Smalltalk Web frameworks

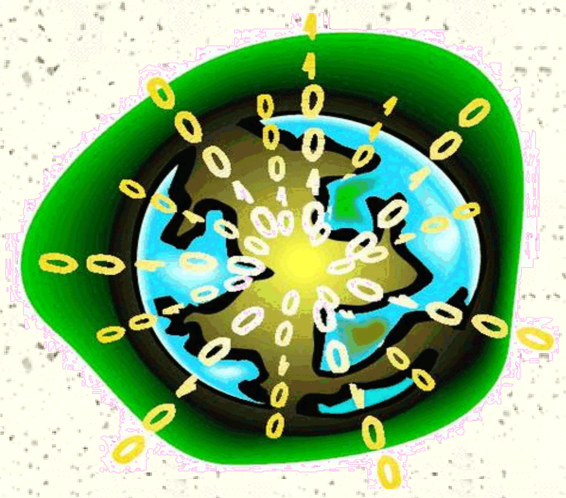
Comparison of Aida/Web and Seaside

Janko Mivšek
janko.mivsek@eranova.si

eranova
Objektne tehnologije

Vsebina

- Definition of Web App server
- Aida’s ideas and architecture
- Comparison to Seaside
- Examples
- Conclusion



eranova Objektne tehnologije

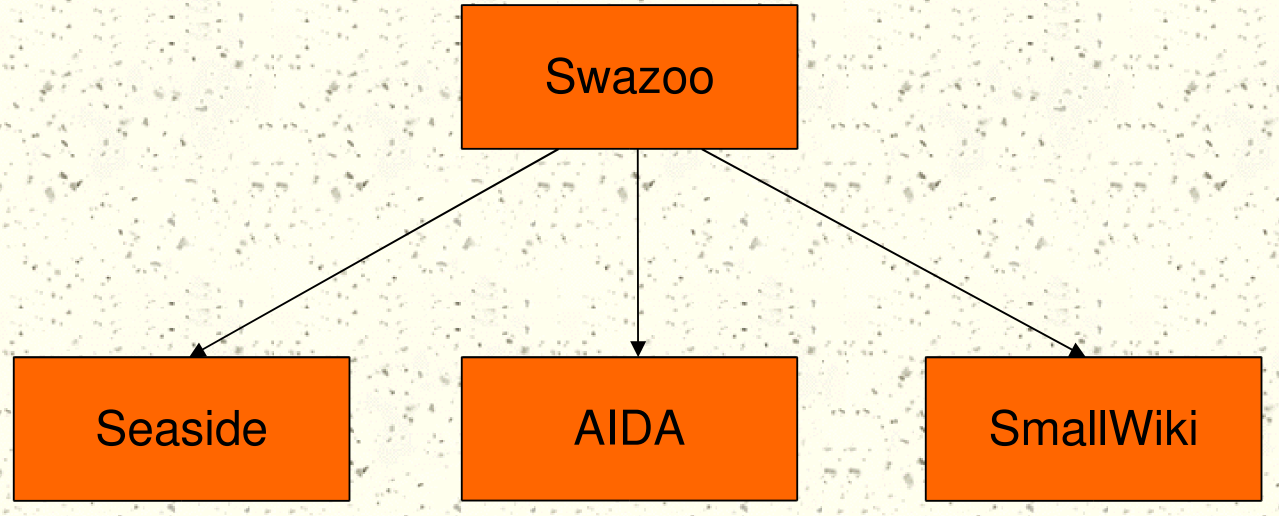
Web application server

- Web + application server
- For web applications
  - Dynamic, on-the-fly built web pages
  - Web pages instead of GUI apps
  - Thin clients instead of fat clients
  - Examples: business apps, portals, ...

Smalltalk powered by

eranova Objektne tehnologije

Swazoo vs. Aida & Seaside



Methods of building web pages

- Embedded tags into HTML
  - jsp, asp, php
  - Cincom Web Toolkit
- Programming
  - Java Servlets
  - Struts
  - AIDA/Web, Seaside

Embedded tags – SSP example

```
<html>
<head><title>Toyz Employees</title></head>
<body>
<table border=1>
<%
    employees := (Toyz new) getEmployees.
    employees do: [:each |
        response write: ‘<tr>’.
        response write: (‘<td>’, each number’, ‘</td>’).
        response write: (‘<td>’, each name, ‘ ‘, each surname, ‘</td>’).
        response write: ‘</tr>’].

%>
</table>
</body>
</html>
```

Enbedded commands

- Pros
  - Simple for web designers
- Cons
  - Functional logic get lost among presentation logic (HTML)
  - “spaghetti” code – hard to maintain and extend

