10 Mistakes
You and Every XPages Developer Make
(Yes, I said YOU!)

Serdar Başeğmez (@sbasegmez)

Develop Information Systems

Serdar Başeğmez

- IBM Champion (2011-2015)
- Developi Information Systems, Istanbul
- Contributing…
  - OpenNTF / LUGTR / LotusNotus.com
- Featured on…
  - The View, NotesIn9
- Presented at…
  - IBM Connect and LUGs
- Also…
  - Blogger and Podcaster on Scientific Skepticism

IBM CHAMPION

developi
INFORMATION SYSTEMS

openntf

Lotus
KULLANICI GRUBU
www.lotusturkiye.org

Today

- Why are we here?
- Revisiting some key concepts
- Browser-Server interaction
- Repeat Control
- Custom Controls
- Dialogs
- Managed Beans
- Java

This Session is About…

- Problems of XPages developers.
- Some from personal experience
  - Workshops, conversations, other XPages apps, etc.
- Some from Stackoverflow…
  - Downloaded all Stackoverflow Q&A on #XPages tag.
  - Filtered code, applied some linguistic magic
  - Analyzed the most common word couples used together (NGram Analysis).

What Topics Asked in Stackoverflow…

The most common keywords in all questions from StackOverflow…

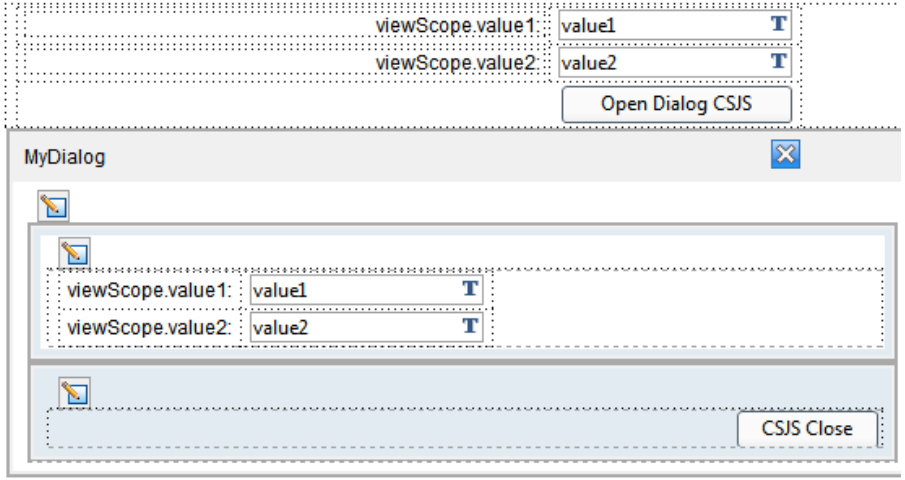
It’s XPages…

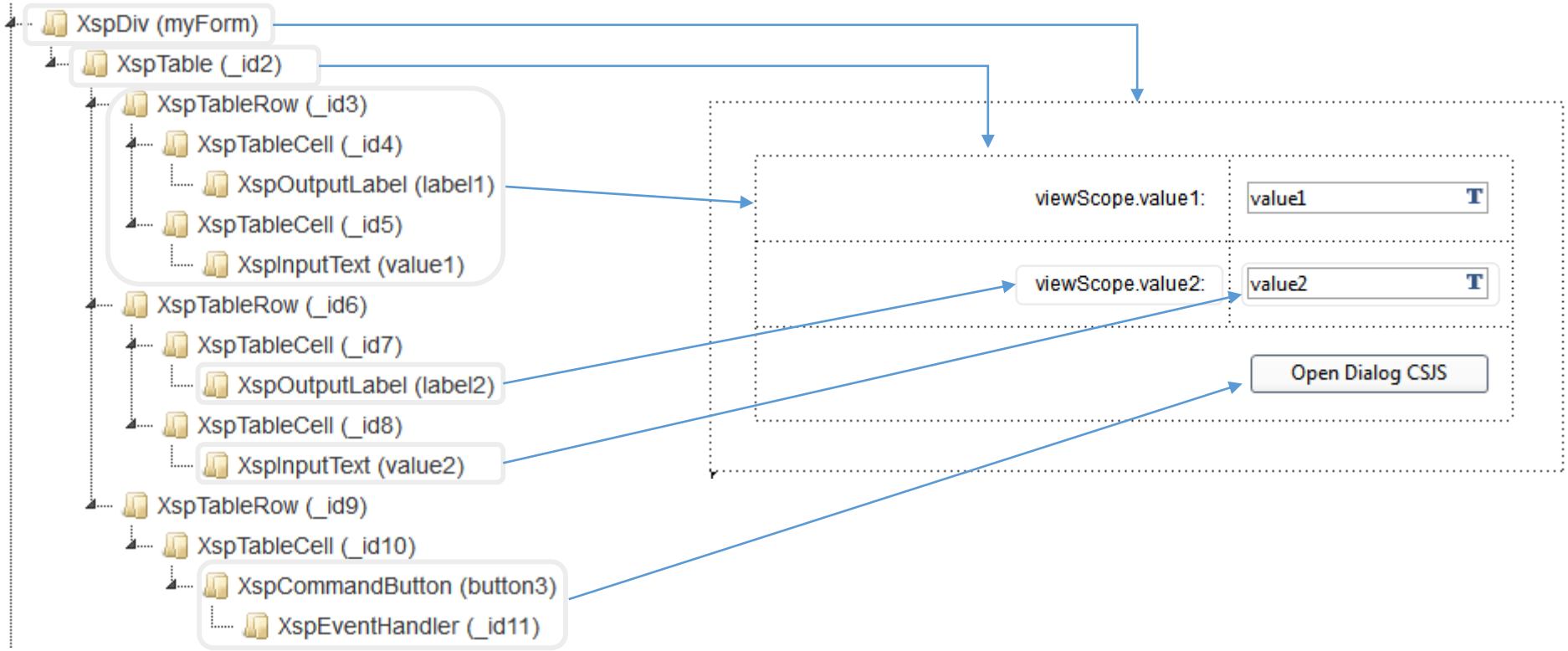
not xpages or Xpages or xPages ☺

Component Tree, View and Page States

revisiting some key concepts

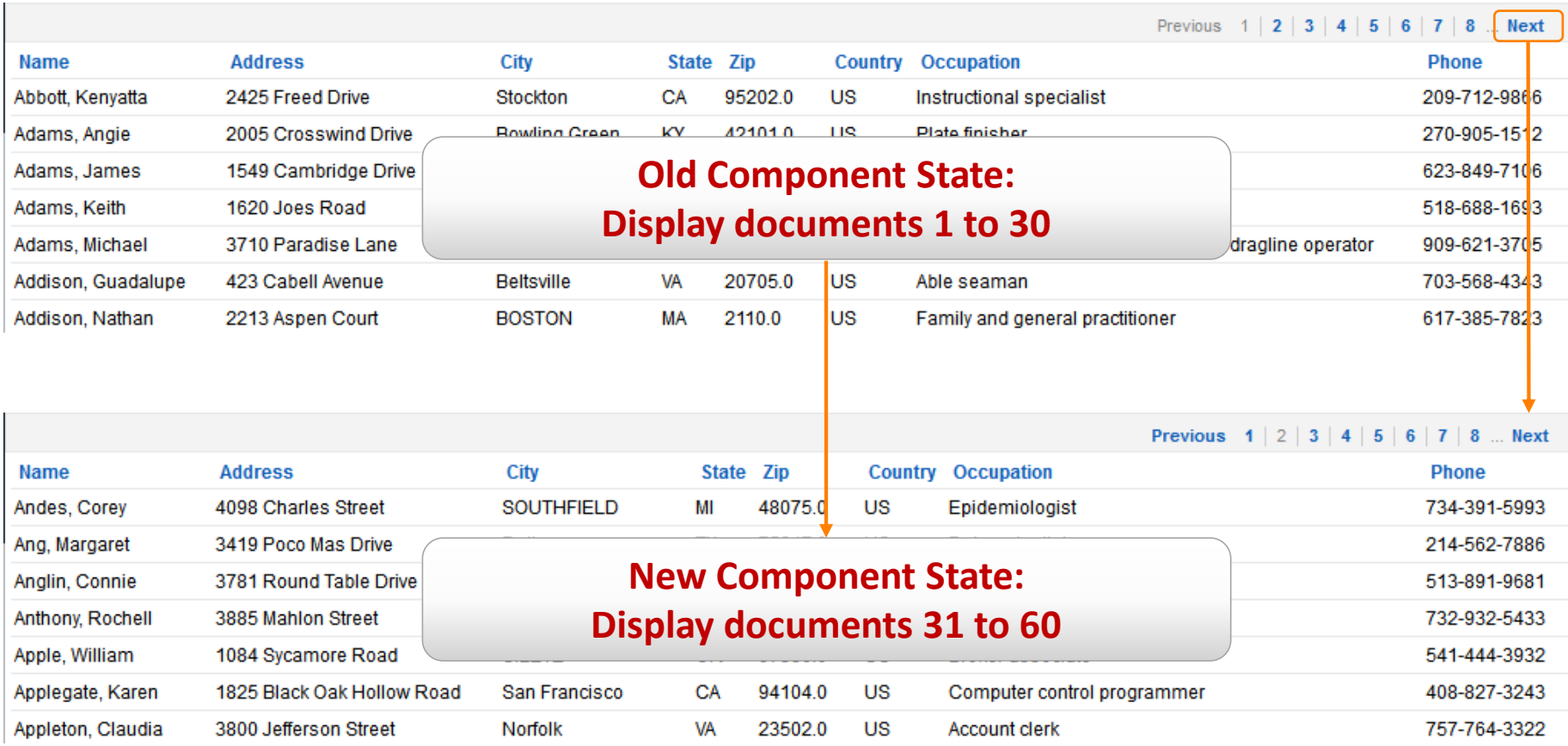
- UIViewRootEx2 → Root Component (Global «view» variable)
  - XspScriptCollector (_id12) → Collects all JavaScript imports used throughout the page.
  - XspForm (_id1) → Automatically added by default…
    - XspDiv (myForm)
      - XspTable (_id2)
        - XspTableRow (_id3)
          - XspTableCell (_id4)
            - XspOutputLabel (label1)
          - XspTableCell (_id5)
            - XspInputText (value1)
        - XspTableRow (_id6)
          - XspTableCell (_id7)
            - XspOutputLabel (label2)
          - XspTableCell (_id8)
            - XspInputText (value2)
        - XspTableRow (_id9)
          - XspTableCell (_id10)
            - XspCommandButton (button3)
              - XspEventHandler (_id11) → Event Handlers are components too
    - UIDialog (myDialog)
      - UIDialog$PopupContent (_content) → Some components does not inject all children initially




