# GT2003Stefano

Talk by Stefano Mazzocchi at the 2003 GetTogether: How I see Cocoon - a visual journey.

Raw notes taken collaboratively during the talk.

# Stefano takes the stage

Says he could not come last year because of ApacheCon.

Shows an unusual presentation, in the very good sense, very graphical, mostly photos and short comments.

A written report will not do justice to this talk. You had to be there.

### What is Cocoon:

- Cocoon is designed for People, not machines - empower all the different kinds of people who work on projects
- Guided by Open Standards (but not trapped by them)
- Free to make mistakes, in public
- Creative & original, not afraid to copy good stuff
- Looks to the past for inspiration or warnings
    - Like GOTO was found harmful in the past, finite state machines for web apps might be the next thing that we find harmful...
- No fear of innovation
- Balance, between innovation & compatibility
    - It will not collapse beneath your feet, because we need it to work too
- Modular, adaptable, etc. yet solid
- Care for details
    - It's ok to take six months to polish the sitemap contracts
- Self-organising organism

### Concepts

- "Creativity is making the complicated simple" - Charles Mingus
- Separation of concerns - simple yet very powerful concept
    - Seperating overlaps in responsibilities
    - Allow people to do what they do best and leave the rest to others
    - Pyramid of contracts
    - Concern Islands, kept seperate
        - Management
        - Logic
        - Content
        - Style

### Design

- "To create, one must first question everything" - Eileen Grey
- Architecture of Cocoon (shows diagram)
    - Many different web clients (browser, PDA, cell phone, TV, agent, etc)
    - Try doing this in JSP
    - Back-end: web services, data sources, content repositories
    - Two interfaces:
        - Command Line
        - Servlet

(Stefano shows another diagram)

- Cache is essential to performance
- Pipeline components can allow you to create a complete site without writing a single line of Java code

- Blocks (Another diagram)
    - Cocoon 2.1 without Blocks is limited, we have distribution problems with it as everything possible has to be contained.
    - (Real) Blocks will be components at the application level
    - User connects to the Block Deployer (command-line, GUI, web-based tool...time will tell) to select the required Blocks for a given application
    - Block deployer can automatically download required (dependant) blocks at runtime
    - Cocoon keeps seperation between blocks that are not dependent (allows easy versioning and hot-deploy scenarios)
    - Possible scenario: install a complete web site as a Block
    - Rely on dependencies between Blocks
- Blocks wiring (Diagram)
    - Connect website paths to specific blocks
    - Connect 'Behaviours' (Interfaces) automatically, hot deployable, polymorphically
    - Extension mechanism, so you can overlaod a Block from another one (whether this is logic, content or style)
- Blocks will become available in Cocoon 2.2
- The design is completed

- Please start planning to use them

## Flow (Another diagram)

- Continuations are not new, they have been use on the web as early as 1995 in Lisp and Scheme (Paul Graham)
- A logic engine, connected to the SiteMap
- Based on JavaScript (current implementation)
- Took the ideas of Continuations from Scheme
    - Pauses the flow script while the user fills in their data
    - Not something that has normally been possible for web programming environments
    - (Discussion on why FSM is not good)
    - Very fast prototyping, increases productivity
    - Pipeline Teeing with Flowscript (Another diagram)
        - processPipelineTo ()
        - Your FlowSript can use a Pipeline to output to a Source (Disc etc.)

Stefano gives a very clear explanation of Flowscript with Continuations - looks like today might be the Day of the Web Application for many Cocooners 😉

Diagram 6 shows how the results of a pipeline can be saved as well as being sent (maybe after further processing) to the client.

## Sitemap Components

Now we're into 'normal' stuff, explaining generators, transformers, serializers etc...

New for 2.2 will be "pipeline snippets", assembled into "virtual components"

- Several components put together, that can be reused elsewhere in your sitemap, as if it were a single component.
- Virtual Generator: Generator followed by optional Transformers
- Virtual Transformer: One or several Transformers
- Virtual Serializer: Optional Transformers followed by a Serializer

Goes on to describe the current components: Readers, Selectors.

Map:call: call resources, (javascript) functions or continuations.

Redirect to a different URL. Mount a different sitemap (isolating parts of a project, or teams etc.)

## Sitemap notation

"Example sitemap" page shows an overview Stefano's notation for the sitemap - it would be good for Cocoon documentation or articles writers to standardize on a notation, this one looks good!

Brings talk to an end, with a famous photo of himself as the "evil fairy"

## Audience questions:

- Someone talks about MIT's haystack project (http://haystack.lcs.mit.edu/), which is supposed to become the killer app of the web (smiles all around). Did we look at this for inspiration?
    - Not really says Stefano, but we're always looking at similar projects for inspiration.

(Gianugo on chat) "a huge bloated desktop replacement but with a few nice concepts"