Programming – two examples

Aida:

```
element := WebElement new.
element table width: 500;
    cell color: (self navigatorColor);
    cell colspan: 3; addText: self title header: 3; newRow;
    cell colspan: 3; addRulerSize: 1; newRow;
    cell addText: 'uptime: ';
    newCell align: #right; addText: (self printSeconds: self session site uptime);
    newRow.
^self pageFrameWith: element title: self title .
```

Seaside:

```
renderContentOn: html
html form: [
    html table: [
          html tableRowWithLabel: 'Username‘ column: username.
          html tableRowWithLabel: 'Password'
            column: [html passwordInputWithCallback: [:v | password := v]].
          html tableRowWithLabel: 'Confirm Password'
            column: [html passwordInputWithCallback: [:v | confirmation := v]].
          html spacerRow.
          html tableRowWith: [
            html submitButtonOn: #changePassword of: self. html space.
            html submitButtonOn: #cancel of: self
            ] span: 2] ]
```

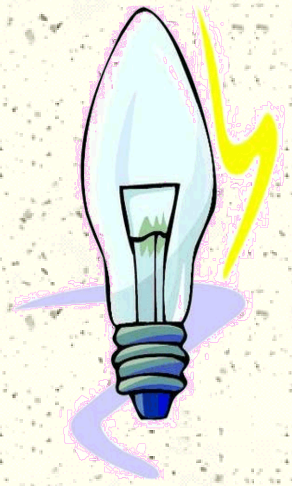
Objektne tehnologije

Programming

- Pros
  - No HTML, just one programming language
  - Separation of presentation from model
  - Comonents, reusability
- Cons
  - Difficult for web designers
  - Difficult to move design into code
- Solution: CSS !

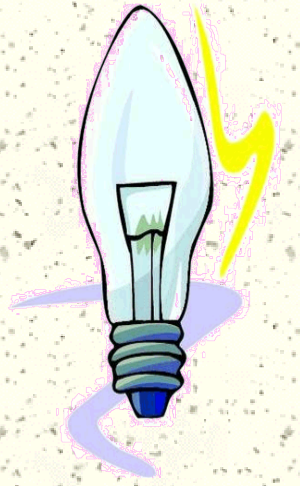
Aida's basic ideas

- Object system = web of objects
  ... connected with object references
- World Wide Web = web of pages
  ... connected with Url links



Aida’s basic ideas

- Every object should present itself as web page
- Object references should map to Url links and vice-versa – automatically!



eranova Objektne tehnologije

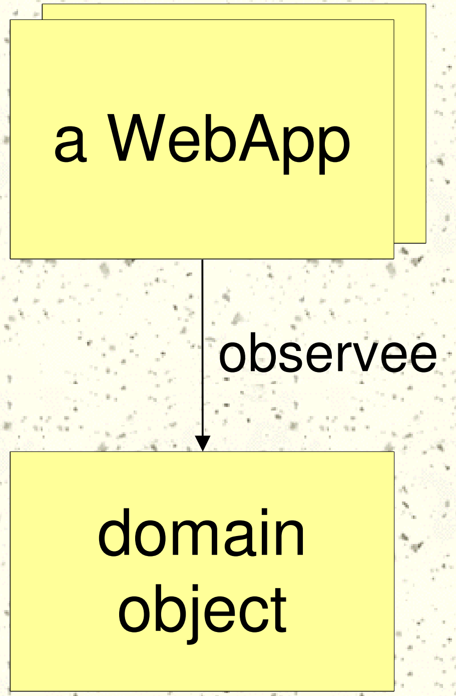
REST arhitectural principle

- **RE**presentational **S**tate **T**ransfer
- **Web** comprised of **resources**, addressed with **Uniform Resource Locators** (Url).
  - Example: http://www.eranova.si/aida
- As answer to request: **presentation** of resource
- Only ***GET, PUT, POST*** etc