XspDiv (myForm)
XspTable (_id2)
XspTableRow (_id3)
XspTableCell (_id4)
XspOutputLabel (label1)
XspTableCell (_id5)
XspInputText (value1)
XspTableRow (_id6)
XspTableCell (_id7)
XspOutputLabel (label2)
XspTableCell (_id8)
XspInputText (value2)
XspTableRow (_id9)
XspTableCell (_id10)
XspCommandButton (button3)
XspEventHandler (_id11)
viewScope.value1:
value1
viewScope.value2:
value2
Open Dialog CSJS

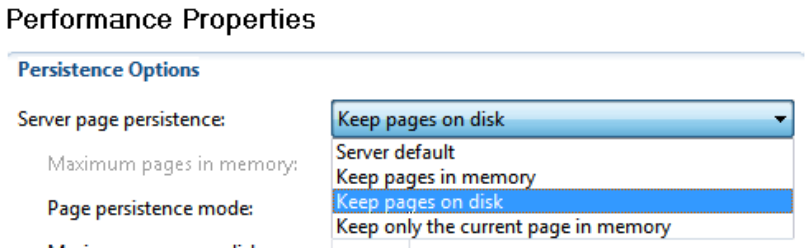
Components and Component Tree

- Every Component is Java magic…
  - Component is a Java Object.
  - Generally, it has a different renderer.
  - Sometimes it has other helper classes.
- Component Tree is an hierarchical representation
  - ‘Prepared’ during page load, ‘Revised’ between JSF phases.

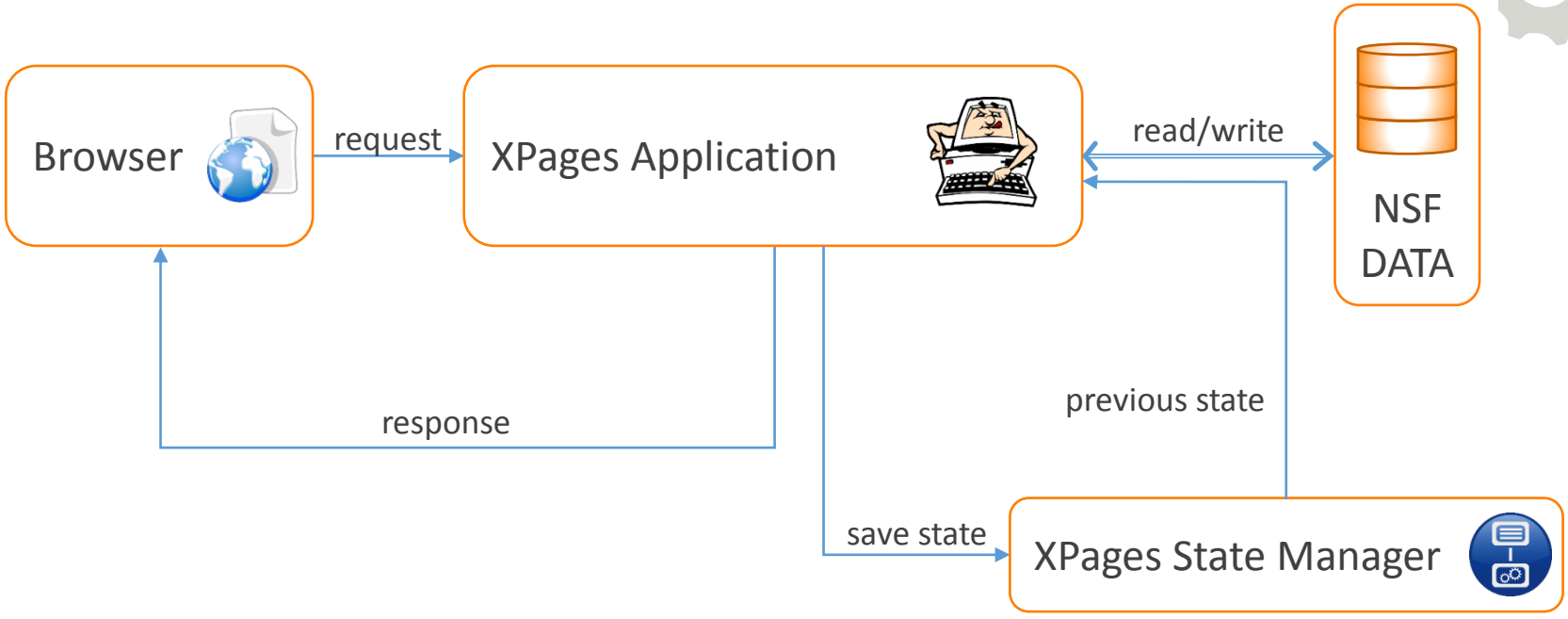

Previous 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 ... Next
Name Address City State Zip Country Occupation Phone
Abbott, Kenyatta 2425 Freed Drive Stockton CA 95202.0 US Instructional specialist 209-712-9866
Adams, Angie 2005 Crosswind Drive
Adams, James 1549 Cambridge Drive
Adams, Keith 1620 Joes Road
Adams, Michael 3710 Paradise Lane
dragline operator 909-621-3705
Addison, Guadalupe 423 Cabell Avenue Beltsville VA 20705.0 US Able seaman 703-568-4343
Addison, Nathan 2213 Aspen Court BOSTON MA 2110.0 US Family and general practitioner 617-385-7823
Old Component State:
Display documents 1 to 30
Previous 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 ... Next
Name Address City State Zip Country Occupation Phone
Andes, Corey 4098 Charles Street SOUTHFIELD MI 48075.0 US Epidemiologist 734-391-5993
Ang, Margaret 3419 Poco Mas Drive 214-562-7886
Anglin, Connie 3781 Round Table Drive 513-891-9681
Anthony, Rochell 3885 Mahlon Street 732-932-5433
Apple, William 1084 Sycamore Road 541-444-3932
Applegate, Karen 1825 Black Oak Hollow Road San Francisco CA 94104.0 US Computer control programmer 408-827-3243
Appleton, Claudia 3800 Jefferson Street Norfolk VA 23502.0 US Account clerk 757-764-3322
New Component State:
Display documents 31 to 60

Page State

- A page has a lifecycle in the server.
  - Persists its state as a **component tree**
  - Component tree also contains **viewScope**
- Page State is stored on Disk or Memory
  - Disk: Good for Scalability
  - Memory: Good for Performance
  - Only the current page in memory: Optimized
    - One page per each user kept in memory.



