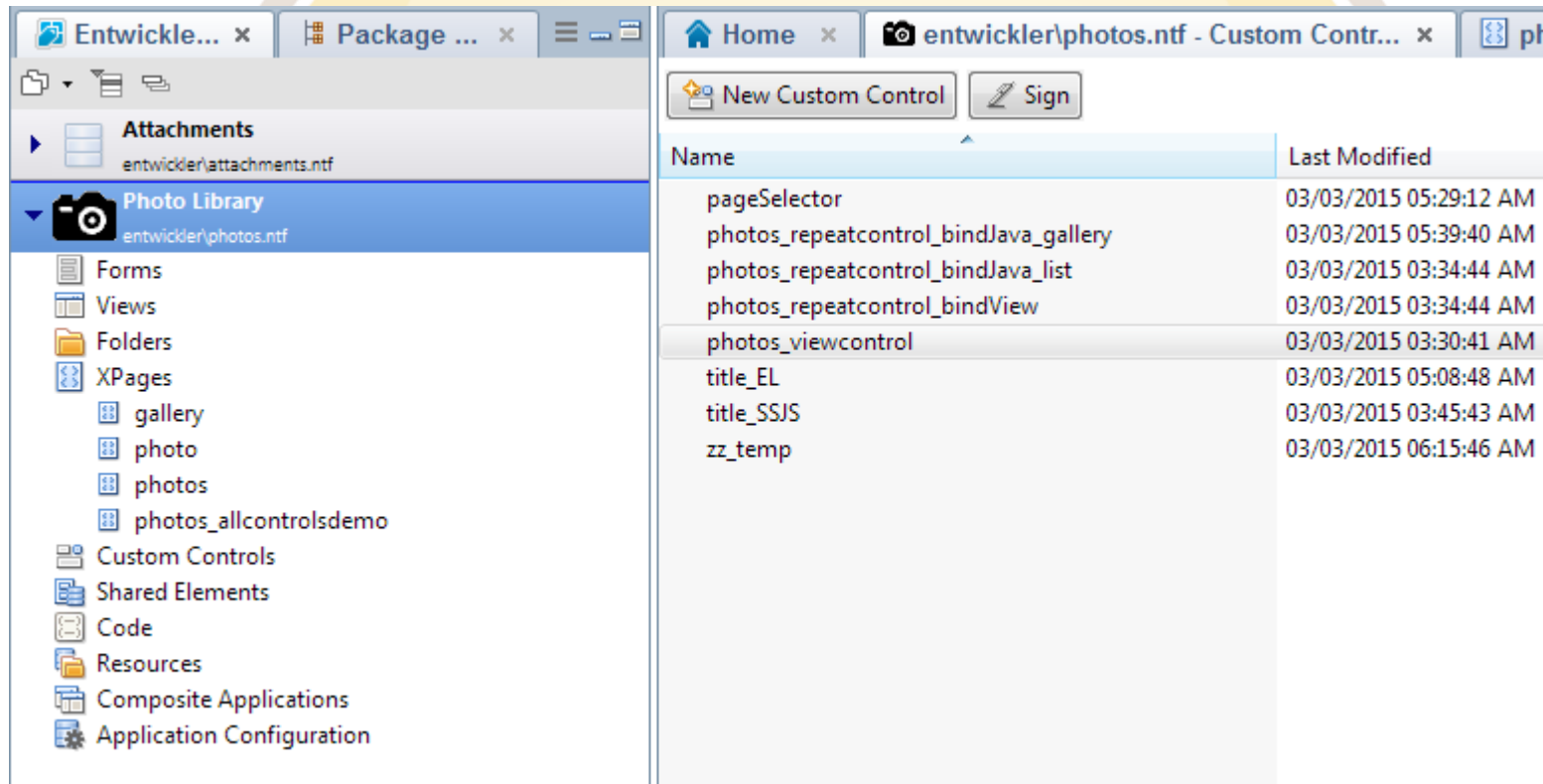
Name	Last Modified
pageSelector	03/03/2015 05:29:12 AM
photos_repeatcontrol_bindJava_gallery	03/03/2015 05:39:40 AM
photos_repeatcontrol_bindJava_list	03/03/2015 03:34:44 AM
photos_repeatcontrol_bindView	03/03/2015 03:34:44 AM
photos_viewcontrol	03/03/2015 03:30:41 AM
title_EL	03/03/2015 05:08:48 AM
title_SSJS	03/03/2015 03:45:43 AM
zz_temp	03/03/2015 06:15:46 AM

# Custom Controls



```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <xp:view
3     xmlns:xp="http://www.ibm.com/xsp/core">
4     <xp:panel
5         rendered="#{javascript:!@IsBlank(compositeData.title)}"
6         style="margin:0; margin-top:2em;">
7         <xp:text
8             escape="true"
9             id="title1"
10            tagName="h1"
11            value="#{javascript:compositeData.title}" />
12     </xp:panel>
13 </xp:view>
14
```

# Custom Controls

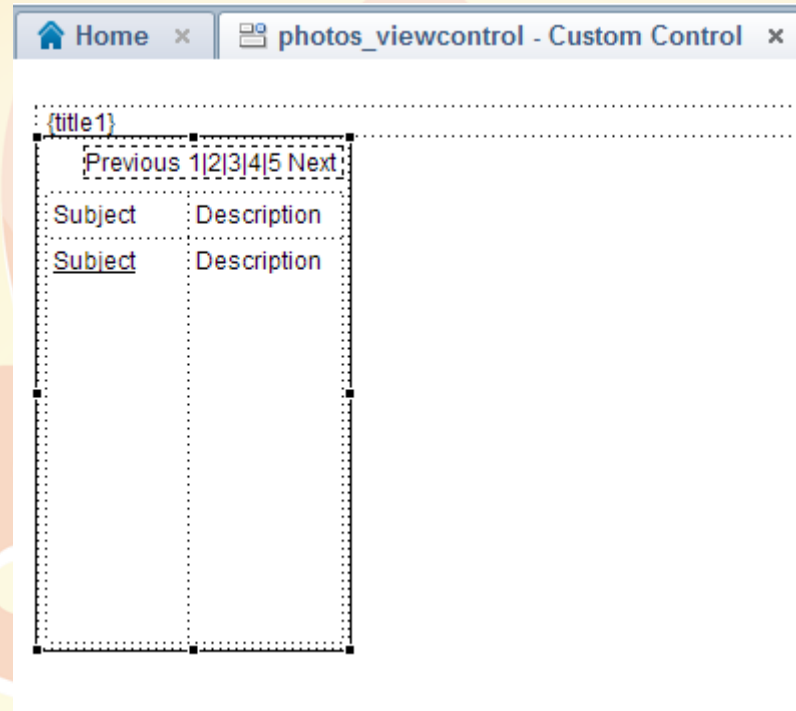


The screenshot shows the Lotus Notes interface. On the left, the 'Photo Library' folder is selected, showing a list of sub-items: Forms, Views, Folders, XPages, gallery, photo, photos, photos\_allcontrolsdemo, Custom Controls, Shared Elements, Code, Resources, Composite Applications, and Application Configuration. On the right, a table lists custom controls with their names and last modified dates.

