

AD507 – Leveraging the Power of Object Oriented Programming in LotusScript

Jens-B. Augustiny – LIGONET GmbH

Bill Buchan – HADSL

The logo for Lotusphere 2007 features the word "Lotusphere" in a bold, sans-serif font, with a registered trademark symbol. The letter "h" is partially enclosed by a black circle. To the right of the circle is the year "2007". The entire logo is set against a bright yellow sunburst background with radiating lines.The IBM logo, consisting of the letters "IBM" in a bold, sans-serif font, with a registered trademark symbol.

Agenda

- **Introductions**
- Why Object Orientated Programming
- Basic OO techniques
- An Object Orientated Project
- Encapsulating Classes
- Extending Classes
- Summary
- Questions and Answers

Introductions

- Who are we ?
 - ▶ Jens-B. Augustiny, LIGONET GmbH
 - 25 years IT experience, 12 with Lotus Notes
 - Dual PCLP v4, v5, v6, v7
 - Founder and CTO of a Business Partner – LIGONET – in Switzerland
 - ▶ Bill Buchan, HADSL
 - 23 years IT experience, 12 with Lotus Notes
 - Dual PCLP in v3, v4, v5, v6, v7
 - Founder and CEO of a Business Partner – HADSL – in the UK
- Who are you ?
 - ▶ A LotusScript developer
 - ▶ You wish to improve your code, write more efficiently, & write more robust code

Agenda

- Introductions
- **Why Object Orientated Programming**
- Basic OO techniques
- An Object Orientated Project
- Encapsulating Classes
- Extending Classes
- Summary
- Questions and Answers

Why Object Orientated Programming?

- Object Orientated Programming:
 - ▶ Allows you to design higher-level objects.
 - ▶ Allows far more code-reuse.
 - ▶ Allows you to write less code.
 - And therefore leads to easier maintenance.
- What does this presentation deliver ?
 - ▶ We aim to show a typical application, and how OO techniques help the design process.
 - ▶ We show basic and more complex design techniques during the development of the application.
 - ▶ And we show common questions and answers as the presentation unfolds.

Agenda

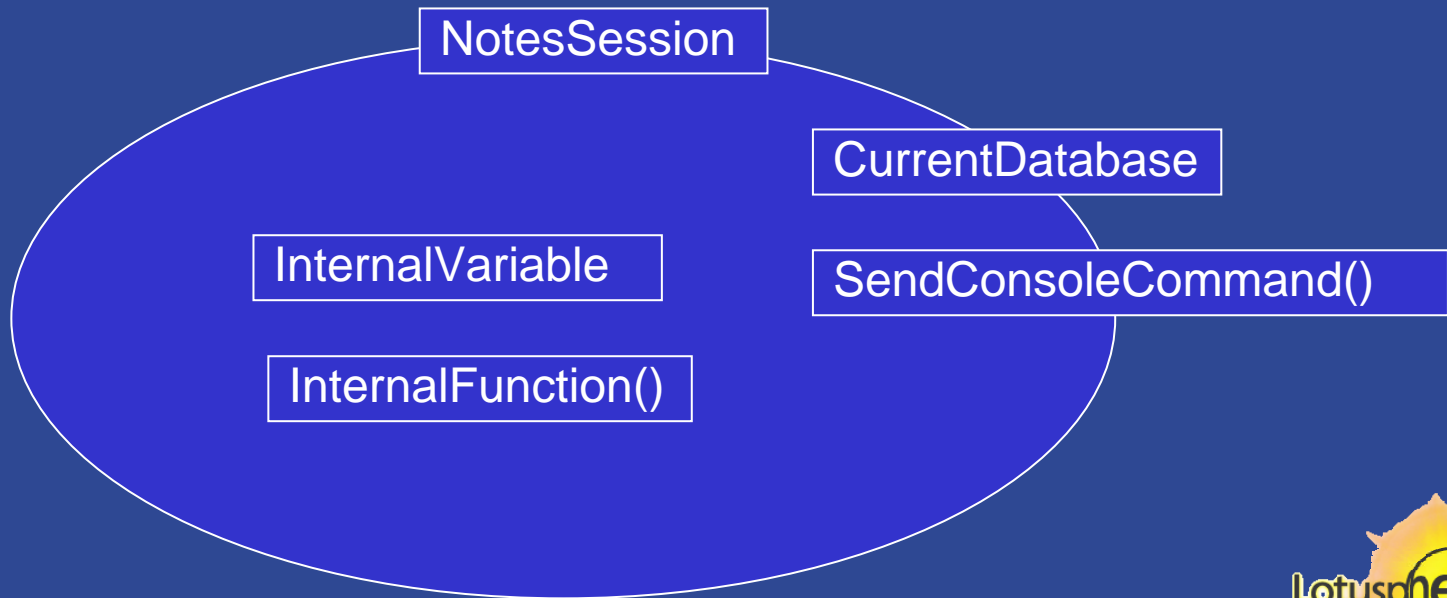
- Introductions
- Why Object Orientated Programming
- **Basic OO techniques**
- An Object Orientated Project
- Encapsulating Classes
- Extending Classes
- Summary
- Questions and Answers