XPages Stateful Architecture

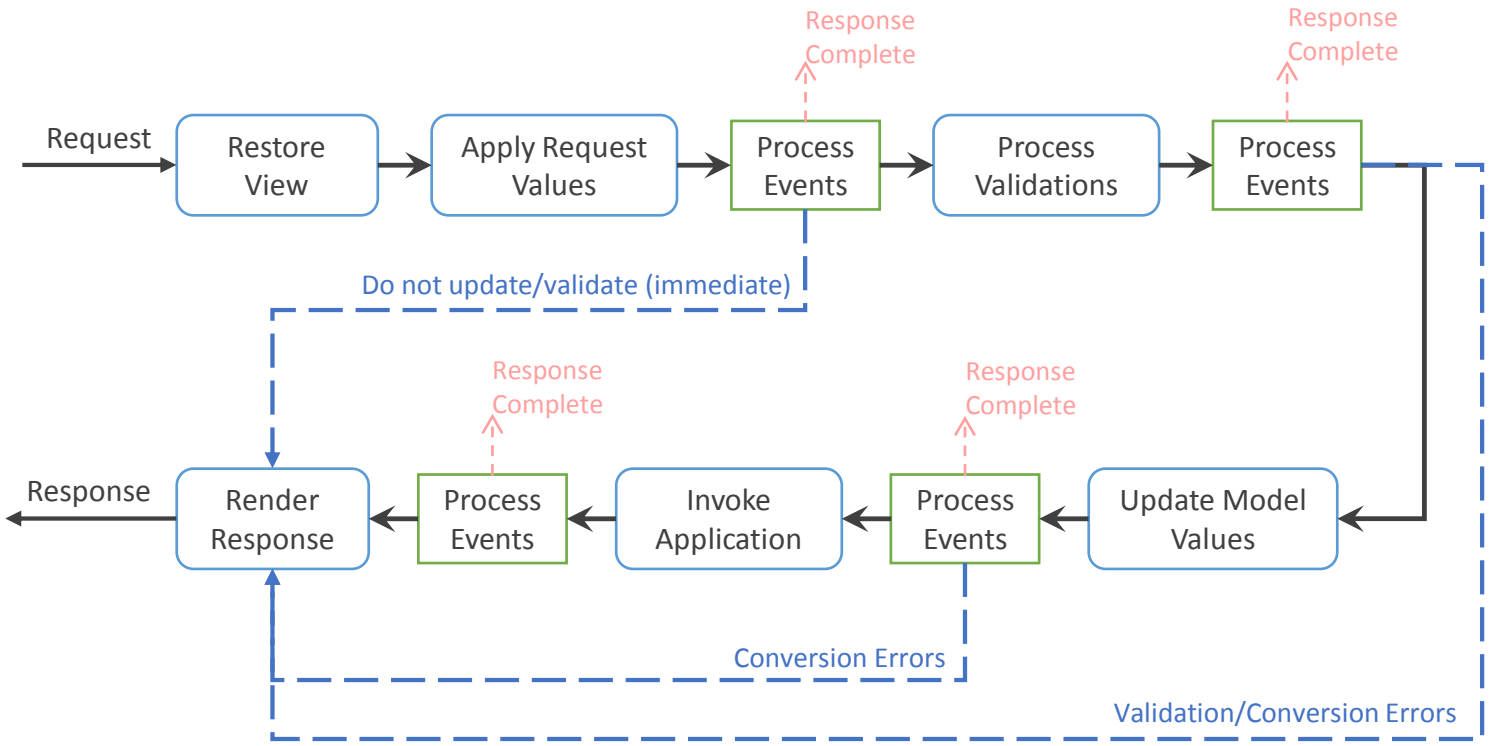


Each request contributes into a previous state…

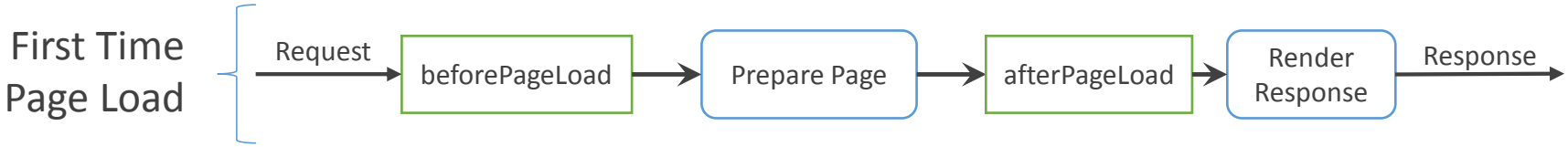
JSF Lifecycle

what happens under the cover?