Name	Last Modified
pageSelector	03/03/2015 05:29:12 AM
photos_repeatcontrol_bindJava_gallery	03/03/2015 05:39:40 AM
photos_repeatcontrol_bindJava_list	03/03/2015 03:34:44 AM
photos_repeatcontrol_bindView	03/03/2015 03:34:44 AM
photos_viewcontrol	03/03/2015 03:30:41 AM
title_EL	03/03/2015 05:08:48 AM
title_SSJS	03/03/2015 03:45:43 AM
zz_temp	03/03/2015 06:15:46 AM



# Custom Controls

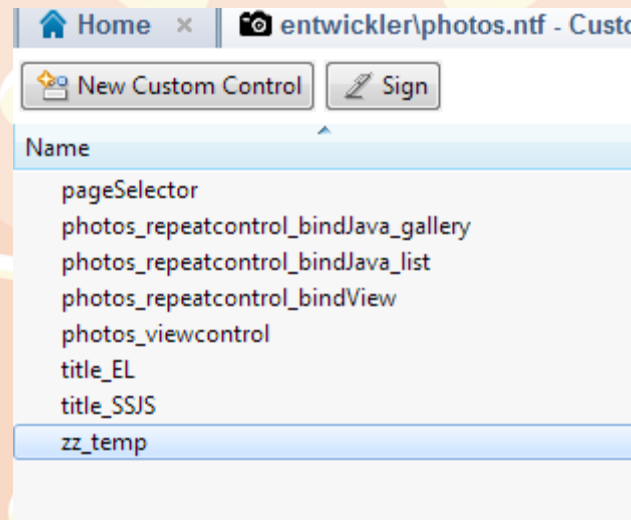


# Custom Controls

```
<xp:viewPanel
  value="#{view1}"
  id="viewPanel1"
  pageName="/photo.xsp">
  <xp:this.facets>
    <xp:pager
      partialRefresh="true"
      layout="Previous Group Next"
      xp:key="headerPager"
      id="pager1" />
  </xp:this.facets>
  <xp:viewColumn
    columnName="Subject"
    id="viewColumn1"
    displayAs="Link"
    openDocAsReadonly="true">
    <xp:this.facets>
      <xp:viewColumnHeader
        value="Subject"
        xp:key="header"
        id="viewColumnHeader1" />
    </xp:this.facets>
  </xp:viewColumn>
</xp:viewPanel>
```

```
<xp:viewColumn
  columnName="Description"
  id="viewColumn2">
  <xp:this.facets>
    <xp:viewColumnHeader
      value="Description"
      xp:key="header"
      id="viewColumnHeader2" />
  </xp:this.facets>
</xp:viewColumn>
</xp:viewPanel>
```