Object Orientated Programming:

- Conceptually
 - ▶ An object is a collection of properties (data) and methods (code) which encapsulates a logical entity.
 - ▶ For instance, a Car is made up of objects –
 - 4 instances of "wheel", 1 instance of "engine", etc.
 - ▶ This allows us to break our problem set into logical units – objects.
- In LotusScript, we express this as a "class".
 - ▶ Our consumer code then creates new instances of classes – objects – that we then utilise.
 - ▶ Remember – NotesSession, NotesDocument – are all classes!

Public and Private:

- An object has to have
 - ▶ some public methods (functions/subs).
 - ▶ may have public members (variables).
- Internal code for the object need not be exposed
 - ▶ “Code Hiding” means not showing the consumer more than he needs to see.
 - ▶ Importantly – the consumer now codes to the public interface of that object.



How to design an Object Orientated Application

- Decide which objects make sense in terms of the problem space.
 - ▶ Make the object represent logical entities within the application.
 - ▶ “Person”, “PersonCollection”, “Transaction”, etc are good examples.
- Write these object names down on separate pieces of paper
- Perform an application “walk through”
 - ▶ This will quickly show which objects need to perform particular operations, and which objects interact with other objects.
 - ▶ This helps define the “public” interface to the Object.
 - ▶ Don’t worry as to how operations within objects are performed at this stage.
- Note:
 - ▶ This is a very simplistic approach, and far more approaches exist. Remember, OO programming is common in other environments.

OOP & LotusScript

- To create a class in LotusScript, wrap code and data around “class” definitions. Such as:

```
Class Person
  Private nnName as NotesName
  Public strInternetAddress as String
  Sub new(doc as NotesDocument)
    Set nnName = doc.GetItemValue("FullName")(0)
    me.strInternetAddress = doc.getItemValue("InternetAddress")(0)
  End sub
  Public function getName
    If (me.nnName is nothing) then exit function
    getName = nnName.Abbreviated
  End function
End class
```

- And consume it with:

```
Dim P as new Person(doc)
Print "Person: " + P.getName()
```

Why does that help ?

- It helps us construct complex (and possibly interconnected) in-memory data structures
- It sites the code for that data structure alongside the data structure
 - ▶ Making maintenance easier
- We can use these complex data structures to represent far more complex algorithms.
- We can architect solutions at a far higher level, using objects (instances of these classes).
- Classes can inherit from other classes, thus aiding code re-use(dramatically).

Where does this help us ?

- Complex code.
- Large applications.
- Applications that have or may change data structure frequently.
- Groups of applications that share common data structures.

Examples:

- Directory Comparison tools.
 - ▶ Its far faster to read multiple directories into memory and perform comparisons between in-memory objects.
- Workflow systems
 - ▶ Where you can “hide” the complexity of the workflow from other code.
- Interfaces to External Systems
 - ▶ Where the Interface can be complex.
 - ▶ Or where you wish to support multiple interfaces.
- Where you are attempting to perform complex relational operations.