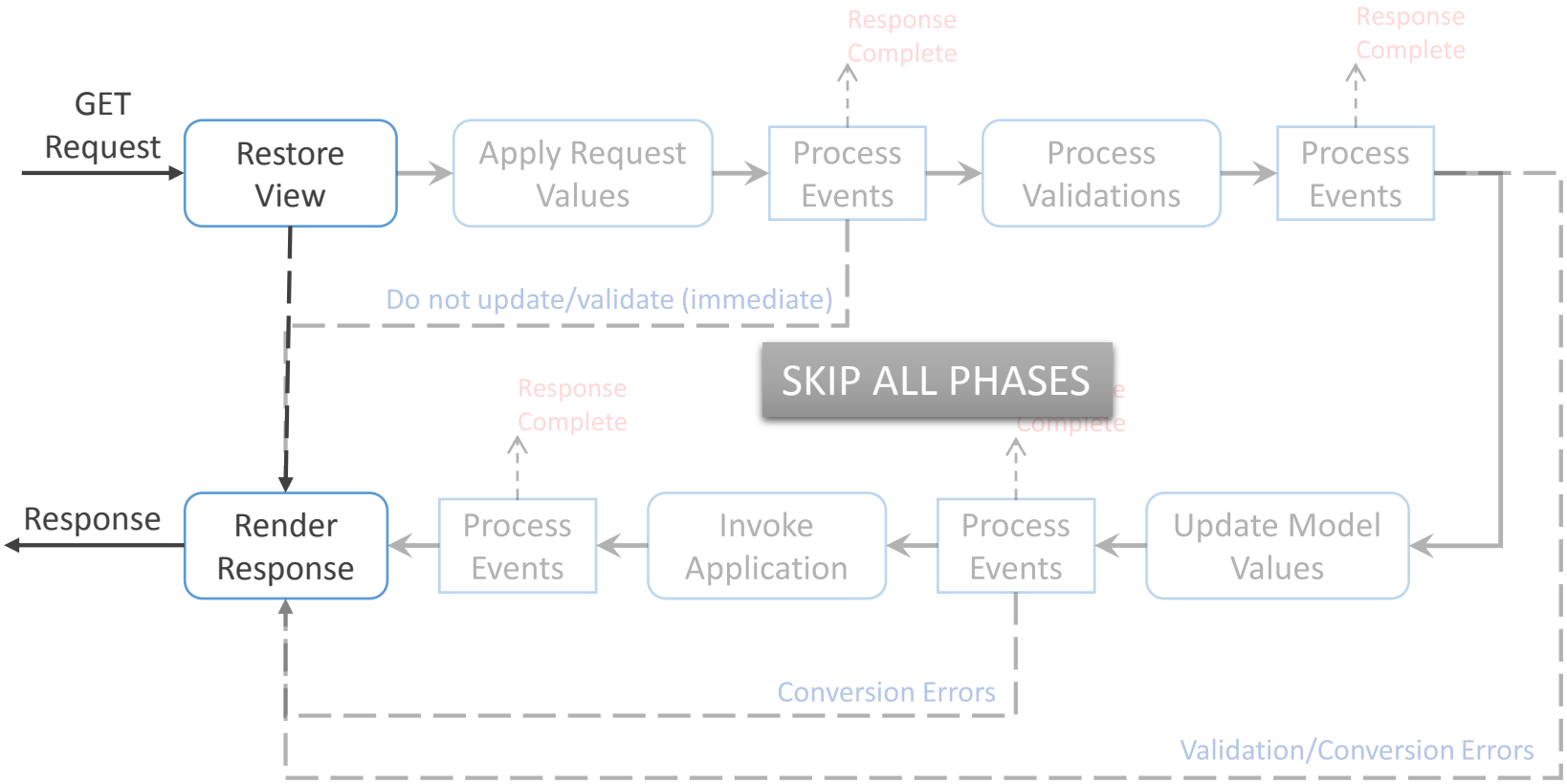
Full Request – Response Lifecycle



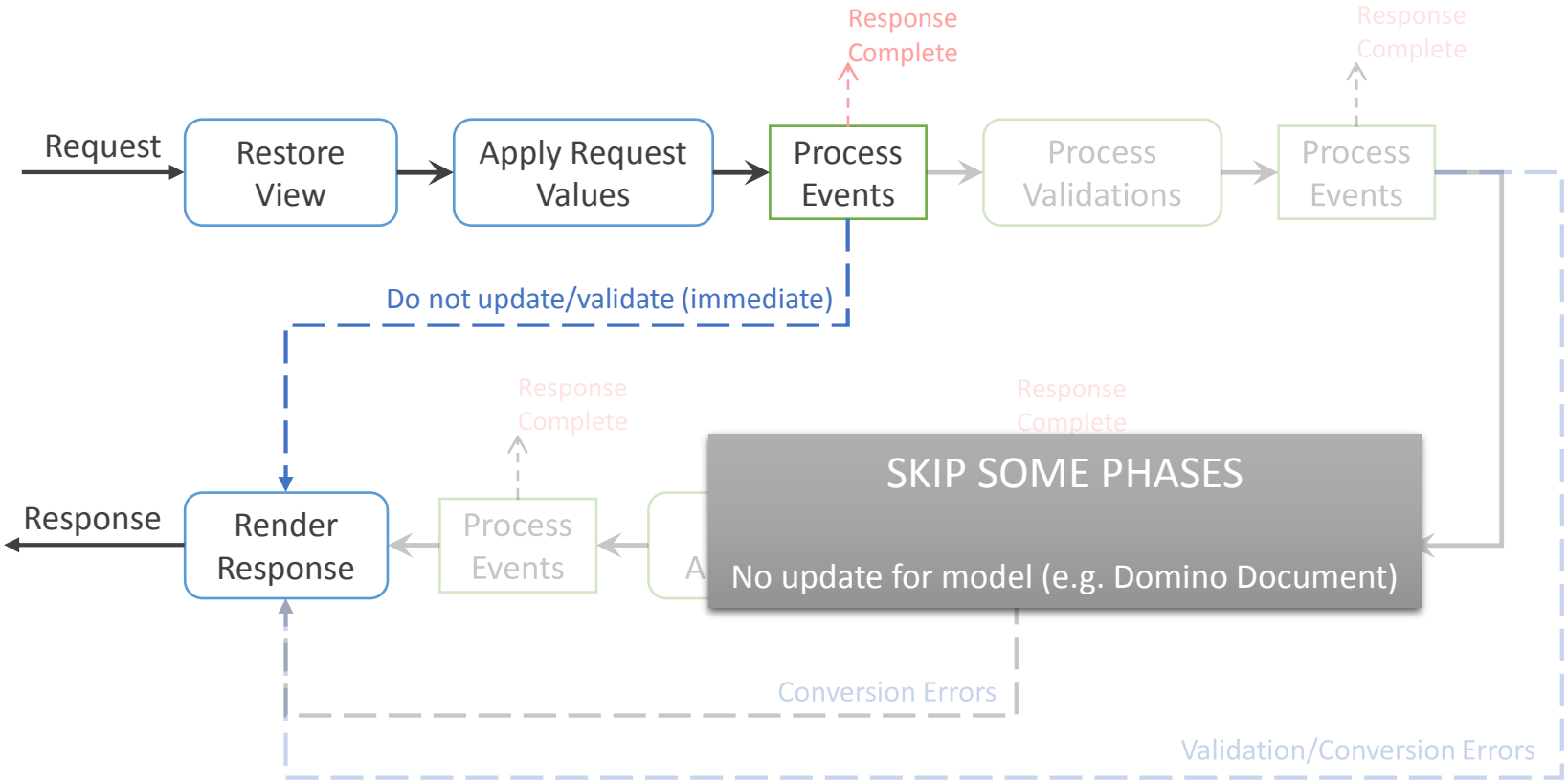
Full Request – Response Lifecycle



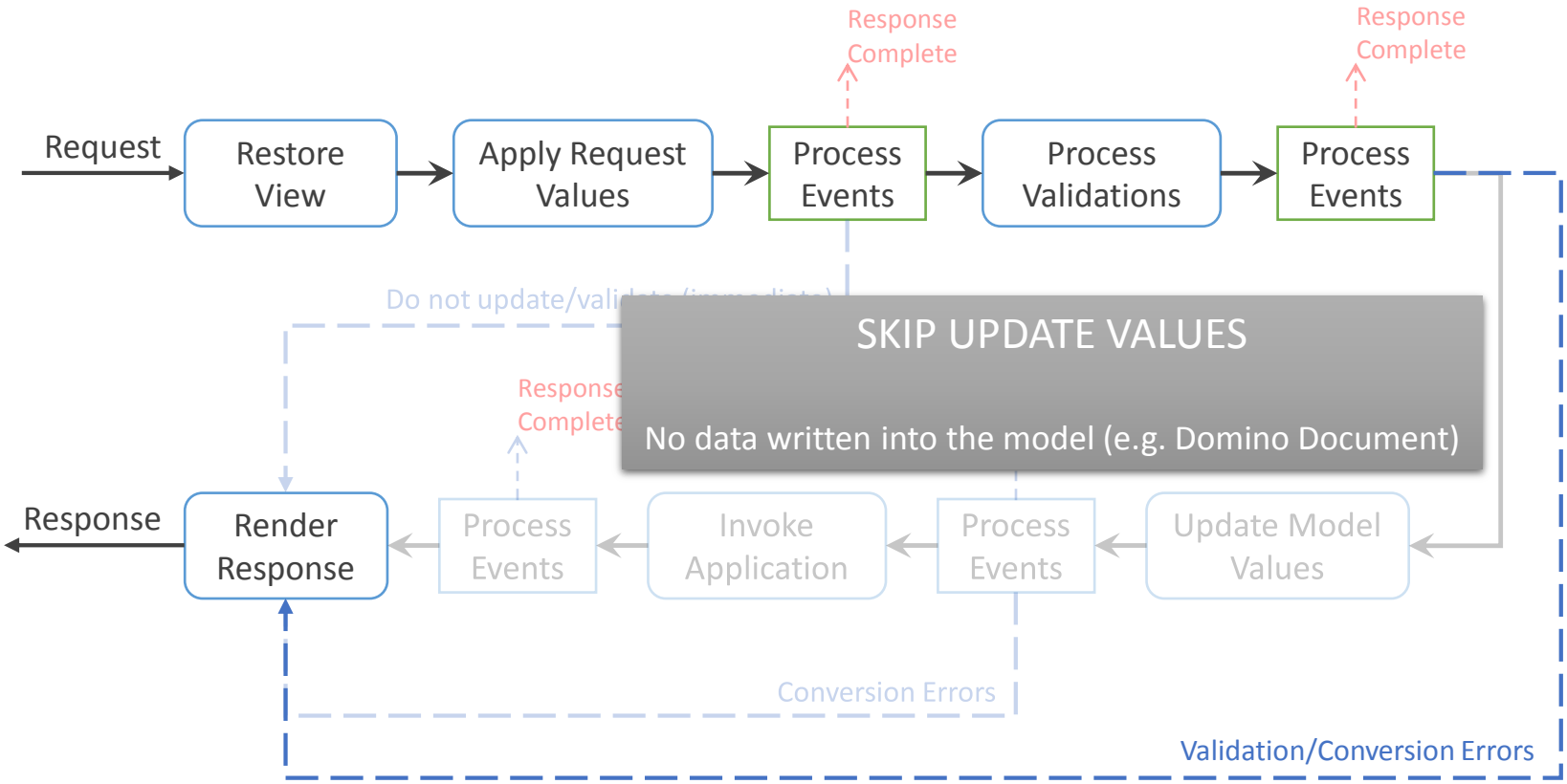
GET Requests – Lifecycle



Immediate Lifecycle



Lifecycle for Validation Error



Why Lifecycle is so Important…

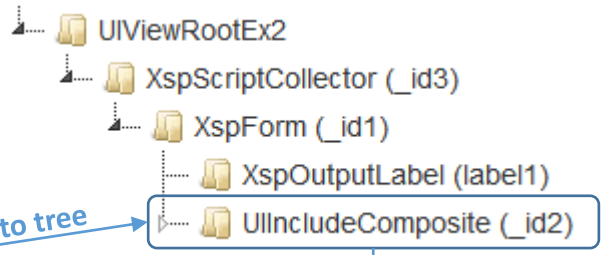
- Performance, scalability, etc.
  - Between phases, dynamic values are computed many times
  - Combine Lifecycles and Partial Execute model
  - Don’t underestimate the load of hundreds of users.
  - If no need for any update, use immediate.
  - If no need for any validation, skip validation.
  - More information on links…
- Troubleshooting very simple problems!
  - Validation/Update issues during Partial Refresh
  - e.g. when you use .getValue() or .getSubmittedValue()?

Dynamic or not: #{...} vs. ${...}

- #{…} bindings run whenever time they needed
  - Once at start, once between almost all phases!
  - Root causes of many performance issues.
    - Large view lookups for a combobox selection list?
- ${…} bindings run when the page is loaded
  - At the time XSP prepares the page tree.
  - Once in the entire lifecycle of the page.
  - Mandatory for some attributes (e.g. «loaded»).

"rendered" vs. "loaded"

```
<xp:view
     xmlns:xp="http://www.ibm.com/xsp/core"
     xmlns:xc="http://www.ibm.com/xsp/custom">
     <xp:label
          value="Here is a Hidden CC:"
          id="label1">
     </xp:label>

     <xc:ccBasic
          rendered="#{javascript:return false;}">
     </xc:ccBasic>

     <xc:ccBasic
          loaded="${javascript:return false;}">
     </xc:ccBasic>

</xp:view>
```



Added to tree

XSP ignores this CC…

When a CC included into the tree,
${…} bindings and page events will be evaluated!

XPages Scopes

Scope	Validity	Deletion
Application	Global* *(* Except local XPiNC apps)*	Idle timeout of the app. (xsp.application.timeout – 30 mins)
Session	Per browser session	Idle timeout of the session (**) (xsp.session.timeout – 30 mins)
View	Entire lifecycle of pages	Limited number of Pages per user (Disk: xsp.persistence.file.maxviews – 16) (Mem: xsp.persistence.tree.maxviews – 4)
Request	A single request	After Response Complete

Application and Session scopes consume JVM memory… Cache wisely!

XPages Scopes

- SessionScope is a Browser session
  - It's not related to authentication. Just a cookie…
    - Same user from different browser = Different Session
  - When you close the browser, you lose the session access.
  - If you logoff, things get complicated!
    - If anonymous access enabled, sessionScope cleared.
    - If another user logs in, sessionScope persists!
  - A big issue while testing the code with different users.
- All Scopes are defined per NSF application…
  - One application cannot access scopes of another!
  - NSF application is where your XPages code runs.