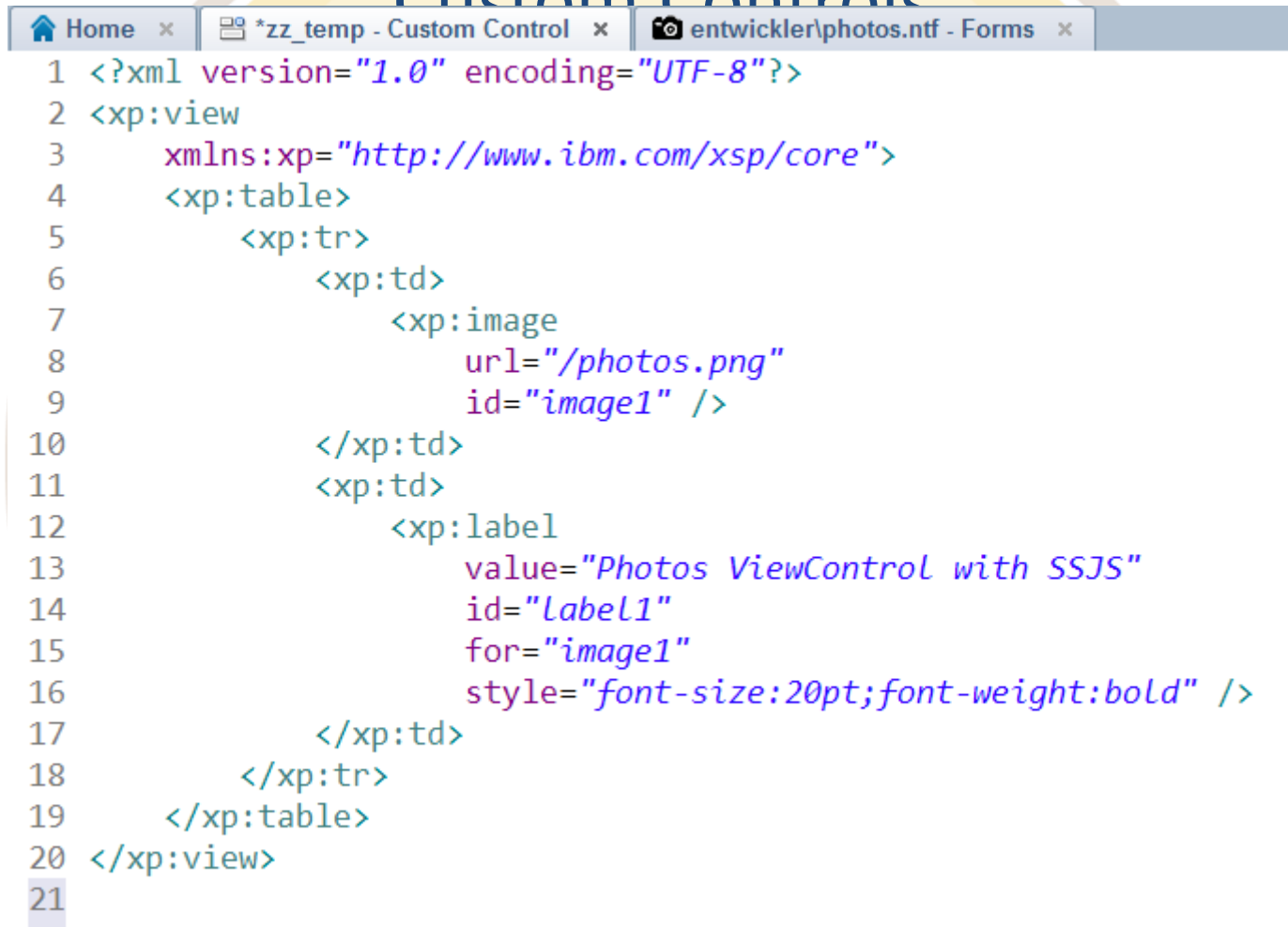
# Custom Controls



# Custom Controls

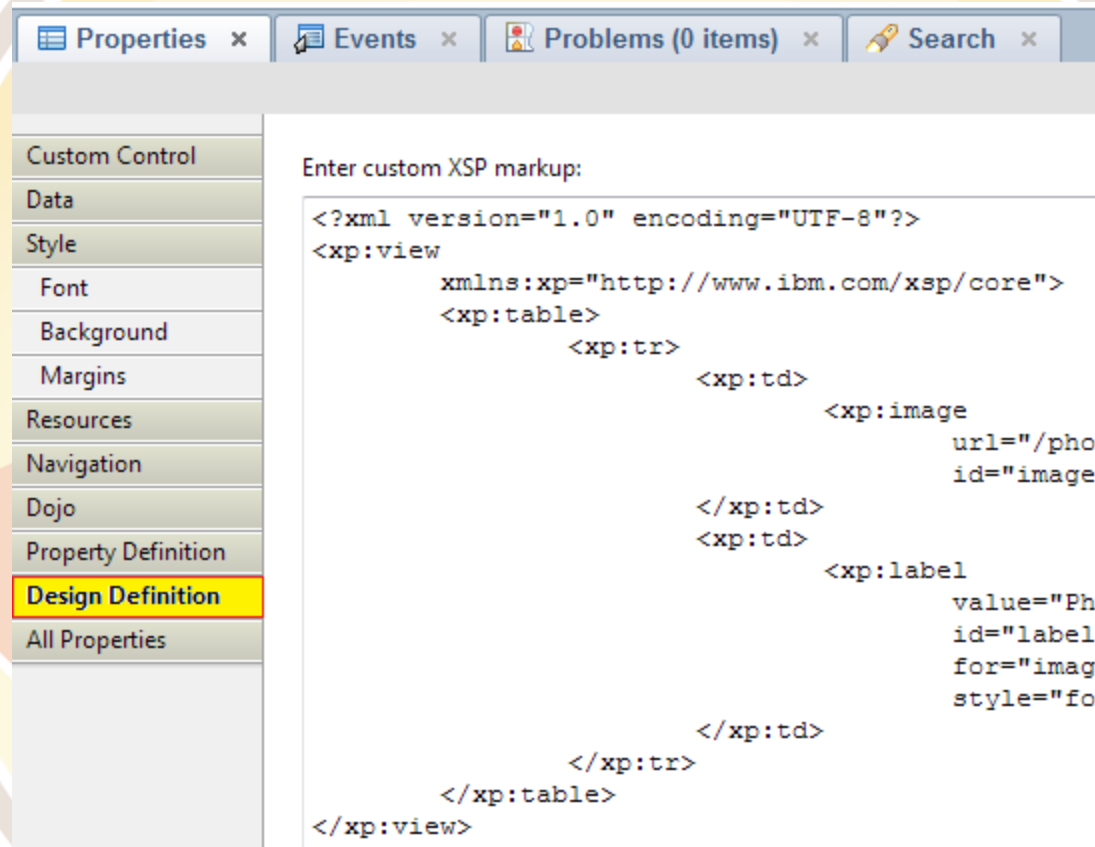


## Custom Controls



```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <xp:view
3     xmlns:xp="http://www.ibm.com/xsp/core">
4     <xp:table>
5         <xp:tr>
6             <xp:td>
7                 <xp:image
8                     url="/photos.png"
9                     id="image1" />
10            </xp:td>
11            <xp:td>
12                <xp:label
13                    value="Photos ViewControl with SSJS"
14                    id="label1"
15                    for="image1"
16                    style="font-size:20pt;font-weight:bold" />
17            </xp:td>
18        </xp:tr>
19    </xp:table>
20 </xp:view>
21
```

# Custom Controls





# Custom Controls

The screenshot shows a Lotus Notes XPage interface with a browser window titled 'photos - XPage'. The page has a navigation bar with four buttons: 'Add Photo', 'Photo Gallery', 'Photo View Control', and 'All Photo Controls Demo'. Below the navigation bar is a section titled 'Photos'. Inside this section, there is a custom control with a title '{title1}' and a set of navigation links 'Previous 1|2|3|4|5 Next'. Below the links is a table with two columns: 'Subject' and 'Description'. The 'Subject' column contains a single entry 'Subject', and the 'Description' column is empty.

Subject	Description
Subject	

# Custom Controls



# Custom Controls



# JAVA OBJECTS



# Implementing Java in an XPages Application

```
8 public class Image implements Serializable, Comparable<Image> {
9
10     private static final long serialVersionUID = 1L;
11
12     private String _subject;
13     private String _description;
14     private String _universalID;
15
16     /**
17      * Zero-Argument Constructor
18      */
19     public Image() {
20     }
21
22     public Image(Document document) {
23         this.load(document);
24     }
25
26     public void load(Document document) {
27         try {
28             this._subject = document.getItemValueString("subject");
29             this._description = document.getItemValueString("description");
30             this._universalID = document.getUniversalID();
31         } catch (NotesException e) {
32             System.out.println("*****");
33             System.out.println("EXCEPTION in Image.load(document) method");
34             e.printStackTrace();
35         }
36     }
37 }
```

# Implementing Java in an XPages Application

- In XPages, everything you write will be turned into Java
- EXCEPT the Java you write.
- Writing your own Java avoids the middle stuff, and gives you explicit control over what your stuff does.

# Implementing Java in an XPages Application

# DEMONSTRATION

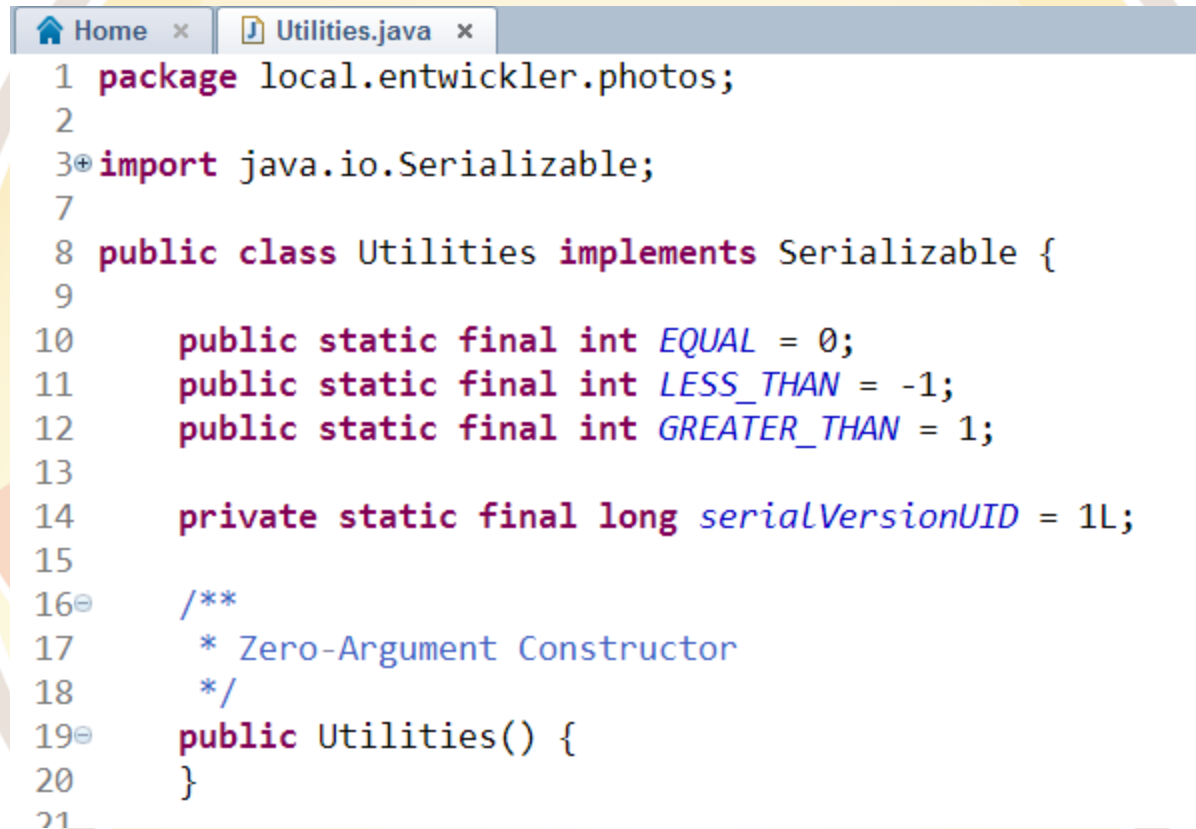
```
8 public class Image implements Serializable, Comparable<Image> {
9
10     private static final long serialVersionUID = 1L;
11
12     private String _subject;
13     private String _description;
14     private String _universalID;
15
16     /**
17      * Zero-Argument Constructor
18      */
19     public Image()
20     {
21     }
22
23     public Image(Document document) {
24         this.load(document);
25     }
26
27     public void load(Document document) {
28         try {
29             this._subject = document.getItemValueString("subject");
30             this._description = document.getItemValueString("description");
31             this._universalID = document.getUniversalID();
32         } catch (NotesException e) {
33             System.out.println("*****");
34             System.out.println("EXCEPTION in Image.load(document) method");
35             e.printStackTrace();
36         }
37     }
38 }
```