Agenda

- Introductions
- Why Object Orientated Programming
- Basic OO techniques
- **An Object Orientated Project**
- Encapsulating Classes
- Extending Classes
- Summary
- Questions and Answers

Customer Specification

- Our Company – Cheese International – is going through a sensitive “media event” after the leader of a (large, dangerous and heavily armed) country almost choked to death on some “stinking bishop”.
- We have therefore been tasked with writing a Notes database which will read all on-line news sites looking for mention of our company.
- This database will then be replicated to all the people in our Marketing department, who can then respond.
- Today.
- This sounds hard, right ?

High Level Architecture

- This may be a little simpler than we thought.
- All our targeted news sites allow people to use RSS:
 - ▶ Really Simple Syndication.
 - ▶ It's a way of downloading a summary of their site by HTTP (the same protocol that a web browser uses), in XML format.
- So what we need to do is
 1. construct a database with a list of RSS URL's.
 2. scan these URL's.
 3. Open up the RSS feed.
 4. Download the XML.
 5. Parse the XML.
 6. Extract the news Items
 7. Write them to documents.
 8. Go down the pub.

Remember the golden rule:

We construct business applications from pre-existing components.

We do NOT construct new technology if at all possible.

Design Considerations:

- The only part that we cant do within LotusScript is the
 - ▶ Use Http to download XML
- We could parse XML within LotusScript from version 6 onwards.
- After extensive research (10 minutes on Google.com)
 - ▶ We find the MSXML v6 Software Development Kit.
 - ▶ It can download XML, parse it and uses the COM interface.
 - ▶ So we can use "CreateObject" within LotusScript to interface with it.
 - ▶ Its built into IE7 and Vista
 - But we don't have that.
 - ▶ But you can install it on the client (and/or server)
 - ▶ Its platform specific – to windows.
 - In this case, so are we.

Pitfalls

- You have to instantiate new Objects within COM, and make absolutely sure that these are deallocated.
 - ▶ Classes can help with that
 - Classes Support Constructors and Destructors
- We're keeping a complex relationship between Sites and their news stories
 - ▶ Classes can help there too!

Implementation Design

- Lets have a class for a site
- It can allocate all our COM stuff and make sure its shut down properly
- It can then keep track of all News Items for that site.
- It can write out new Notes Documents for new News items
 - ▶ Maintaining a unique reference for each story is actually easy, as each story will have its own unique URL.

First Attempt at our Site Class

- A first attempt at our "Site" class could be:

```
Class SiteClass
  Private strSiteName as String
  Private strURL as String
  private msXml as Variant
  sub new(doc as NotesDocument)
    ' get the site name and URL from the document
    ' Instantiate the COM objects
    ' Open up the Site and get the XML
    ' Parse the XML, and for each News Story,
    ' Do something...
  end sub
  sub delete()
    set msXML = nothing
  end sub
End class
```

First Attempt at our Site Class....

- And we'd use this class by:

```
Dim sSession as new NotesSession
Dim dbThis as NotesDatabase
Set dbThis = sSession.currentDatabase
Dim vLookup as NotesView
Set vLookup = dbThis.getView("Sites")
Dim doc as NotesDocument
Set doc = vLookup.getFirstDocument
While not doc is nothing
    dim S as new SiteClass(doc)
    set doc = vLookup.getNextDocument(doc)
wend
```


Results:

- We're clearly getting the XML back from the site
- We are not yet parsing the Stories out of the site XML information
- What to do with the stories themselves ?
 - ▶ It's a relational construct
 - ▶ One site can have zero or more stories
 - ▶ Classes to the rescue ?
 - ▶ But how ?

Agenda

- Introductions
- Why Object Orientated Programming
- Basic OO techniques
- An Object Orientated Project
- **Encapsulating Classes**
- Extending Classes
- Summary
- Questions and Answers

Encapsulating Classes

- So far we now understand how to use a single class by itself.
- How do we include - encapsulate - other classes within the first class ?
- Can we encapsulate a collection of other objects – instances of classes within this class ?