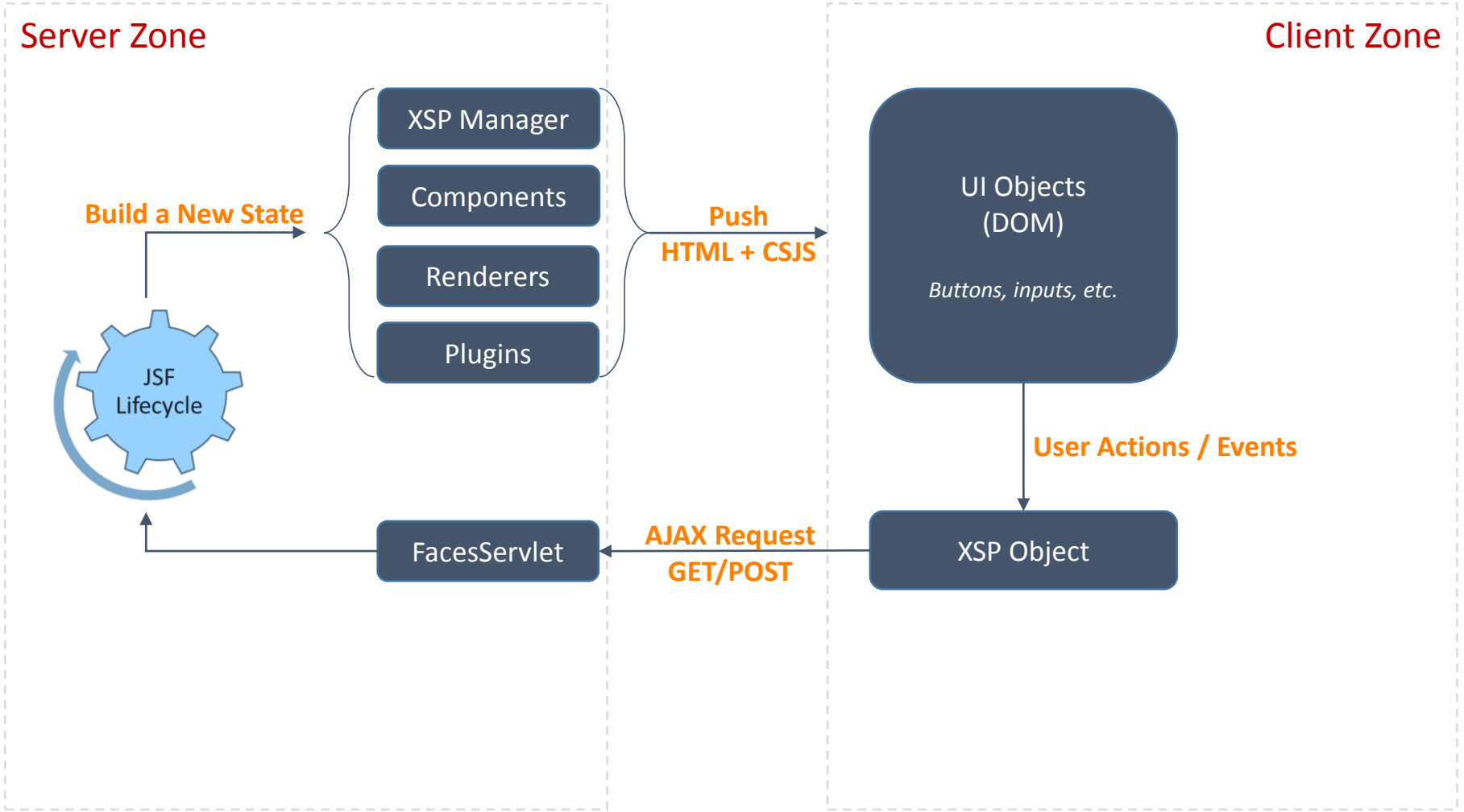
Browser – Server Interaction

two sides of the same coin

Browser – Server Interaction

- Understanding the XSP object…
  - CSJS API for browser-server interaction
  - Provides basic AJAX functionality
    - Checks preconditions, errors and timeouts, Aggregates necessary header and POST content, Guarantees only one AJAX request at a time,
  - Widgets extend XSP for additional functionality
    - e.g. dialog adds openDialog(…) and closeDialog(…)
- Event Handlers attach JavaScript to components

```
function view__id1__id13_clientSide_onclick(thisEvent) {
XSP.openDialog("view:_id1:myDialog")

}

XSP.addOnLoad(function() {
XSP.attachEvent("view:_id1:_id13", "view:_id1:button1", "onclick", view__id1__id13_clientSide_onclick, false, 2);
});
```


Server Zone
Client Zone
XSP Manager
Components
Renderers
Plugins
Build a New State
JSF Lifecycle
Push
HTML + CSJS
UI Objects (DOM)
Buttons, inputs, etc.
User Actions / Events
FacesServlet
AJAX Request
GET/POST
XSP Object

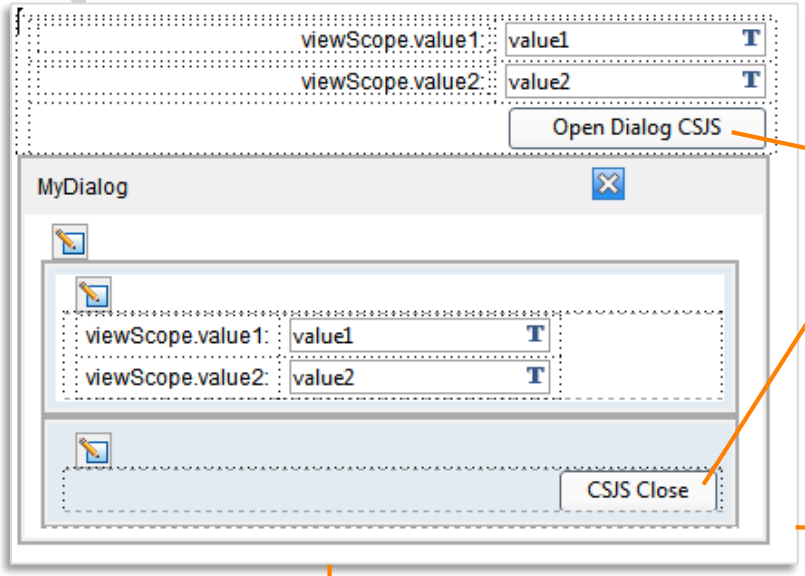
<xe:dialog/>

a potential code-breaker in wrong hands…

The heir of the Dojo Dialog

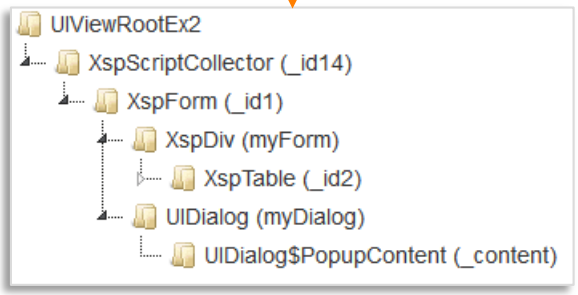
- Ext.Lib. Dialog is an extended Dojo Dialog…
  - Dojo Dialog is incompatible to the XPages.
  - Dojo injects the dialog out of the <form> tag.
    - From XPages standpoint, this is a problem.
    - So Ext.Lib uses a custom interaction mechanism for dialogs.
- Dialog Component as an Example
  - Open/Close dialogs from SSJS or CSJS???
  - Let's zoom at what's going on with the Dialog…

A Basic Page with a Dialog



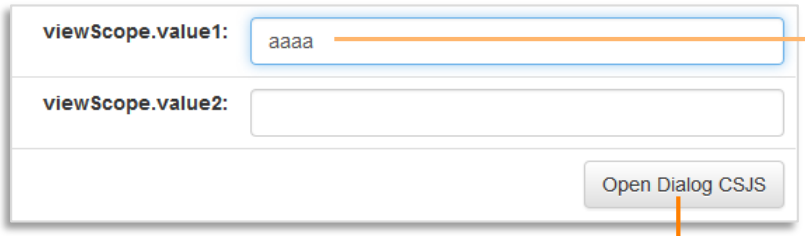

```
XSP.openDialog("#{id:myDialog}")
```

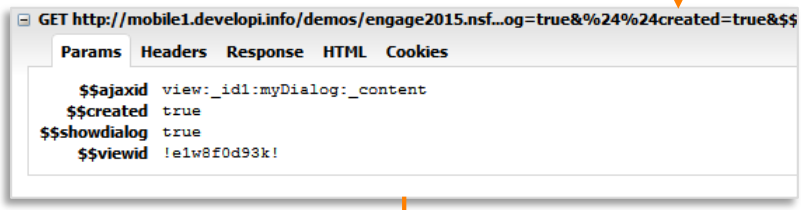
```
XSP.closeDialog("#{id:myDialog}", "#{id:myForm}")
```

