



## Web dialogs as drop-in components

*Killing the monolithic webapp*

Håkan Dahl, Callista Enterprise AB



## Agenda

- Background
- Problem
- Vision
- Demo
- Solution
- Conclusion

CADEC2007, Web dialogs as drop-in components, Slide 2  
Copyright 2007, Callista Enterprise AB



## Background

- Concept born last year on Callista's blog
  - response to typical web-related problems
  - web-navigation is tricky!
  - recurring pattern in customer projects
- Valid context is medium-to-large web-applications

CADEC2007, Web dialogs as drop-in components, Slide 3  
Copyright 2007, Callista Enterprise AB



## The problem

Web application development generally has these problems:

- no re-use in presentation layer (*at dialog level*)
- no modularization
- navigation and state management

resulting in products with characteristics:

- monolithic
  - a single big piece of software containing all complexity
  - implicit dependencies (*brittle to change*)
  - no parts can be re-used (*violating the "Don't Repeat Yourself" principle*)
- expensive maintenance

CADEC2007, Web dialogs as drop-in components, Slide 4  
Copyright 2007, Callista Enterprise AB



## The vision

Composite web applications built from modules where:

- dialogs are included in a webapp as jar-files (*components*)
- no special config (in web.xml etc) needed to include a jar
- dialogs can be put together to form "modal" dialogs
- the webapp (war-file) is only an assembly
- robust navigation and state management

Advantages:

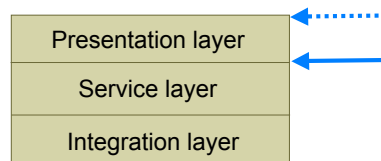
- encapsulation of well defined functionality (high coherence)
- enables re-use
- components can be developed and tested "standalone"

CADEC2007, Web dialogs as drop-in components, Slide 5  
Copyright 2007, Callista Enterprise AB



## The vision - raising the bar for re-use

- The level of re-use is traditionally at the service layer
  - but we are aiming higher!

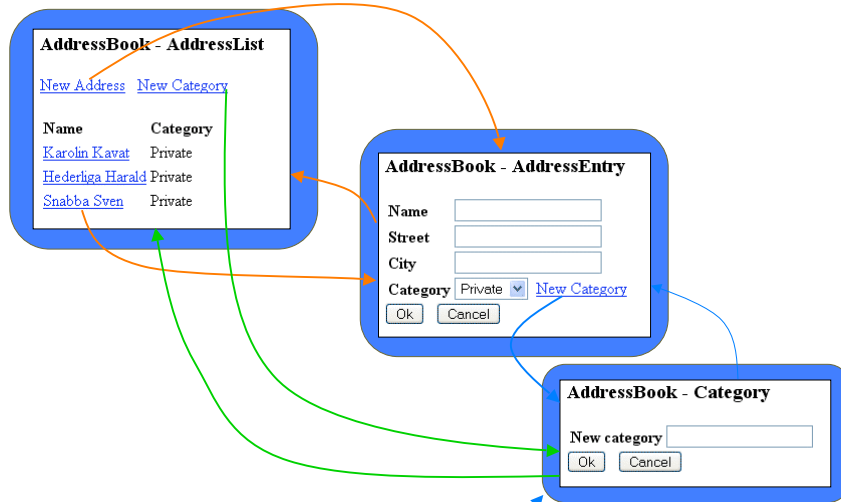


- Typical re-use of presentation would be in:
  - modal dialogs (*the webapp equivalent to modal dialogs*)
  - search dialogs
  - other webapp's in the same system (*for example admin-app*)

CADEC2007, Web dialogs as drop-in components, Slide 6  
Copyright 2007, Callista Enterprise AB



## Navigation flow example and the components!



CADEC2007, Web dialogs as drop-in components, Slide 7  
Copyright 2007, Callista Enterprise AB

re-used!



## Demo

**Proof-of-Concept!**

CADEC2007, Web dialogs as drop-in components, Slide 8  
Copyright 2007, Callista Enterprise AB



## war-file content - compared to Struts 1

The image compares the directory structures of two WAR files. On the left, 'Struts1\_webapp.war' shows a traditional structure with 'css', 'images', 'META-INF', 'WEB-INF', 'classes', and 'lib' directories. The 'classes' directory contains numerous JSP files, with an orange arrow pointing to them labeled 'loads of classes and view files'. On the right, 'poc-webapp-addressbook.war' shows a more modular structure with 'META-INF', 'WEB-INF', and 'lib' directories. The 'lib' directory contains several JAR files, with a blue circle around them labeled 'modularization support lib'. A yellow box at the bottom right contains the text 'The webapp has become an assembly!'. The Callista logo is in the bottom right corner.