Lists

- Lists. A collection construct that's been in LotusScript since LotusScript was born.
- It allows you to construct a memory construct where you have zero or more items in a list, and each item is referenced by a lookup value. Like an in-memory "view".
- How many people know how to use lists ?
 - ▶ Resources:
 - The View – December 2006 article.
 - <http://www.billbuchan.com/web.nsf/htdocs/BBUN67JJCV.htm>
 - ▶ Can Lists work with Classes ?
 - Of course!

A List of Classes.

- In fact, we can have a list of classes. For instance:

```
Dim sSession as new NotesSession
Dim dbThis as NotesDatabase
Set dbThis = sSession.currentDatabase
Dim vLookup as NotesView
Set vLookup = dbThis.getView("Sites")
Dim doc as NotesDocument
Dim Sites list as SiteClass
Set doc = vLookup.getFirstDocument
While not doc is nothing
    dim S as new SiteClass(doc)
    Set Sites( S.getname() ) = S
    set doc = vLookup.getNextDocument(doc)
wend
```

Lets define our Item Class

- This class will hold each News Item:

```
Class ItemClass
  Private strTitle as String
  Private strLink as String
  private strDescription as String
  sub new(Story as Variant)
    ' populate this
    ' object from the variant
  end sub
End class
```

```
Class SiteClass
  ...
  Private Items List as ItemClass
  ...
End class
```

Lets see this in Action

- Second Application Demonstration

Agenda

- Introductions
- Why Object Orientated Programming
- Basic OO techniques
- An Object Orientated Project
- Encapsulating Classes
- **Extending Classes**
- Summary
- Questions and Answers

Extending Classes

- Classes can be extended from other classes, and inherit code+properties.
- This leads to code-reuse.
- Example:

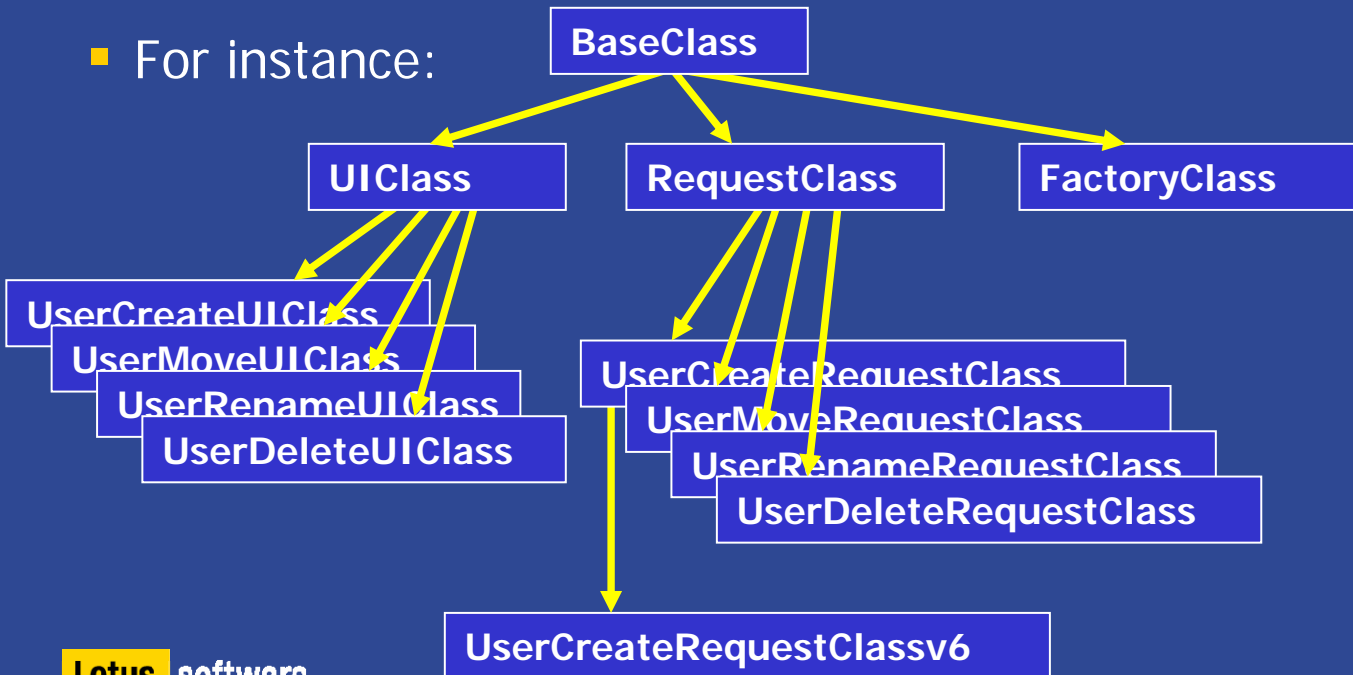
```
Class baseClass
...
Private Function log(msg as String)
...
end function
End class
```

```
Class SiteClass as BaseClass
...
sub new(doc as NotesDocument)
...
call me.log("Started")
...
end sub
End class
```