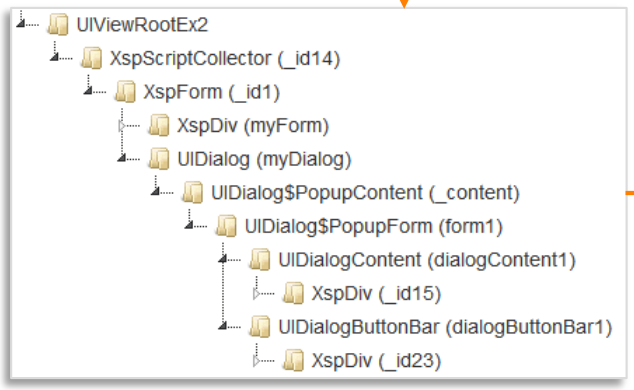



```
<script type="text/javascript">
function view__id1__id13_clientSide_onclick(thisEvent) {
XSP.openDialog("view:_id1:myDialog")
}

XSP.addOnLoad(function() {
XSP.attachEvent("view:_id1:_id13", "view:_id1:button1",
"onclick", view__id1__id13_clientSide_onclick, false, 2);
});
</script>
```

Showing Dialog via CSJS

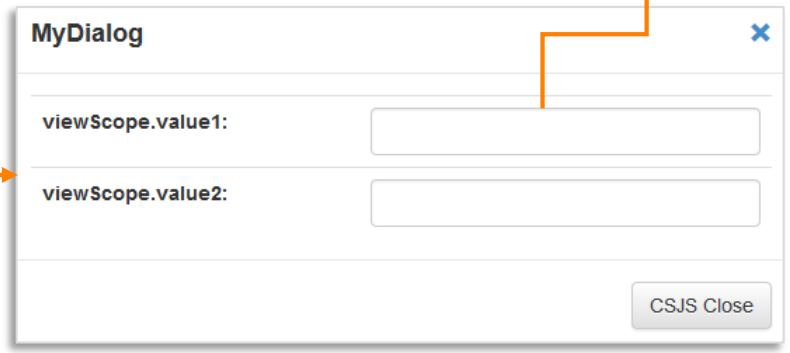






- XSP.openDialog() sends a GET request
- The dialog prepares the contents
- The response is added to the DOM.

Where is the value???



Common Issues with Dialogs

- When we open a dialog from CSJS…
  - XSP.openDialog() sends a GET request to the server
  - The dialog prepares the contents and sends back.
  - The result is added to the DOM.
- It DOES NOT submit any value to the server…
  - Server is not aware of any changes in other field values.
- Almost the same when you close the dialog…
  - XSP.closeDialog(dialogId, refreshId): Closes the dialog and partially refresh a component. No POST request!
- Use SSJS if you need to submit values…

Common Issues with Dialogs

- Using SSJS to open/close dialogs

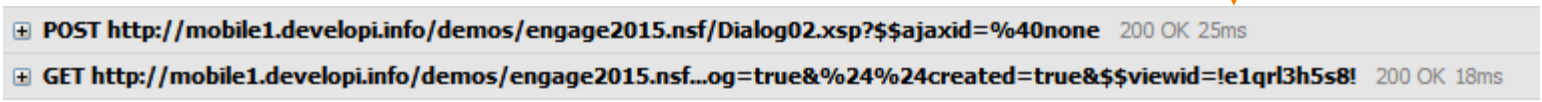
```
<xp:button
     value="Open Dialog SSJS"
     id="button3">
     <xp:eventHandler
          event="onclick"
          submit="true"
          refreshMode="partial"
          refreshId="button3">                    <!- It should refresh something! -->
          <xp:this.action><![CDATA[#{javascript:
               getComponent("myDialog").show()
          }]]></xp:this.action>
     </xp:eventHandler>
</xp:button>
<xp:button
     value="SSJS Close"
     id="button4">
     <xp:eventHandler
          event="onclick"
          submit="true"
          refreshMode="partial"
          refreshId="myForm">                     <!- We refresh the form so it reflects changes -->
          <xp:this.action><![CDATA[#{javascript:
               getComponent("myDialog").hide()
          }]]></xp:this.action>
     </xp:eventHandler>
</xp:button>
```

Common Issues with Dialogs

- If you really need to use CSJS…
  - You can initiate a POST request first.

```
<xp:button
     value="Open Dialog CSJS"
     id="button1">
     <xp:eventHandler
          event="onclick"
          submit="false">
          <xp:this.script><![CDATA[
               XSP.partialRefreshPost("@none", {
                    "onComplete": function() {
                         XSP.openDialog("#{id:myDialog}");
                    }
               });
          ]]></xp:this.script>
     </xp:eventHandler>
</xp:button>
```

- Submit a POST request
- Refresh no element (refresh needed on close)
- When you're done, launch my dialog…
- Attention: It took two AJAX requests!



Common Issues with Dialogs

- Need a single value? Here is a trick…
  - You can provide parameters to the dialog on CSJS…
  - XSP.openDialog(dialogId, dialogOptions, dialogParams)

```
<xp:button
     value="Open Dialog"
     id="button1">
     <xp:eventHandler
          event="onclick"
          submit="false">
          <xp:this.script><![CDATA[
               XSP.openDialog(
                    "#{id:myDialog}",    /* dialogId */
                    {},                  /* options */
                    {"tab" : "tab1" }    /* params */
               );
          ]]></xp:this.script>
     </xp:eventHandler>
</xp:button>
```

Adds a query string parameter to the GET request…

We can use it within the dialog…

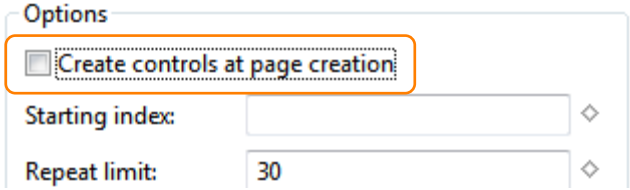
```
(…)
<xp:tabbedPanel id="tabbedPanel1" selectedTab="#{param.tab}">
(…)
```

<xp:repeat/>

what's wrong with this mysterious component?

The Basics of <xp:repeat/>

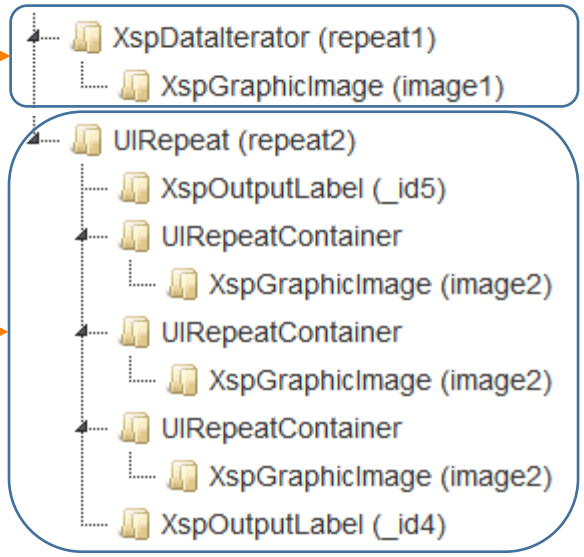
- Repeat component iterates its child components
  - «value» attributes should be an iterable component (collection, array, list, etc.)
  - «first» and «rows» attributes for the start index and count.
  - It can be binded to any pager component.
  - Renders "header" and "footer" facets.
- Repeat has two modes
  - DataIterator mode (default)
  - Repeat mode (when «repeatControls="true"»)



Two Modes of <xp:repeat/>

```
<xp:repeat
     id="repeat1" var="value"
     value="#{repeatValues}">
     <xp:this.facets>
          <xp:label xp:key="header" value="START:" />
          <xp:label xp:key="footer" value=":END" />
     </xp:this.facets>
     <xp:image id="image1" url="#{value}" />
</xp:repeat>

<xp:repeat
     id="repeat2" var="value"
     repeatControls="true"
     value="#{repeatValues}">
     <xp:this.facets>
          <xp:label xp:key="header" value="START:" />
          <xp:label xp:key="footer" value=":END" />
     </xp:this.facets>
     <xp:image id="image2" url="#{value}" />
</xp:repeat>
```

Does not inject new components.
Only one set of child components, render multiple times



Injects multiple containers for every iteration.

Implications of «repeatControls»

- Child component limitations in DataIterator mode:
  - ${…} bindings will be processed only once.
  - ${…} bindings cannot consume repeat variables.
    - ${value} will be ignored.
    - ${javascript:value} will throw error.
- Repeat mode:
  - Provides computed id's for iterated fields (e.g. Radio)
  - Iterated columns for Data Tables
  - Limitation: Child elements are fixed, can't be changed later.

<xc:customControls/>

do you miss subforms?

