BP212 Deep Dive into IBM XPage Expression Language Syntax

Colin MacDonald | Senior Engineer | GBS
Karsten Lehmann | CEO | Mindoo GmbH
Agenda

- Setting the context, revise familiar old LotusScript
- Moving to the less familiar, are there similarities with XPages?
- Brief History – DNA of EL and XPages
- A closer look at Value Properties and EL
- EL Syntax
- Managed Beans
Agenda

- How do we use managed beans in XPages?
- Which interface do you serve?
- Demo
- Wrap-up
- Q & A
- Complimentary Sessions
Setting the context - Familiar

- Flashback to a pre-XPage world
- Configurable, highly dynamic pages were difficult
  - Feed the data as JSON/XML to the browser and use JavaScript to write the page.
  - Run a LotusScript agent to print HTML
- LotusScript Evaluate
  - Gave you the ability to take a STRING and evaluate code
  - Could be user driven data
- Use Cases
  - Content Management
  - Web based workflow
Show some LotusScript code
Setting the context - Unfamiliar

- We're assuming you know
  - XPages is a UI with underlying XML
  - XML → Java → Java Byte Code
  - You place UI Components and bind them to a data source

- XPages are really just a Java agent.
  - Emits HTML markup by printing to a stream
  - Executes code in events
Show XPage Java code
Setting the context – Unfamiliar continued

- We're assuming you know
  - XPages is a UI with underlying XML
  - XML → Java → Java Byte Code
  - You place UI Components and bind them to a data source

- XPages are really just a Java agent.
  - Emits HTML markup by printing to a stream
  - Executes code in events

- Evaluate == EL
  - @Formula
  - Addition of Server Side JavaScript
  - Includes base EL Syntax*
Introduction to Value Property and EL

- You should be familiar with the source tab of an XPage
  - Key to understanding XPage's JSF & JSP roots
  - It's all just XML → Strings

- You should be familiar with these already
  - Simple Data Bindings
  - Javascript

- You'd think we'd only be interested in EL in the Advanced Tab
  - Expression Language (EL) ← Is this the Jackpot?
  - Scoped Variable
  - Component Parameter
  - Custom
Introduction to Value Binding and EL

• Lets look at the XML Source for:
  ─ A Simple data binding
  ─ A Scoped Variable
  ─ Expression Language (EL)

• Why is it all the same?
  ─ Could it be because it's all EL under the hood?
Lets take a look
So it is really just a STRING!
Brief History – DNA of EL and XPages
(explaining what and why)

- As the 20\textsuperscript{th} Century drew to a close we had
  - Java Server Pages (JSP)
  - Standard Tag Library (STL or JSTL)
  - STL had Simplest Possible Expression Language (SPEL)

- By 2004 – 2005
  - JSP 2.0 with enhanced SPEL in JSR Review
  - Java Server Faces (JSF) was introduced and needed to extend SPEL
  - JSP and JSF aligned on a Unified Expression Language

- XPages is JSF under the hood
  - JSF 1.2 with some JSF 2.0 features
  - Plus all that Domino stuff
Let's look at the JSF EL Syntax

- `#{expr}` or `${expr}`
  - # indicates dynamic
  - $ static after construction (on page load). i.e. more efficient
  - expr is a combination of literals, identifiers and operators

- Literals (derived from Java)
  - Boolean: true, false
  - Integer: As in Java
  - Floating point: As in Java
  - String: 'xyz' or “xyz”,
    - includes escaped characters \"\\
  - null

- Operators
  - Arithmetic: +, -, *, /, div, mod
  - Logical: and, &&, or, ||, not, !
  - Relational: ==, eq, !=, ne, <, lt, >, gt, <=, le, >=, ge
  - Conditional: A ? B : C
Let's look at some more of the JSF EL Syntax

- Not to be forgotten
  - Null check: empty (use as a unary operator to get a boolean return)
  - Type check: instanceof

- Identifiers
  - Implicit Objects available in JSF Framework
    - FacesContext
    - Param
    - Cookie
    - etc.
Lets take a look
(and see if that works in XPages)
Lets look at some more of the JSF EL Syntax

- Identifiers
  - Implicit Objects in JSF Framework (IBM)
  - Managed Beans we inject via faces-config.xml

- Properties of Identifiers
  - You can access properties of the bean using “.” and “[ ]” notation
    - myBean.name → myBean.getName();
    - myBean[“Phil”] → myBean.get(“Phil”);
    - myBean.addresses[“home”].street → myBean.getAddresses().get(“home”).getStreet();

- So what are these beans?
Managed Beans

- What is a Bean
  - It's a Plain Old Java Object (POJO)
  - It's created and managed by the JSF Servlet
  - Can contain whatever you want
  - ANYTHING YOU CAN CODE!

- JSF uses Faces-Config.xml to control Managed Beans
  - It's an XML document that defines what Managed Beans are available
  - Controls the lifespan of the bean (application, session, request, view)

```xml
<?xml version="1.0" encoding="UTF-8"?>
<faces-config>
  <managed-bean>
    <managed-bean-name>JumpingBean</managed-bean-name>
    <managed-bean-class>com.sample.bean.JumpingBean</managed-bean-class>
    <managed-bean-scope>application</managed-bean-scope>
  </managed-bean>
</faces-config>
```
How do we do this in XPages?