# Implementing Java in an XPages Application

## Utilities Class

# Implementing Java – the Utilities class

A screenshot of a code editor window. The title bar shows two tabs: 'Home' and 'Utilities.java'. The code is written in Java and is as follows:

```
1 package local.entwickler.photos;
2
3 import java.io.Serializable;
4
5
6
7
8 public class Utilities implements Serializable {
9
10     public static final int EQUAL = 0;
11     public static final int LESS_THAN = -1;
12     public static final int GREATER_THAN = 1;
13
14     private static final long serialVersionUID = 1L;
15
16     /**
17      * Zero-Argument Constructor
18      */
19     public Utilities() {
20     }
21 }
```

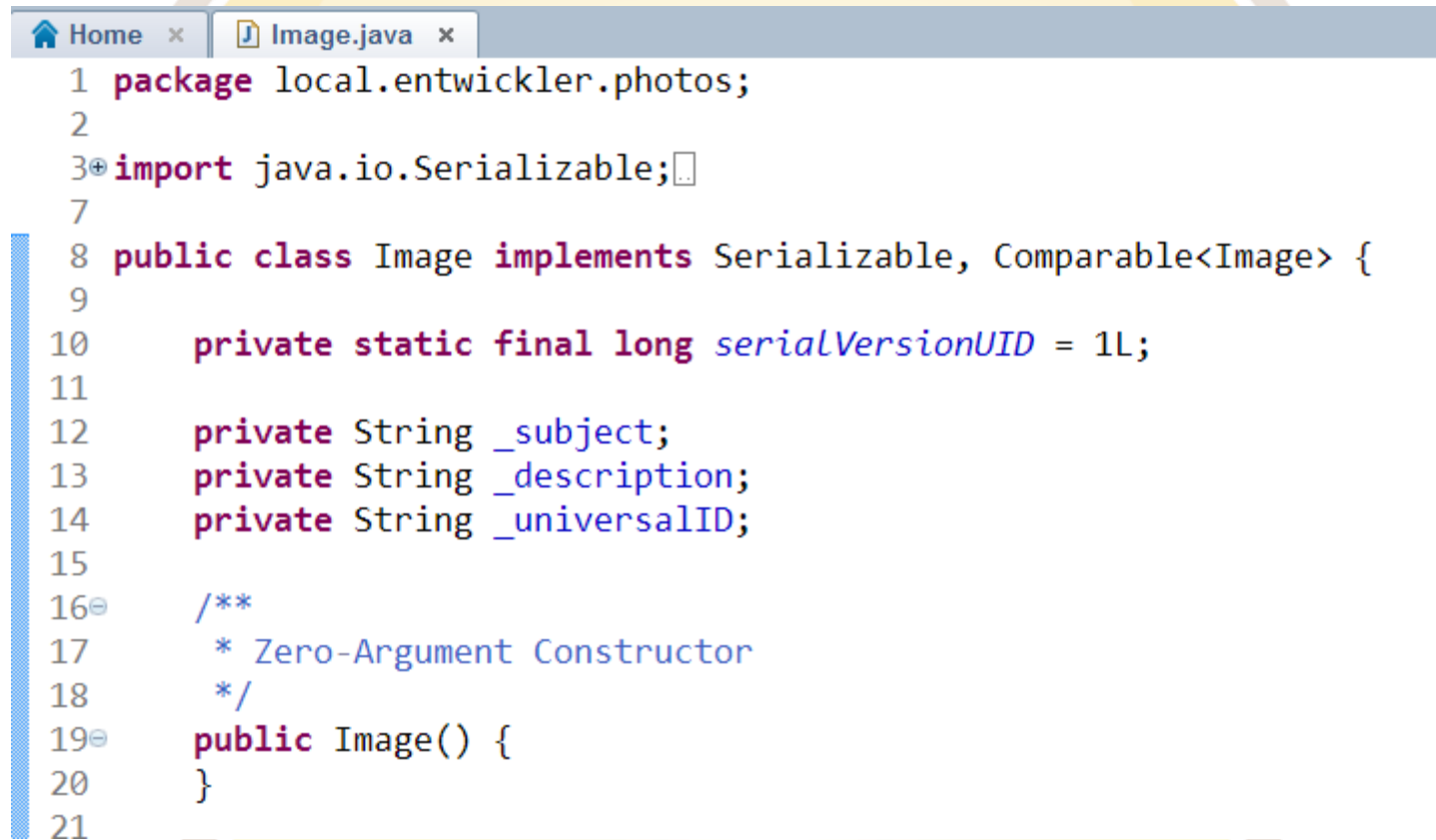
## Implementing Java – the Utilities class

```
21
22  /**
23   * Tim Tripcony's awsome simple incinerate method handles recycle issues
24   * for Domino Objects.
25   *
26   * @param dominoObjects
27   *         Domino Objects to recycle.
28   */
29  public static void incinerate(Object... dominoObjects) {
30      for (Object dominoObject : dominoObjects) {
31          if (null != dominoObject) {
32              if (dominoObject instanceof Base) {
33                  try {
34                      ((Base) dominoObject).recycle();
35                  } catch (NotesException recycleSucks) {
36                      // optionally log exception
37                  }
38              }
39          }
40      }
41  }
42 }
43
```

# Implementing Java in an XPages Application

## Image Class

# Implementing Java – the Image class



The screenshot shows a code editor window with two tabs: 'Home' and 'Image.java'. The 'Image.java' tab is active, displaying the following Java code:

```
1 package local.entwickler.photos;
2
3 import java.io.Serializable;
4
5
6
7
8 public class Image implements Serializable, Comparable<Image> {
9
10     private static final long serialVersionUID = 1L;
11
12     private String _subject;
13     private String _description;
14     private String _universalID;
15
16     /**
17      * Zero-Argument Constructor
18      */
19     public Image() {
20     }
21 }
```

## Implementing Java – the Image class

```
21
22 public Image(Document document) {
23     this.load(document);
24 }
25
26 public void load(Document document) {
27
28     try {
29         this._subject = document.getItemValueString("subject");
30         this._description = document.getItemValueString("description");
31         this._universalID = document.getUniversalID();
32     } catch (NotesException e) {
33         System.out.println("*****");
34         System.out.println("EXCEPTION in Image.load(document) method");
35         e.printStackTrace();
36     }
37
38     System.out.println("Image.load(document)");
39     System.out.println("\t Subject: " + this.getSubject());
40     System.out.println("\t Description: " + this.getSubject());
41     System.out.println("\t UniversalID: " + this.getUniversalID());
42
43 }
```

## Implementing Java – the Image class

```
44
45 public String getSubject() {
46     return this._subject;
47 }
48
49 public String getDescription() {
50     return this._description;
51 }
52
53 public String getUniversalID() {
54     return this._universalID;
55 }
56
```