Custom Controls

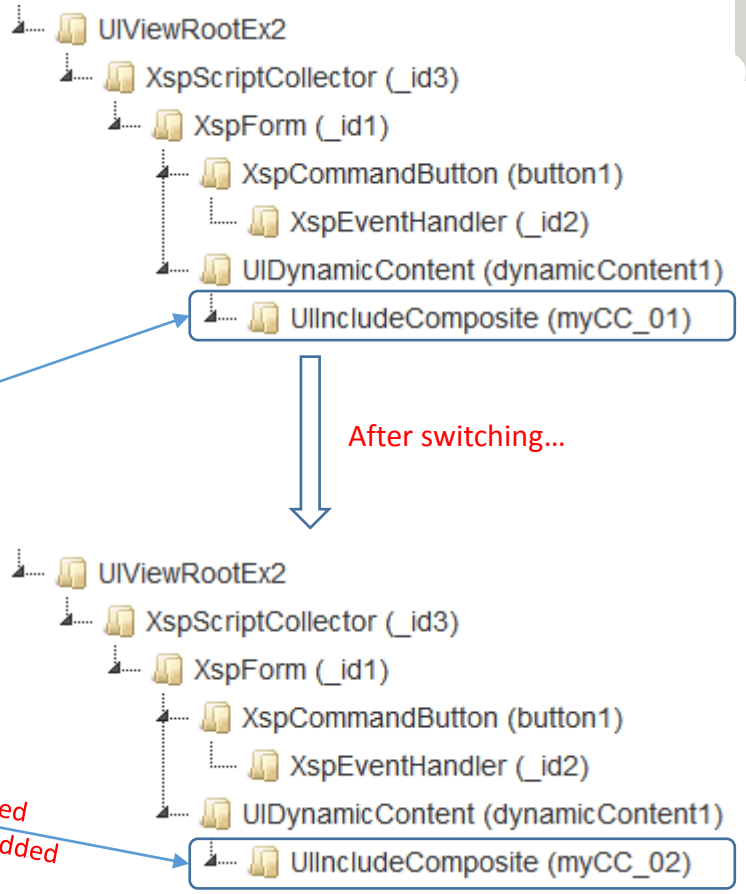
- Specific to XPages…
  - IBM's implementation of JSF Composite Components.
  - Similar to Subforms (but much better!)
- Injects XPages into your XPages…
  - Custom Controls are XPages…
  - Reusable components with their own **xsp-config** files.
  - They support parameters, designer extensions, etc.

‘Multiple Choice’ Custom Controls

- Among multiple CCs, one will be displayed.
  - If you can decide before loading page…
    - URL parameter, user role, data source, etc.
    - Static «loaded» can be used…
  - Want to switch during run-time?
    - According to an input, based on a dynamic value, etc.
    - Dynamic «rendered» values preferred mostly.
    - BUT, as shown before, this will evaluate the CC!
      - All ${…} bindings and pageLoad events

A Better Approach: <xe:dynamicContent>

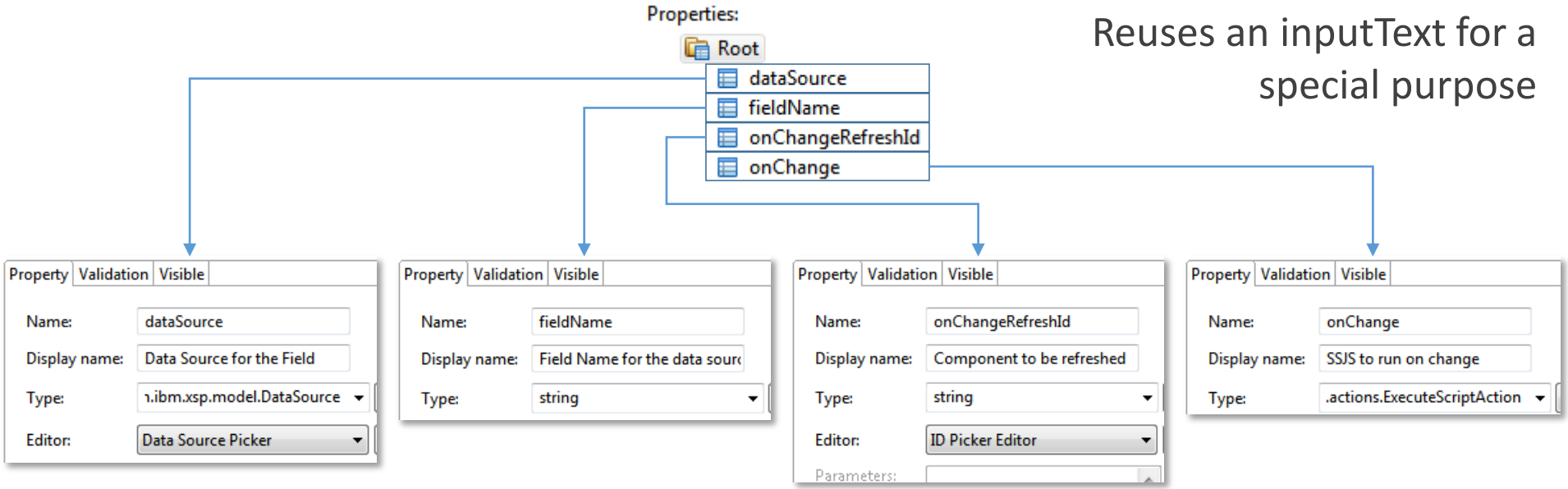
```
<xp:button
     value="Switch Dynamic Content"
     id="button1">
     <xp:eventHandler
          event="onclick"
          submit="true"
          refreshMode="complete">
          <xp:this.action>
               <xe:changeDynamicContentAction
                    for="dynamicContent1"
                    facetName="cc02">
               </xe:changeDynamicContentAction>
          </xp:this.action>
     </xp:eventHandler>
</xp:button>
<xe:dynamicContent
     id="dynamicContent1"
     defaultFacet="cc01">
     <xp:this.facets>
          <xc:ccBasic
               xp:key="cc01"
               id="myCC_01"></xc:ccBasic>
          <xc:ccBasic
               xp:key="cc02"
               id="myCC_02"></xc:ccBasic>
     </xp:this.facets>
</xe:dynamicContent>
```



Common Issues for Custom Controls

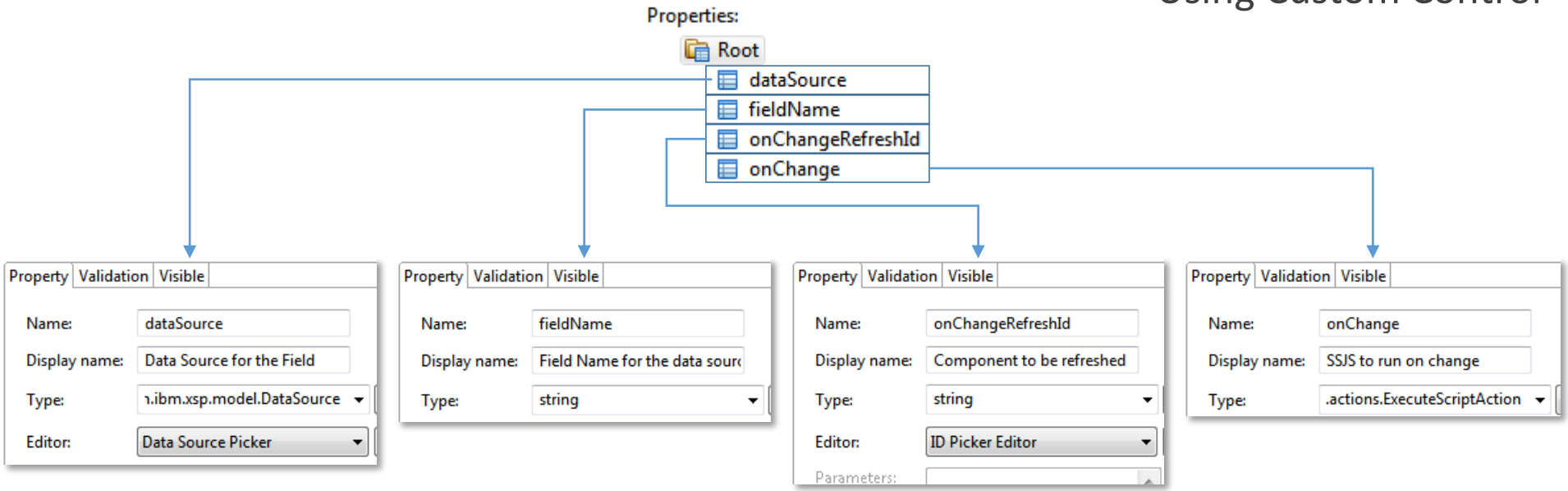
- Black box approach with Custom Properties…
  - Prioritize Reusability.
  - Do not hardcode ‘things’ inside your custom control…
    - Parent XPage – Custom Control interactions
    - Interact to the data sources or components from the parent XPage
    - Trigger an event from the parent XPage?