Extending Classes...

- When one class is extended from another class
 - ▶ It inherits all code and data from that original class
 - ▶ It can execute “private” methods and access private “members”
- There is no limit to the level of inheritance
 - ▶ But don't go mad

■ For instance:

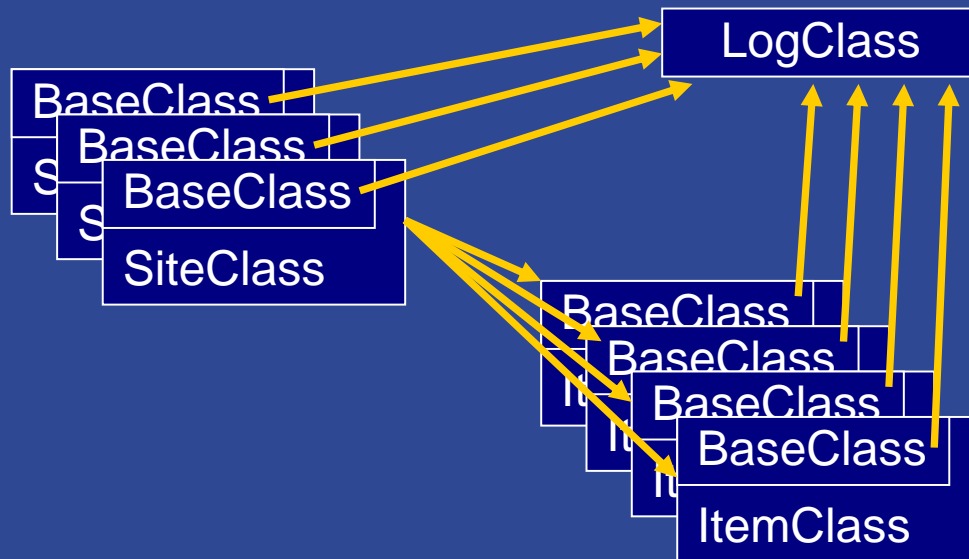


Production Quality

- How can we make this application more “production quality ? ”
 - ▶ Adding a logging subsystem.
 - ▶ But we don't want to make significant changes to the existing code
- We can do this by creating a “BaseClass”, and have the other classes inherit from that.
- Our BaseClass can incorporate a common logging subsystem
 - ▶ And any other code we wish to use.