- If you juggle you Eclipse Views, you get access to faces-config.xml
  - Add the Java Package Explorer view to your DDE Perspective
  - Look for WebContent/WEB-INF folder

- You have to add your Java files to a source folder inside the NSF
  - Add that source folder to the build path
  - Make sure you compile/build

- You should now have managed beans available in your XPages EL
Let's take a look
(at that working in XPages)
What interface do you serve?

- Java interfaces are an extremely powerful feature
  - No implementation, but defines a **contract**.
  - If your bean implements an interface:
    - It can be used by any code that understands that interface
    - The calling code only cares about the **contract**
    - Can implement multiple interfaces

- Why is this important?
  - The EL resolver is just a bit of Java code
  - It looks for beans that either conform to Java Get/Set Value Semantics
    - That's standard Java syntax and is found using reflection
  - OR, conforms to a supported interface

- So what interfaces are supported?
  - Java Map
  - XPages DataObject Interface
    - That's com.ibm.xsp.model.DataObject
What interface do you serve?

• So really, why is this important?
  – Dynamic Data API
  – Extensible & flexible
  – You really don't want to pre-populated a Map of all customers to do this
    – CustomerDB["Phil"]
What interface do you serve?

```java
package com.acme.demo.persondata;
import com.ibm.xsp.model.DataObject;

public class PersonData implements DataObject {

  public Class<?> getType(Object id) {
      // Return the type of class that id resolves to. Complex case could return Employee or Customer
      return Person.class;
  }

  public Object getValue(Object id) {
      // Retrieve a record from some store, based on id
      return null;
  }

  public boolean isReadOnly(Object id) {
      // You are free to implement your own, or rely on your underlying data store
      return false;
  }

  public void setValue(Object id, Object value) {
      // Store value in your data store using id
  }
}
```
Use Cases in the Demo

• Purchase Request
  – We will take a really classic Notes Application pattern
    - Table entry of line items
    - No embedded view element
    - Fields like Descrip_1 ... Descrip_n & Qty_1 ... Qty_n
  – And show you how to implement an XPage version
    - No conversion of data necessary
    - Just add an XPage

• Resolution of client-side ID inside Repeat controls
  – Complex XPages produce identifiers like this:
    - view_1:somePanel_3:ID_4
    - Very Bad for client-side JavaScript
  – Fortunately EL is just string replacement, so we'll show you how.
Use Cases in the Demo

• Moving your application logic into back-end Java classes
  – Share code between multiple applications (NSF's)
    - Version control, Unit Tests
    - All the Java goodness
  – Separate UI development from back-end code development
    - Best Practice for large projects, skill-sets differ
    - Model View Controller pattern.
    - Logging can be integrated transparently
  – Opens door to other data stores and data providers

• Using Xpath in EL
  – Xpath is part of Domino today
  – We'll go over what you have to do to use it

• Can we extend EL beyond JavaScript?
Demo
Wrap-up

- EL is just String transformation.
  - Multi-layered like an onion
    - ${expr} at page load (construction)
    - #{expr} on page refresh (dynamic)

- Managed Beans open the door
  - Limited by your imagination

- Gone are the days of “Notes can't do that”!
Question and Answer Time
Complementary Sessions

- **AD102 Hacking IBM Lotus Designer (Gently)**
  - By Tim Tripcony and Maureen Leland
  - Take-away:
    - Minimum: A FacesConfig editor plugin.
    - Maximum: Techniques you can use to empower your XPage development.

- **AD114 There and Back Again: Strategies for Re-factoring Notes Applications to XPages**
  - By Nathan T. Freeman and Philippe Riande
  - Take-away: Turning classic into modern.
    - Armed with your improved EL knowledge and these strategies, you will be better able to answer the migration question.
Legal Disclaimer

© IBM Corporation 2011. All Rights Reserved.

The information contained in this publication is provided for informational purposes only. While efforts were made to verify the completeness and accuracy of the information contained in this publication, it is provided AS IS without warranty of any kind, express or implied. In addition, this information is based on IBM’s current product plans and strategy, which are subject to change by IBM without notice. IBM shall not be responsible for any damages arising out of the use of, or otherwise related to, this publication or any other materials. Nothing contained in this publication is intended to, nor shall have the effect of, creating any warranties or representations from IBM or its suppliers or licensors, or altering the terms and conditions of the applicable license agreement governing the use of IBM software.

References in this presentation to IBM products, programs, or services do not imply that they will be available in all countries in which IBM operates. Product release dates and/or capabilities referenced in this presentation may change at any time at IBM’s sole discretion based on market opportunities or other factors, and are not intended to be a commitment to future product or feature availability in any way. Nothing contained in these materials is intended to, nor shall have the effect of, stating or implying that any activities undertaken by you will result in any specific sales, revenue growth or other results.

IBM, the IBM logo, Lotus, Lotus Notes, Notes, Domino and Lotusphere are trademarks of International Business Machines Corporation in the United States, other countries, or both. Unyte is a trademark of WebDialogs, Inc., in the United States, other countries, or both.

Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

All references to ACME refer to a fictitious company and are used for illustration purposes only.