A Custom Control
Reuses an inputText for a special purpose

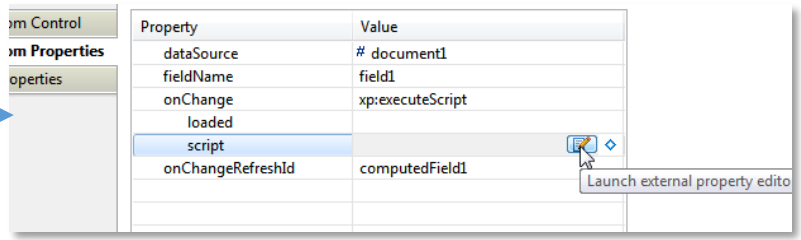



```
<xp:inputText
   id="inputText1"
   value="#{compositeData.dataSource[compositeData.fieldName]}">
   <xp:eventHandler
      event="onchange" submit="true"
      refreshMode="partial" refreshId="#{compositeData.onChangeRefreshId}">
      <xp:this.action><![CDATA[#{javascript:
      var handler = compositeData.onChange;
      if(null!=handler && typeof(handler)=='com.ibm.xsp.actions.ExecuteScriptAction') {
         handler.setParamNames(['value']);
         handler.invoke(facesContext, [this.getParent().getValue()]);
      }
   }]]></xp:this.action>
   </xp:eventHandler>
</xp:inputText>
```

Using Custom Control

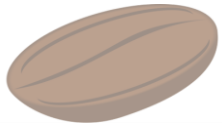



```
<xc:ccInput
    dataSource="#{document1}"
    fieldName="field1"
    onChangeRefreshId="computedField1">

    <xc:this.onChange>
        <xp:executeScript>
            <xp:this.script><![CDATA[#{javascript:
                print("Field1 >>> " + value);
            ]]></xp:this.script>
        </xp:executeScript>
    </xc:this.onChange>

</xc:ccInput>
```



Managed Beans

- approaching to the Java zone -



Managed Beans - Basics

- JavaBean is a simple Java class…
  - Empty constructor, Serializable
  - Has Getter and Setter methods to access its private fields
- Managed Bean is a JavaBean that is managed.
  - It has a name and scope definition.
  - The platform is responsible of the lifecycle.
  - They are created on their first use…
  - Single Instance guaranteed, but not Thread-safe!
- Managed Beans should be Serializable!
  - It's a MUST for the View Scope
  - Best Practice in General!

Common Issues with Managed Beans

```
public Bean01() {
    Database myDb = ExtLibUtil.getCurrentDatabase();
    try {
        View myView = myDb.getView("Test01");
        Document myDoc = myView.getDocumentByKey("key");

        // ... do stuff

    } catch (NotesException e) {
        e.printStackTrace();
    }
}
```

javax.faces.FacesException: javax.faces.FacesException: Can't instantiate class: 'demo.Bean01'.. java.lang.ClassNotFoundException: class java.lang.NullPointerException: null

- Error handling in Constructor…
  - Managed beans are instantiated via Reflection (long story).
  - If constructor fails, it throws meaningless errors.
    - No stack trace for constructor exceptions!
    - USE a proper try-catch block in your constructor.
    - It's important to provide an informative message.
    - Have an error page and/or logging mechanism…

Common Issues with Managed Beans

```
public class Bean01 implements Serializable {

    private static final long serialVersionUID = 1L;

    private long timeCreated;

    private ApplicationSetupBean appSetup;

    private DocumentCollection cachedDocs;

    private transient Database myDb;

    public Bean01() {
        // Stuff
    }

    // Some other stuff

}
```

Do not store other beans!

Never Store Domino Objects!

Transient is also a bad idea!

Common Issues with Managed Beans

- «Do-not-store-these» within managed beans.
  - Never store ANY **Domino objects**!
    - Domino objects has C-handles on back-end. They will be recycled after each request-response cycle.
    - Only for view-scoped beans, you can define transient fields but it's a bad habit!
    - Always store the reference (UniqueId, dbPath, etc.)
  - Never store another bean or static Java object.
    - During lifecycle or design-change objects might be reconstructed.
  - Many of these objects are cheap to reproduce!

Common Issues with Managed Beans

- Memory usage concerns are frequently asked.
- Especially important for caching beans…
  - Application and session scope beans live in the memory
  - View scope beans are stored in either memory or disk
    - It depends on the persistence setting.
  - Request scope is short-lived.
- Cost-benefit analysis for using memory vs. time
  - Caching more information needs more memory.
  - Not caching anything is processing time and I/O activity
- Get creative ☺

Use of Java

welcome to the magical world

Latin Small Letter Dotless-i

- ".toLowerCase()" and ".toUpperCase()"
  - "DOMINO".toLowerCase() = "domıno" on my server!!!
  - Use Locale parameter… e.g. ".toLowerCase(Locale.US)"
  - There are millions of computers with Turkish locale! ☺

i → İ

I → ı

For this section…
If you use OpenNTF Domino API…

Smile towards others

Common Issues with Java

- Recycle… Recycle… Recycle…
  - Domino objects keep a backend C-handle. It's limited!
  - Recycle every Domino objects you have used,
    - …except Session and the current database…
  - Recycle in «finally {…}» section.
  - Also recycle in SSJS.
- Tired of recycling?
  - Use OpenNTF Domino API…

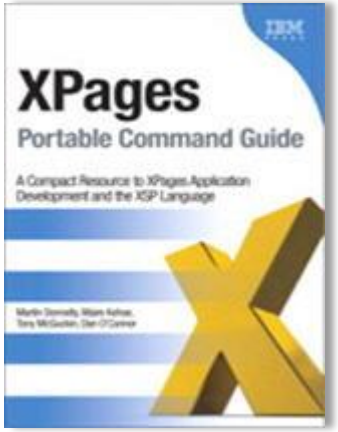
Remember, there is a great team behind the Project.
…and they will be thirsty tonight ☺

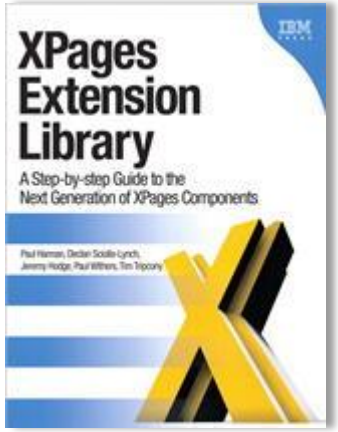
Resources

additional readings

Books

- Mastering XPages 2nd Edition
  http://www.redbooks.ibm.com/Redbooks.nsf/ibmpressisbn/9780133373370?Open
- XPages Portable Command Guide
  http://www.redbooks.ibm.com/Redbooks.nsf/ibmpressisbn/9780132943055?Open
- XPages Extension Library
  http://www.redbooks.ibm.com/Redbooks.nsf/ibmpressisbn/9780132901819?Open





XPages Request Process Lifecycle (X-RPL)

- Blog Series: Understanding Partial Execution, Paul S. Withers
  - Part One – Refresh
    http://www.intec.co.uk/understanding-partial-execution-part-one-refresh/
  - Part Two – Execution
    http://www.intec.co.uk/understanding-partial-execution-part-two-execution/
  - Part Three – JSF Lifecycle
    http://www.intec.co.uk/understanding-partial-execution-part-three-jsf-lifecycle/

- Also check these XSnippets by Tony McGuckin:
  - XPages Request Processing Lifecycle explorer code
    http://openntf.org/XSnippets.nsf/snippet.xsp?id=xpages-request-processing-lifecycle-explorer-code...
  - dynamicContent - Efficient loading/rendering of a Component Tree
    http://openntf.org/XSnippets.nsf/snippet.xsp?id=dynamiccontent-efficient-loadingrendering-of-a-component-tree

Performance and Scalability

- IBM ConnectED 2015 - MAS103 XPages Performance and Scalability, Paul S. Withers & Tony McGuckin
  http://www.slideshare.net/paulswithers1/mas103-xpages-performance-and-scalability
- XPages Performance: pro tips, Nathan T. Freeman
  https://nathantfreeman.wordpress.com/2013/04/12/xpages-performance-pro-tips
- BLUG 2013: Life In The Fast Lane: Full Speed XPages, Ulrich Krause
  http://www.slideshare.net/eknori/blug-2013final
- XPages Tip: Realizing the Cost of a Value Binding, Serdar Basegmez
  http://lotusnotus.com/lotusnotus_en.nsf/dx/xpages-tip-realizing-the-cost-of-a-value-binding....htm

Miscellaneous

- IAmLug 2013: Managed Beans: When, Why and How, Russel Maher
  http://www.slideshare.net/RussellMaher/managed-beans-when-why-and-how
- Tim Explains: SSJS Object Persistence, David Leedy
  http://www.notesin9.com/2014/05/20/tim-explains-ssjs-object-persistence/
- Meet the XSP object, Stephan H. Wissel
  http://www.wissel.net/blog/d6plinks/SHWL-878B9D
- Stackoverflow Answer to "What is the best way to recycle Domino objects in Java Beans", Tim Tripcony
  http://stackoverflow.com/a/11160925/1167922
- Stackoverflow Answer to "recycle domino objects after they are lost", Paul S. Withers
  http://stackoverflow.com/a/28827606/1167922

Miscellaneous

- Accessing the Value of Components within a Repeat Control from Outside, Brad Balassaitis
  http://xcellerant.net/2013/07/29/access-repeat-components-from-outside
- Why it is important to use the Recycle() method on every Java object
  http://www-01.ibm.com/support/docview.wss?uid=swg21097861
- Sessions, logout, sessionScope and userScope, Paul S. Withers
  http://http://www.intec.co.uk/sessions-logout-sessionscope-and-userscope/
- "Dotless i", toLowerCase and toUpperCase functions: Use responsibly!, Serdar Basegmez
  http://lotusnotus.com/lotusnotus_en.nsf/dx/dotless-i-tolowercase-and-touppercase-functions-use-responsibly.htm

The End

ask questions now

get social later

@sbasegmez