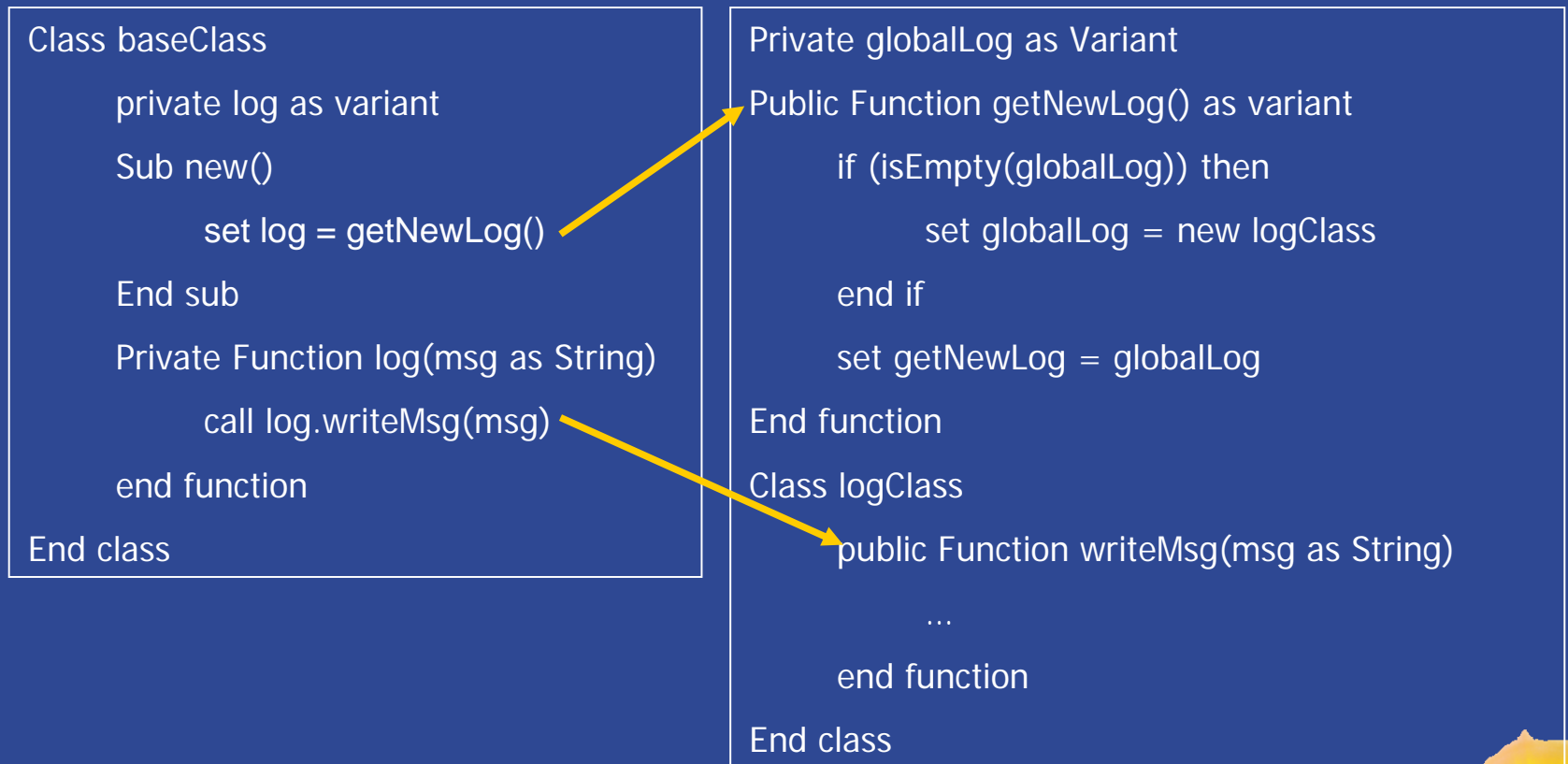
BaseClass Specification

- However, Since we shall have an object in memory for:
 - ▶ Each Site
 - ▶ Each story
- Our Baseclass cannot create a new NotesAgentLog class for each instance.
 - ▶ We need to only create ONE single instance of our NotesAgentLog class.



