NeXTPLAN*

Enterprise software that rocks!
with Seaside & Smalltalk

Johan Brichau - johan@inceptive.be
NeXTPLAN is...

- A collaborative resource planning tool for event organizers of all sizes.
- Developed for and with major player of Belgian cultural scene (Vooruit, Gent).
- Features: Event planning, resource planning & management, contact management, retro planning, billing, reporting.
NeXTPLAN*

Monday 13 September 2010
In this talk....

- **Smalltalk** and **Seaside** are an impressive platform to create highly dynamic web applications that blur the traditional distinction with desktop applications.

- Some details:
  - Local AJAX updates
  - OODBs
  - SVG + AJAX in Seaside
  - Testing, porting to Gemstone
Demo 1

- [demo to show component interactions in calendar: grouped events & bubbles, changing names, decoration]
Component-based
Component-based

<table>
<thead>
<tr>
<th></th>
<th>Theaterzaal</th>
<th>Concertzaal</th>
<th>Balzaal</th>
<th>Domzaal</th>
<th>Minard</th>
<th>Majolica</th>
<th>Emiejewrme</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Mo 30-8-2010</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Tu 31-8-2010</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>We 1-9-2010</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Th 2-9-2010</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Fr 3-9-2010</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Sa 4-9-2010</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Su 5-9-2010</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Mo 6-9-2010</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Tu 7-9-2010</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>We 8-9-2010</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Th 9-9-2010</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Monday 13 September 2010
Component-based
Ajax Updates Architecture

- Every operation eventually invokes the update rendering
- Update rendering
  - Component re-rendering
  - Ajax-jQuery replace operation
Global Ajax Update

Monday 13 September 2010
scriptToUpdateBodyOn: html
  ^ (html jQuery: #cells)
  load: [r | self renderCellsOn: r]

+ Easy and crisp
+ Universally applicable

- Re-renders too much (too long!)
- Wastes in-place editing advantage
- Most of the time updates are local
Local updates
Local updates
Local updates
Local updates

Updates of other users (via database changes)!

Monday 13 September 2010
Pitfalls

• Instance variable values
• Capture concurrent changes to calendar
• Client side behavior (Javascript)
  • Invoke again or not?
• For example:
  • Drag & drop of cells
  • Context menu on cells
  • Dynamic calendar extension on page scroll
Seaside Design Pattern?

<table>
<thead>
<tr>
<th>UpdateableComponent</th>
</tr>
</thead>
<tbody>
<tr>
<td>htmlId</td>
</tr>
<tr>
<td>renderContentOn:</td>
</tr>
<tr>
<td>renderInternalOn:</td>
</tr>
<tr>
<td>needsRefresh</td>
</tr>
<tr>
<td>scriptToUpdateOn:</td>
</tr>
</tbody>
</table>

- **Complete render**
- **Internal render**
- **Wants to render?**
- **Script to render**
Local updates

```small
CalendarComponent>>updateScriptOn: canvas
  ^ canvas jQuery ajax script: [:s |
    self cells do: [:c|
      c needsRefresh ifTrue: [
        s << (c updateScriptOn: s)]]]
```

```small
Cell>>updateScriptOn: canvas
  ^(canvas jQuery id: self htmlId)
  load: [:r | self renderInternalOn: r]
```
Local updates

CalendarComponent>>updateScriptOn: canvas
  ^ canvas jQuery ajax script: [:s |
    self children do: [:c|
      c needsRefresh ifTrue: [
        s << (c updateScriptOn: s)]]]

Cell>>updateScriptOn: canvas
  ^(canvas jQuery id: self htmlId)
  load: [:r | self renderInternalOn: r]
Local updates

**CalendarComponent>>updateScriptOn**: canvas

^ canvas [jQuery ajax script: [:s |
  self children do: [:c|
    c needsRefresh ifTrue: [
      s << (c updateScriptOn: s)]]]]

**Cell>>updateScriptOn**: canvas

^ (canvas jQuery id: self htmlId)

  replaceWith: [:r | self renderContentOn: r]
Local updates

- **Own session** changes
  - True object-based change propagation
- **Concurrent session** changes
  - Manual handling
SVG in Seaside

- **SeasideDynamicSVG** package
  - SVG canvas & brushes
  - SVG + jQuery
- **SVG + AJAX**
  - Dynamic load of svg content
  - **Problem**: Painter’s algorithm for "z-index"
  - **Solution**: svgLoad
Demo 2

- [show Tree view animations driven by Ajax updates]
JQuery svgLoad

z-index 1

z-index 2

z-index 3

...
JQuery-SVG

• jQuery load for SVG content
  • Load new content in SVG canvas
  • Add content at end of SVG Canvas
• + add content to existing SVG element
  • Control "z-index" by adding to SVG groups
  • Not yet present on squeaksource / jQuery plugin
<table>
<thead>
<tr>
<th>OODBs</th>
<th>indexes</th>
<th>nested Tx</th>
<th>write barrier</th>
</tr>
</thead>
<tbody>
<tr>
<td>Magma</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>GOODS</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Gemstone</td>
<td>✔️</td>
<td></td>
<td>✔️*</td>
</tr>
<tr>
<td></td>
<td>indexes</td>
<td>nested Tx</td>
<td>write barrier</td>
</tr>
<tr>
<td>------------</td>
<td>---------</td>
<td>-----------</td>
<td>---------------</td>
</tr>
<tr>
<td><strong>Magma</strong></td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td><strong>GOODS</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Gemstone</strong></td>
<td>✔️</td>
<td></td>
<td>✔️*</td>
</tr>
</tbody>
</table>