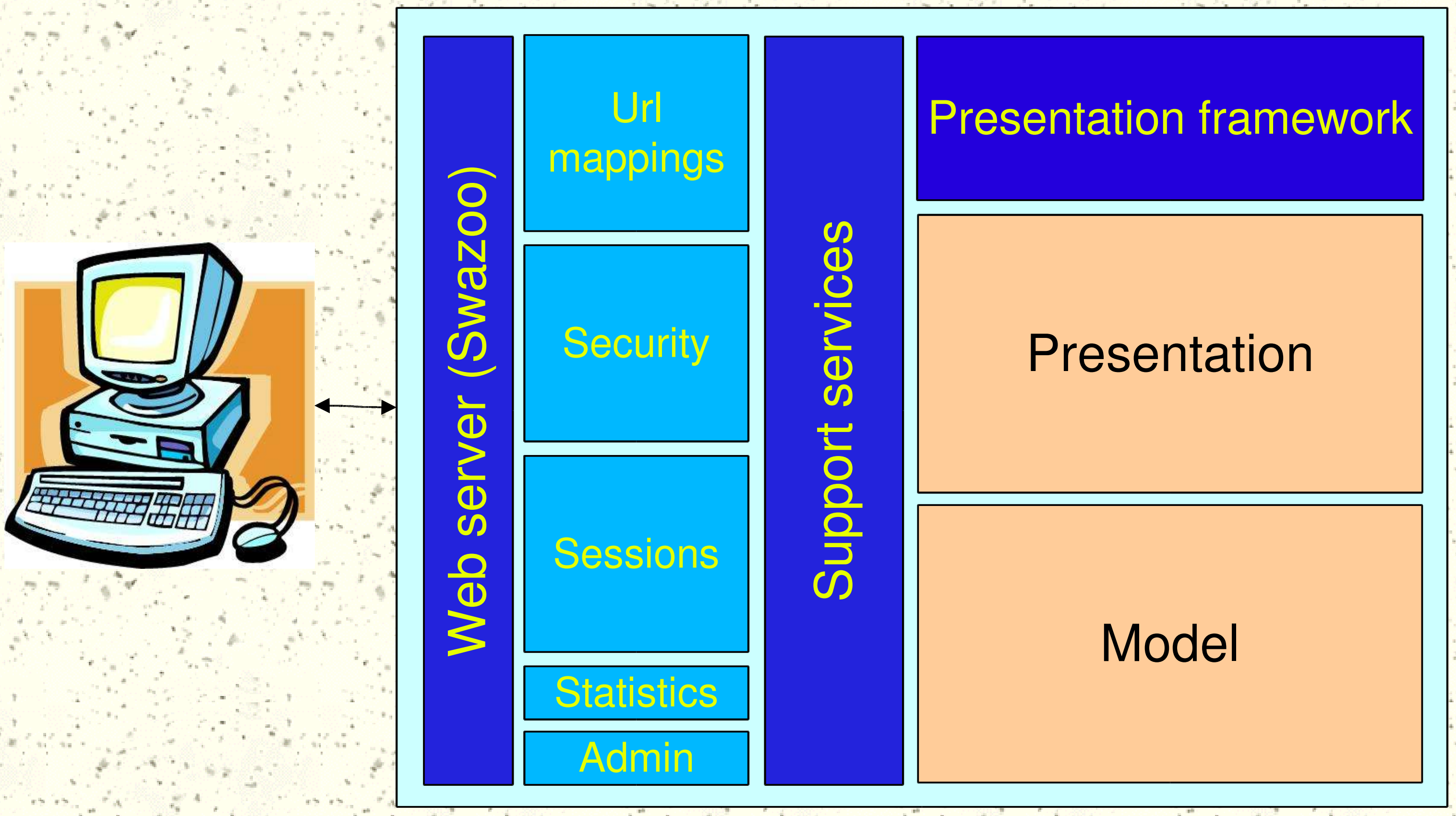
- ***W3C Architecture of the World Wide Web***
  - http://www.w3.org/TR/2004/WD-webarch-20040705/
- Building Web Services the REST Way
  - http://www.xfront.com/REST-Web-Services.html

MVC on Web

- **M**odel-**V**iew-**C**ontroler
- Strict separation of presentation from domain logic
- Every domain object has its presentation counterpart