**Struts1\_webapp.war**

- css
- images
- META-INF
- WEB-INF
- classes
  - com
  - lib
    - createednDeliveryNoteDetails.jsp
    - createednDeliveryNoteDetailsContent.jsp
    - createednExportEDNTToFile.jsp
    - createednExportEDNTToFileContent.jsp
    - createednSearchDeliveryNote.jsp
    - createednSearchDeliveryNoteContent.jsp
    - createednSendDeliveryNote.jsp
    - createednSendDeliveryNoteContent.jsp
    - createedrDeviationReportDetails.jsp
    - createedrDeviationReportDetailsContent.jsp
    - createedrExportRDRToFile.jsp
    - createedrExportRDRToFileContent.jsp
    - createedrSearchDeviationReport.jsp
    - createedrSearchDeviationReportContent.jsp
    - createedrSendDeviationReport.jsp
    - createedrSendDeviationReportContent.jsp

**poc-webapp-addressbook.war**

- META-INF
- WEB-INF
- lib
  - poc-comp-addressentry-1.0-SNAPSHOT.jar
  - poc-comp-addresslist-1.0-SNAPSHOT.jar
  - poc-comp-category-1.0-SNAPSHOT.jar
  - poc-service-1.0-SNAPSHOT.jar
  - webcomponent-support-1.0-SNAPSHOT.jar
- web.xml
- index.html

**view-components**

**service component**

**modularization support lib**

**The webapp has become an assembly!**

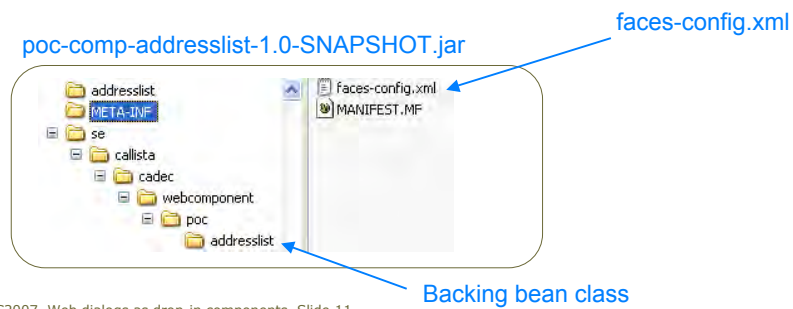
CALLISTA

## The solution - how is it done?

- Utilizing features of enabling technologies
  - JSF (Java Server Faces)
  - Facelets
- Custom code to utilize feature for
  - view resolution
- Custom code for resource resolution
  - stylesheets (CSS), JavaScript-files, images
- Conventions for namespaces

## Enabling technology (1) - JSF

- JSF (Java Server Faces)
  - faces-config.xml (*compare to struts-config.xml*) can be modularized
  - faces-config.xml has a well defined resolution order:  
first search META-INF –dir of JAR's in classpath

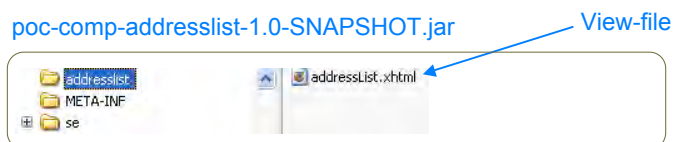


CADEC2007, Web dialogs as drop-in components, Slide 11  
Copyright 2007, Callista Enterprise AB



## Enabling technology (2) - Facelets

- Rendering
  - view-files must be able to exist in jar-file
  - rules out JSP (which must exist in a web-root based path)
- Facelets to the rescue! <https://facelets.dev.java.net/>
  - xhtml-files as views
  - view-files can be resolved from classpath with a custom View-resolver



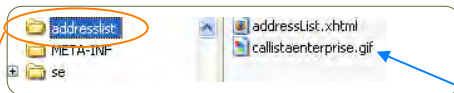
CADEC2007, Web dialogs as drop-in components, Slide 12  
Copyright 2007, Callista Enterprise AB



## Resource resolution & Namespace

- Custom servlet to resolve resources (images, stylesheets) requested by a web-browser for a view (*classpath resolution*)
- Namespace to ensure resource uniqueness

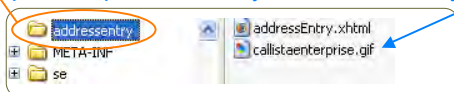
poc-comp-addresslist-1.0-SNAPSHOT.jar



Component namespace

Same filename, would collide without namespace

poc-comp-addressentry-1.0-SNAPSHOT.jar



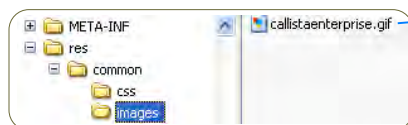
CADEC2007, Web dialogs as drop-in components, Slide 13  
Copyright 2007, Callista Enterprise AB



## Common resources

- Organisation wide resources like
  - logotypes, images
  - fonts and colors (stylesheet)
- Not duplicated into every component
  - common-resources jar

poc-resources-common-1.0-SNAPSHOT.jar



Reference in view-file

```
...  
  
...
```

path-mapping for servlet

CADEC2007, Web dialogs as drop-in components, Slide 14  
Copyright 2007, Callista Enterprise AB



## Component contract

---

- Contract is made up of
  - entry/exit view
  - in/out parameter
  - scope (request, session, ...)
- Navigation and state management
  - use your own strategy with session state (*quickly becomes messy!*)
  - or
  - use a flow engine (like Spring WebFlow)

CADEC2007, Web dialogs as drop-in components, Slide 15  
Copyright 2007, Callista Enterprise AB



## How to partition into components?

---

- During analysis
- Together with interaction design
- Up-front design?

CADEC2007, Web dialogs as drop-in components, Slide 16  
Copyright 2007, Callista Enterprise AB



## Conclusion

---

- Concept works technically
  - but is yet to be proven in a real-world application
- Technology maturity
  - Facelets is a growing community
- Ways to adopt concept
  - full adoption – use a flow engine for navigation and state management!
  - only for modularity (no re-use)

CADEC2007, Web dialogs as drop-in components, Slide 17  
Copyright 2007, Callista Enterprise AB



## Questions

---



CADEC2007, Web dialogs as drop-in components, Slide 18  
Copyright 2007, Callista Enterprise AB