## Implementing Java – the Image class

```
50
57 public String getUrl() {
58     String delimiter = "//";
59     StringBuilder sb = new StringBuilder();
60     sb.append("..");
61     sb.append(delimiter);
62     sb.append("attachments.nsf");
63     sb.append(delimiter);
64     sb.append("0");
65     sb.append(delimiter);
66     sb.append(this.getUniversalID());
67     sb.append(delimiter);
68     sb.append("$FILE");
69     sb.append(delimiter);
70     sb.append("image.jpg");
71
72     return sb.toString();
73 }
74
```

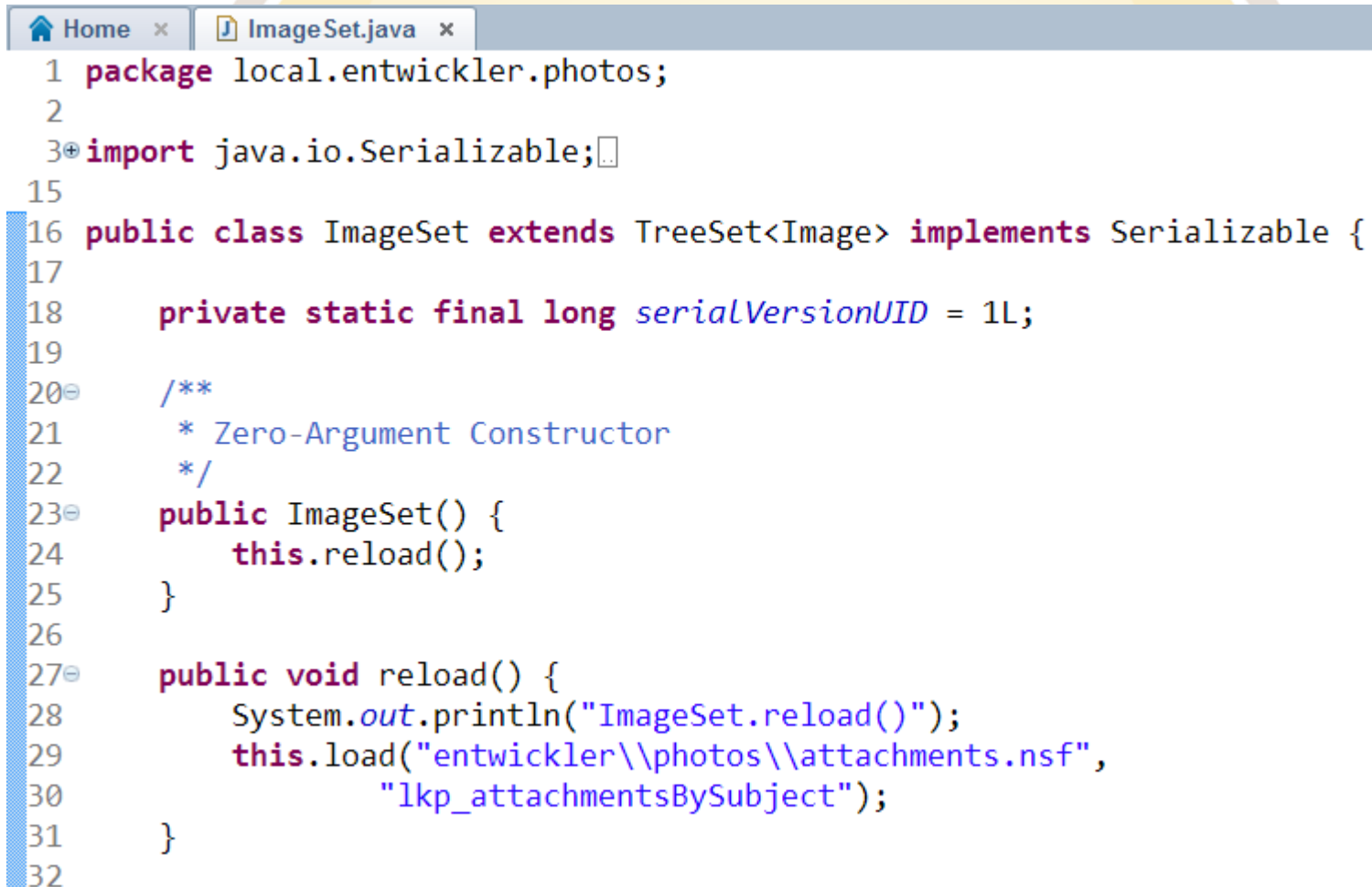
# Implementing Java – the Image class

```
/*
 * *****
 * *****
 *
 * Comparable implementation methods
 *
 * *****
 * *****
 */
// .. .. ..
/*
 * *****
 * *****
 *
 * hashCode and equals
 *
 * *****
 * *****
 */
```

## Implementing Java in an XPages Application

# ImageSet Class

# Implementing Java – the ImageSet class



```
1 package local.entwickler.photos;
2
3 import java.io.Serializable;
15
16 public class ImageSet extends TreeSet<Image> implements Serializable {
17
18     private static final long serialVersionUID = 1L;
19
20     /**
21      * Zero-Argument Constructor
22      */
23     public ImageSet() {
24         this.reload();
25     }
26
27     public void reload() {
28         System.out.println("ImageSet.reload()");
29         this.load("entwickler\\photos\\attachments.nsf",
30                 "lkp_attachmentsBySubject");
31     }
32
```