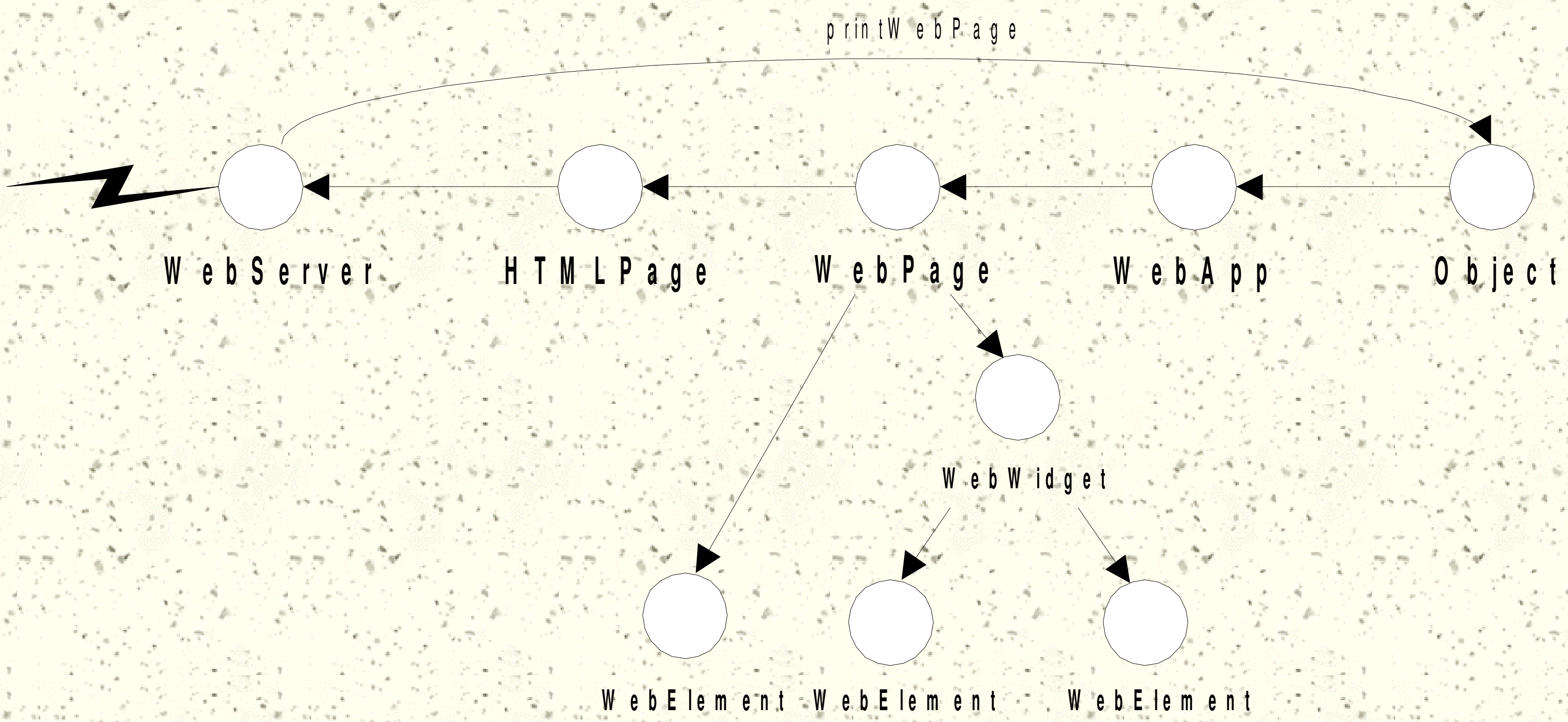


Aida web application



eranova
Objektne tehnologije

Web request path



Seaside concepts

- Continuations
- Control flow easily defined
- Session like process/thread
- Components: UI state and logic
- Subroutine-like calls of another components with results returned

Example – counter with Urls

Seaside:

```
WACounter renderContentOn: html
html heading: count.
html anchorWithAction: [self increase] text: '++'.
html space.
html anchorWithAction: [self decrease] text: '–'.
```

Aida:

```
CounterApp viewMain
self
    addTextH1: self observee count printString;
    addLinkTo: self observee text: '++' view: #increase; addSpace;
    addLinkTo: self observee text: '--' view: #decrease.

CounterApp viewIncrease
self observee increase.
self viewMain

CounterApp viewDecrease
self observee decrease.
self viewMain
```

eranova Objektne tehnologije

Example – counter with buttons

Seaside:

```
WACounter renderContentOn: html
html form:  [
    html heading: count.
    html submitButtonWithAction: [self increment] text: '++'.
    html space.
    html submitButtonWithAction: [self decrease] text: '--'].
```

Aida:

```
CounterApp viewMain
self
    addTextH1: self observee count printString;
    addButtonText: '++' action: #increase; addSpace;
    addButtonText: '--' action: #decrease.

CounterApp actionMainIncrease
self observee increase

CounterApp actionMainDecrease
self observee decrease
```

eranova
Objektne tehnologije

Example – counter with dialog

Seaside:

```
WACounter decrease
count = 0
  ifFalse: [count := count - 1]
  ifTrue:
    [(self confirm: 'Do you want to go negative?')
       ifTrue: [self inform: 'Ok, let''s go negative!'. count := -100]].
```

Seaside programming

- Pros
  - Flow control in one place
  - Easy “subroutine pages” with results returned
  - Back button supported (really needed?)
- Cons
  - Not much OO, more procedural programming
  - Url’s not REST-like
  - Cross-linking pages difficult

eranova Objektne tehnologije

Aida programming

- Pros
  - Every domain object has persistent Url
  - Automatic cross-linking of pages
  - REST principle fully supported
- Cons
  - GOTO like programming for more complex apps
  - control flow not shown in one place
  - Back button not fully supported

Conclusion

- Aida/web
  - http://www.eranova.si/aida
  - janko.mivsek@eranova.si
- Seaside
  - http://www.beta4.com/seaside2/
  - Avi Briant avi@beta4.com

eranova
Objektne tehnologije