**Solution: General database access layer**

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>DAO</strong></td>
<td>✔️*</td>
<td>✔️</td>
<td>✔️</td>
</tr>
</tbody>
</table>
GOODS Manual Write Barrier

Smalltalk runtime

Loaded objects

Database

Monday 13 September 2010
GOODS Manual Write Barrier

Smalltalk runtime

Loaded objects

Write buffer

Database
GOODS Manual Write Barrier

Smalltalk runtime

Loaded objects

Write buffer

Database

Monday 13 September 2010
GOODS Manual Write Barrier

- Track changes to database objects
  - Notify on change (e.g. Announcements)
  - Use customized Collection classes
- GOODS uses the write-buffer on commit
Performance (1)

- Magma
- Goods w/ manual barrier
- Goods
- Gemstone

Monday 13 September 2010
Performance (2)

- **Magma**
- **Goods**
- **Goods w/ manual barrier**
- **Gemstone**

Bar chart showing the performance of various systems for '1 commit of 100,000 objects' with times measured in seconds. The chart indicates that Magma takes approximately 0.68 seconds.
Performance (3)

- Magma
- Goods w/ manual barrier
- Goods
- Gemstone

1 commit after one change in 100,000 loaded objects

0.68s
Demo 3

- [demo of resource Finder & inspector to underline database performance issue]
Transactions & Concurrency
Transactions & Concurrency

User 2 changes objects shared in User 1's view
Transactions & Concurrency

User 1 changes same objects

User 2 changes objects shared in User 1's view
Transactions & Concurrency

User 1 changes same objects

User 1 is notified

User 2 changes objects shared in User 1's view
Web-based Concurrency

- Conceptually, this is a transaction conflict!
- But....
Transactions & Concurrency

- Database session scopes
  - Db session per Seaside session [GOODS]
  - Single Db session [Seaside+Gemstone]
- Transaction boundaries
  - Request [Seaside+Gemstone]
  - Conversation
Application is not designed around conversations
GOODS

• Loading/refreshing loaded objects takes a looooong time

• **Refresh DB objects manually** when user will also see the refresh on the screen (i.e. no refresh in an action callback)

• **Concurrent changes automatically detected** as transaction conflicts w.r.t. concurrent application sessions
Transactions & Concurrency

Seaside + Gemstone

Monday 13 September 2010
Gemstone + Seaside

• Loaded objects are always up-to-date when request is handled (refresh is automatic)
• Concurrent modifications only detected as transaction conflicts w.r.t. concurrently handled requests
• Use object change tracking to detect concurrent changes w.r.t. application sessions.
Object Change Tracking

Smalltalk

- Loaded objects
- Write buffer

Database
“X-db” Tx Framework

self session db
  executeAsTransaction: [ ... ]
  forObjects: #( ... )
  onSuccess: [ ... ]
  onFailure: [ ... ]
  retryIf: [ ... ]
Hybrid components

- JavaScript and Smalltalk
- Client-side interactivity
- Seaside's Javascript generation
- Scalability problems:
  - Passenger / callback registry size
  - Client-side performance
Demo 4

- Show in-place editing of Large amount of data: focus in client-Side functionality
Callbacks

assignCallbackScriptOn: html
  ^ (html jQuery ajax callback: [ ... ])
  asFunction
  assignTo: 'someJsVar'

assignCallbackOn: html
  ^ (html urlForAction: [...])
  assignTo: 'someJsVar'
Testing

- Unit testing
- SeasideTesting component testing
  - Added callback simulation
- WebTester testing
testDeleteGroup
  | theCell group theNewCell1 theNewCell2 |
  theCell := self cellAtRow: 3 forLocationWithName: 'Theaterzaal'.
group := theCell menuFocusEvent.

  self executeAsCallback: [
    self component currentMenuFocusEvent: group.
    self component removeSelectedEvents].

  theNewCell1 := self cellAtRow: 3 forLocationWithName: 'Theaterzaal'.
  theNewCell2 := self cellAtRow: 4 forLocationWithName: 'Theaterzaal'.
  self assert: theNewCell1 isEmpty.
  self assert: theNewCell2 isEmpty
Porting to Gemstone

- OODB transaction model
- Grease + Lint rules
- Pharo <-> Gemstone
  - Strict compiler (no empty statements, shadowing,...)
  - Exception -> ExceptionA
  - Blockclosure -> ExecutableBlock
  - Date/Duration/Timespan protocol
  - Assignments to temporaries (GS <3.0)
NeXTPLAN*

Thank you!

Questions?
Want a demo offline?

http://www.inceptive.be