# Implementing Java – the ImageSet class

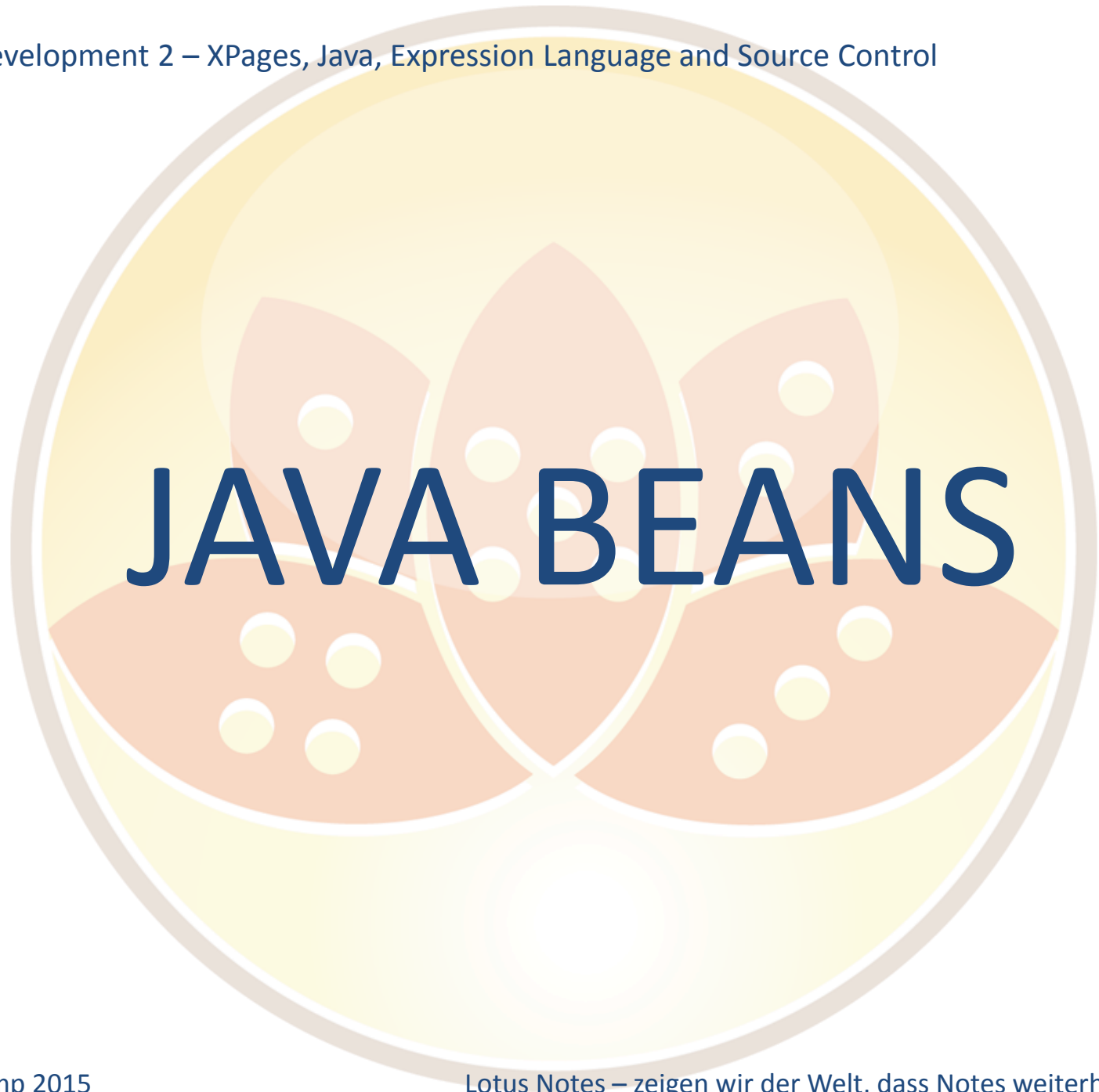
```
33 public void load(String filepath, String viewname) {  
34     System.out.println("ImageSet.load(filepath, viewname)");  
35     System.out.println("filepath: " + filepath);  
36     System.out.println("viewname: " + viewname);  
37  
38     Database database = null;  
39     View view = null;  
40     ViewNavigator navigator = null;  
41     ViewEntry viewentry = null;  
42     ViewEntry stupidRecycleHack = null;  
43     Document document = null;  
44 }
```

# Implementing Java – the ImageSet class

```
15 Session session = DominoUtils.getCurrentSession();
16 try {
17     |
18     // code goes here
19
20 } catch (NotesException e) {
21     System.out.println("*****");
22     System.out.println("EXCEPTION in ImageSet.load(filepath, viewname");
23     e.printStackTrace();
24 } finally {
25     Utilities.incinerate(document, stupidRecycleHack, viewentry,
26         navigator, view, database);
27 }
28 }
```

# Implementing Java – the ImageSet class

```
try {  
    database = session.getDatabase(session.getServerName(), filepath);  
    view = database.getView(viewname);  
    view.setAutoUpdate(false);  
    navigator = view.createViewNav();  
    viewentry = navigator.getFirstDocument();  
    while (null != viewentry) {  
        document = viewentry.getDocument();  
  
        Image image = new Image(document);  
        this.add(image);  
        Utilities.incinerate(document);  
  
        stupidRecycleHack = navigator.getNextDocument();  
        Utilities.incinerate(viewentry);  
        viewentry = stupidRecycleHack;  
    }  
}
```





## Java Beans

**DEMONSTRATION**

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <faces-config>
3   <managed-bean>
4     <managed-bean-name>Images</managed-bean-name>
5     <managed-bean-class>local.entwixler.photos.ImageSet</managed-bean-class>
6     <managed-bean-scope>view</managed-bean-scope>
7   </managed-bean>
8   <!--AUTOGEN-START-BUILDER: Automatically generated by IBM Domino Designer. Do not modify.-->
9   <!--AUTOGEN-END-BUILDER: End of automatically generated section-->
10 </faces-config>
```

# Java Beans