Singleton

- "Singleton" is a design pattern which ensures that ONLY one copy of an object exists at one time:



Lets incorporate BaseClass

- Third Application Demonstration

Agenda

- Introductions
- Why Object Orientated Programming
- Basic OO techniques
- An Object Orientated Project
- Encapsulating Classes
- Extending Classes
- **Summary**
- Questions and Answers

Summary

- Object Orientated Programming in LotusScript:
 - ▶ Is easy.
 - ▶ Allows far more code reuse.
 - ▶ Allows you to focus on the design.
- It's a stepping stone to other Object Orientated Languages
 - ▶ Java, anyone ?
 - ▶ Java as a language is NOT DIFFICULT
 - ▶ Its the Object Orientated Part that traditional LotusScript programmers find hard.

Some Resources

- Notes Designer help: look for "class statement", "custom classes", "derived classes" ... and more
- LotusScript Programmer's Guide, Developer Works
<http://www-10.lotus.com/ldd/notesua.nsf>
- Dynamic Loading
<http://www.hadsl.com/hadslblog.nsf/d6plinks/RCAS-6VDQV4>
- Article "Creating Custom Classes in LS" by Johan Känngård
<http://dev.kanngard.net/dev/home.nsf/ArticlesByTitle/LSCClass.html>
- Teamstudio ScriptBrowser:
<http://blogs.teamstudio.com>
- OpenDom Blog
<http://opendom.blogspot.com/> - Alain H Romedenne

Related Sessions at Lotusphere 2007

- BP301 Advanced Object Oriented Programming for LotusScript
Bill Buchan
- AD506 Intermediate LotusScript
John Dillon
- BP308 Leverage DXL and OOP to Build Powerful Tools
Mikkel Heisterberg
- AD504 What's New in the IBM Lotus Domino Objects?
James Cooper

Agenda

- Introductions
- Why Object Orientated Programming
- Basic OO techniques
- An Object Orientated Project
- Encapsulating Classes
- Extending Classes
- Summary
- **Questions and Answers**

Questions and Answers

- We are:
 - ▶ Jens-B Augustini – LIGONET GmbH:
 - <http://www.ligonet.ch> - Augustiny.j@ligonet.ch
 - ▶ Bill Buchan – HADSL
 - <http://www.hadsl.com> - bill.buchan@hads.com

- This was:
 - ▶ AD507 - Leveraging the Power of Object Oriented Programming in LotusScript

- Remember to fill in your Evaluations.
 - ▶ This is how YOU influence Lotusphere 2008!

- Limited Time for Questions
 - ▶ We're available in the speaker room immediately after this session.

© IBM Corporation 2007. All Rights Reserved.

The workshops, sessions and materials have been prepared by IBM or the session speakers and reflect their own views. They are provided for informational purposes only, and are neither intended to, nor shall have the effect of being, legal or other guidance or advice to any participant. While efforts were made to verify the completeness and accuracy of the information contained in this presentation, it is provided AS IS without warranty of any kind, express or implied. IBM shall not be responsible for any damages arising out of the use of, or otherwise related to, this presentation or any other materials. Nothing contained in this presentation 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.

Performance is based on measurements and projections using standard IBM benchmarks in a controlled environment. The actual throughput or performance that any user will experience will vary depending upon many factors, including considerations such as the amount of multiprogramming in the user's job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve results similar to those stated here.

All customer examples described are presented as illustrations of how those customers have used IBM products and the results they may have achieved. Actual environmental costs and performance characteristics may vary by customer.

IBM, the IBM logo, Lotus, Lotus Notes, Notes, Domino, Domino.Doc, Domino Designer, Lotus Enterprise Integrator, Lotus Workflow, Lotusphere, QuickPlace, Sametime, WebSphere, Workplace, Workplace Forms, Workplace Managed Client, Workplace Web Content Management, AIX, AS/400, DB2, DB2 Universal Database, developerWorks, eServer, EasySync, i5/OS, IBM Virtual Innovation Center, iSeries, OS/400, Passport Advantage, PartnerWorld, Rational, Redbooks, Software as Services, System z, Tivoli, xSeries, z/OS and zSeries are trademarks of International Business Machines Corporation 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.

Lotus software

Microsoft and Windows are trademarks of Microsoft Corporation in the United States, other countries, or both.

Intel and Pentium are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