```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <faces-config>
3   <managed-bean>
4     <managed-bean-name>Images</managed-bean-name>
5     <managed-bean-class>local.entwickler.photos.ImageSet </managed-bean-class>
6     <managed-bean-scope>session</managed-bean-scope>
7   </managed-bean>
8   <!--AUTOGEN-START-BUILDER: Automatically generated by IBM Domino Designer. Do not modify.-->
9   <!--AUTOGEN-END-BUILDER: End of automatically generated section-->
10 </faces-config>
11 |
```

The Lotus Notes logo is a large, stylized lotus flower in shades of yellow and orange, centered on the slide. The text "NATIVE CONTROLS" is overlaid on the logo in a bold, dark blue, sans-serif font.

# NATIVE CONTROLS

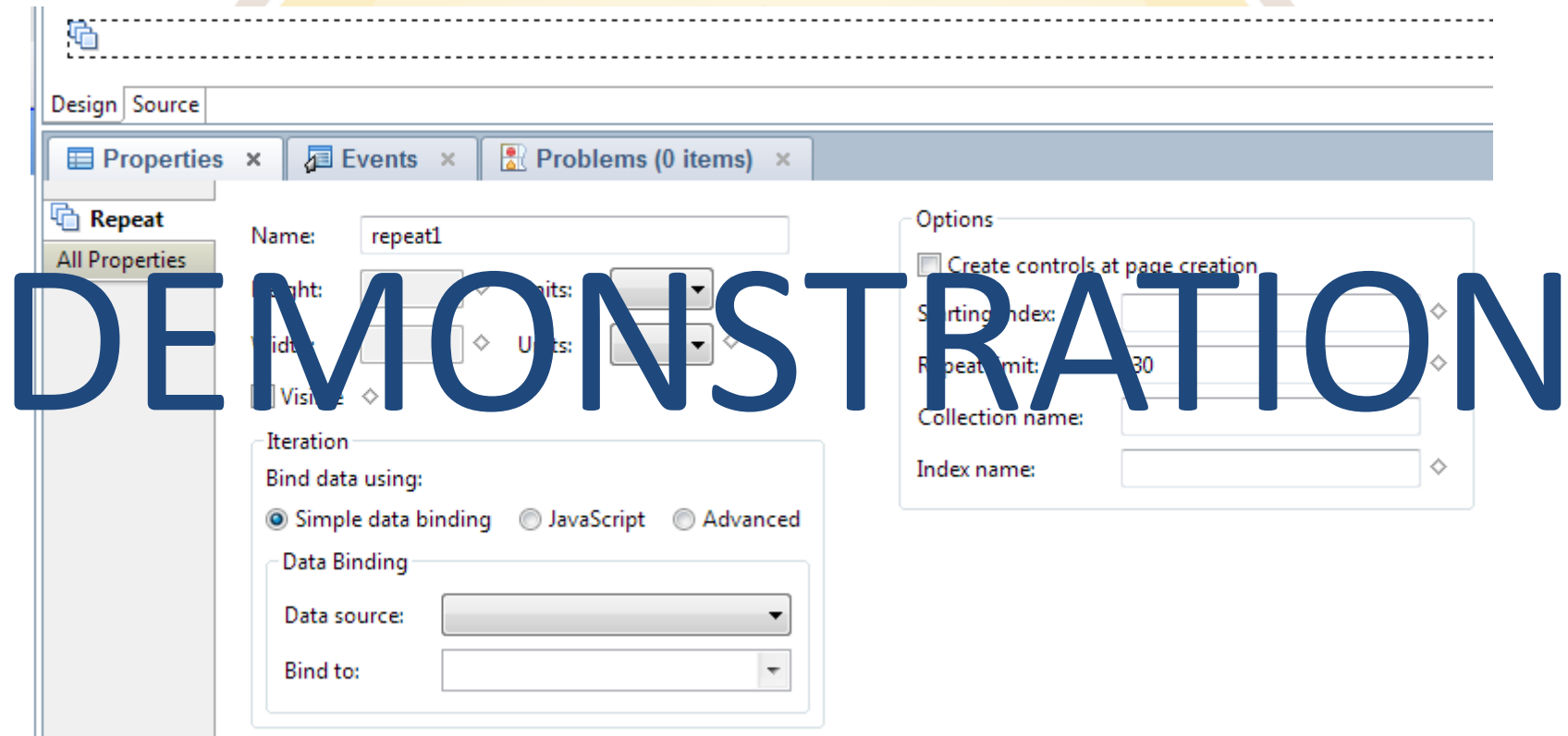


**MOST  
IMPORTANT  
CONTROL YOU  
WILL EVER USE**



MOST  
IMPORTANT  
**REPEAT CONTROL**  
YOU  
WILL EVER USE

# Repeat Control



# Repeat Control

The screenshot displays the IBM Notes client interface for configuring a Repeat Control. The top bar shows the 'Home' icon and the document title 'photos\_repeatcontrol\_bindJava\_list ...'. The main design area shows a dashed box containing a 'title1' label and a 'computedField1 {computedField2}' label, with a small image icon in the center. Below the design area, the 'Design' tab is selected, and the 'Properties' pane is open. The 'Repeat' control is selected, and its properties are displayed. The 'Name' is 'repeat1'. The 'Height' and 'Width' are set to default values. The 'Visible' checkbox is checked. The 'Iteration' section shows 'Bind data using' set to 'Simple data binding'. The 'Data Binding' section shows 'Data source' set to 'Images' and 'Bind to' set to an empty field. The 'Options' section shows 'Create controls at page creation' unchecked, 'Starting index' set to 0, 'Repeat limit' set to 30, 'Collection name' set to 'rowData', and 'Index name' set to 'rowIndex'.

Home x photos\_repeatcontrol\_bindJava\_list ... x

{title1}

{computedField1} {computedField2}

Design Source

Properties x Events x Problems (0 items) x Search x

Repeat

All Properties

Name: repeat1

Height: Units:

Width: Units:

☒ Visible

Iteration

Bind data using:

☒ Simple data binding ☐ JavaScript ☐ Advanced

Data Binding

Data source: Images

Bind to:

Options

☐ Create controls at page creation

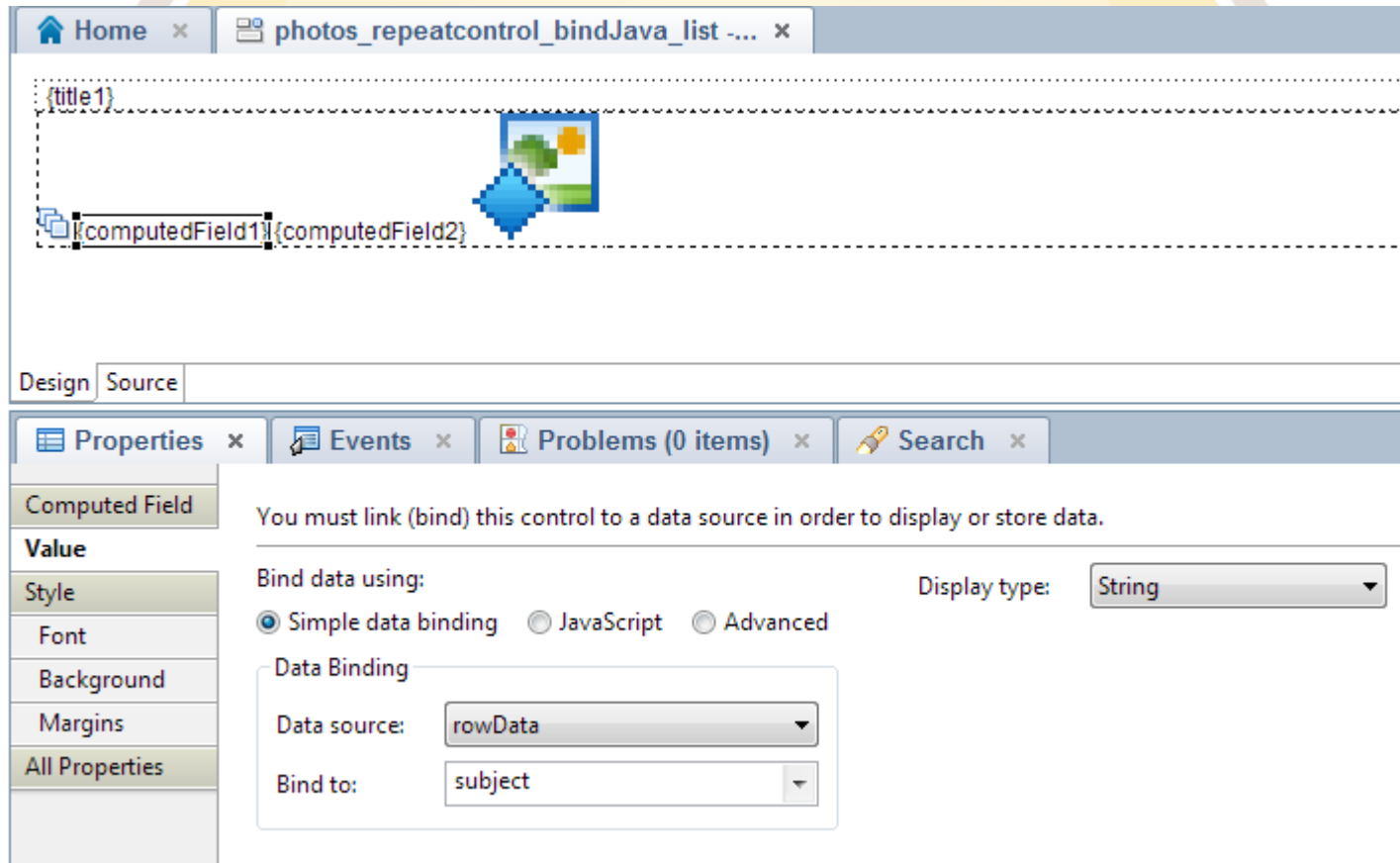
Starting index:

Repeat limit: 30

Collection name: rowData

Index name: rowIndex

# Repeat Control





A large, stylized Lotus logo is centered on the slide. It features a circular design with a light yellow background and a brown outer ring. Inside the ring, there are several overlapping, rounded shapes in shades of orange and yellow, creating a lotus-like pattern. The text "EXPRESSION LANGUAGE" is overlaid on this design in a bold, dark blue, sans-serif font.

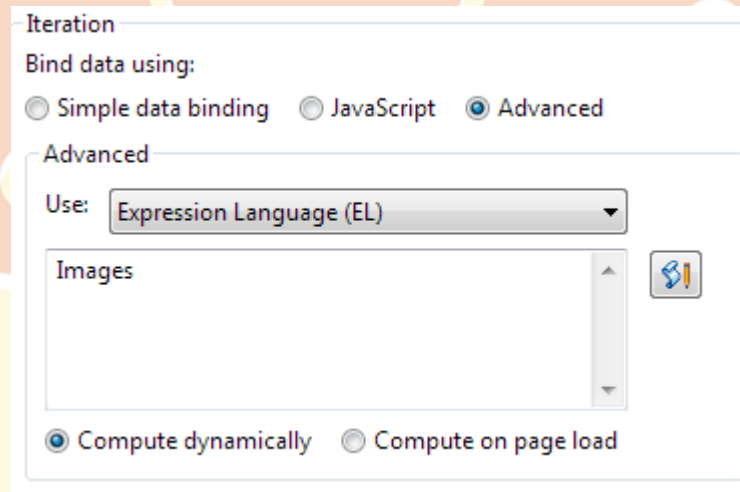
# EXPRESSION LANGUAGE

## The Power of EL (Expression Language)

- Java Server Pages scripting language
- Syntactically and Semantically Similar to JavaScript
- Both Deferred and Immediate Evaluation
- Read from JavaBeans, Data Structures, Implicit Objects
- Write data, such as user input, to JavaBeans Objects
- Invoke public methods, both static and implicit
- Dynamically perform arithmetic and logic operations
- Bridge between XML markup and Server Java Objects

## The Power of EL (Expression Language)

# DEMONSTRATION



Iteration

Bind data using:

☐ Simple data binding ☐ JavaScript ☒ Advanced

Advanced

Use: Expression Language (EL)

Images

☒ Compute dynamically ☐ Compute on page load

# The Power of EL (Expression Language)

```
<xp:tr
  id="tr1">
  <xp:this.styleClass><![CDATA[#{javascript:((rowIndex % 2) == 0)? "evenrow" : "oddrow";}]]></xp:this.styleClass>
  <xp:td>
    <xp:text
      escape="true"
      id="computedField1">
      <xp:this.value><![CDATA[#{javascript:rowData.getColumnValue("Subject")}]]></xp:this.value>
    </xp:text>
  </xp:td>
  <xp:td>
    <xp:text
      escape="true"
      id="computedField2">
      <xp:this.value><![CDATA[#{javascript:rowData.getColumnValue("Description")}]]></xp:this.value>
    </xp:text>
  </xp:td>
  <xp:td>
    <xp:image
      id="image1"
      style="height:64px">
      <xp:this.url><![CDATA[#{javascript:"../attachments.nsf/0/" + rowData.getUniversalID() + "/$FILE/image.jpg"}]]></xp:this.url>
    </xp:image>
  </xp:td>
  <xp:eventHandler
    event="onClick">
```

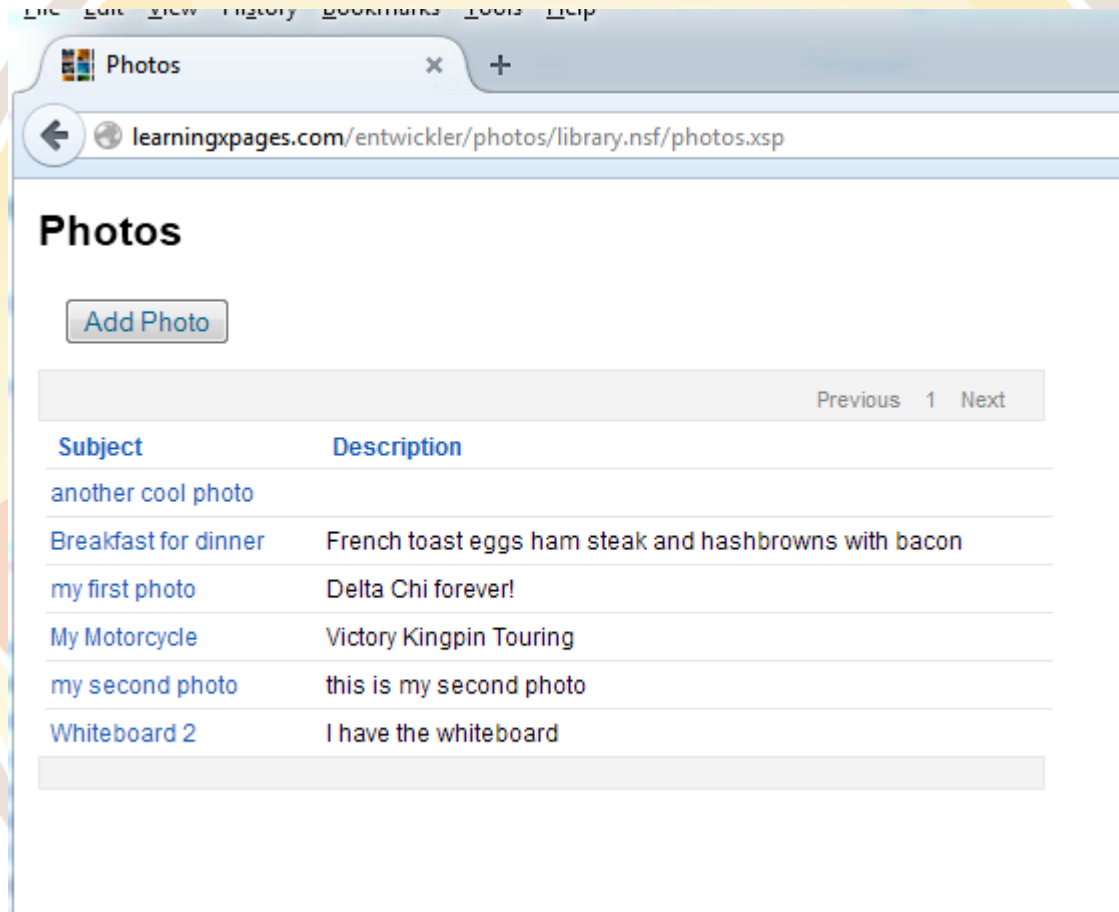
# The Power of EL (Expression Language)

```
<xp:tr
  id="tr1"
  styleClass="#{((rowIndex mod 2) eq 0)? 'evenrow' : 'oddrow'}">
  <xp:eventHandler
    event="onclick"
    submit="true"
    refreshMode="complete">
    <xp:this.action>
      <xp:openPage
        name="/photo.xsp"
        target="openDocument"
        documentId="#{rowData.universalID}" />
    </xp:this.action>
  </xp:eventHandler>
  <xp:td>
    <xp:text
      escape="true"
      id="computedField1"
      value="#{rowData.subject}" />
  </xp:td>
  <xp:td>
    <xp:text
      escape="true"
      id="computedField2"
      value="#{rowData.description}" />
  </xp:td>
```

Putting it All Together

# DEMONSTRATION

# Putting it All Together





# QUESTIONS?



## Additional Resources

- LearningXPages <http://learningXPages.com>
- XPages.info
- OpenNTF.org
- XPages Development Wiki (URL is too long, Google it)
- Coding Horror (SE Founder Jeff Atwood's blog)
- Joel on Software (SE Founder Joel Spolsky's blog)
- Mastering XPages – IBM Press
- XPages Extension Library – IBM Press
- The Rabbit Hole <http://nathantfreeman.wordpress.com>
- NotesIn9 <http://notesin9.com>

# THANK YOU FOR ATTENDING

My Contact Info:

**Devin S. Olson**

[devin.olson@azlighthouse.com](mailto:devin.olson@azlighthouse.com)

+1 616-295-1683

Skype: spanky762

Twitter: spanky762

Facebook: [Facebook.com/default.xsp](https://www.facebook.com/default.xsp)