ADVANCED PROGRAMING

AMIT CARMON

AMITCARMON10@GMAIL.COM

2025

BACKGROUND & GOALS

Project Context:

- Publish–subscribe computational agents
- Each agent = specialized operation (add / mul / inc / etc.)
- Topics = named data channels

Goal:

- Complete View layer (upload config + interact)
- Real-time visibility & manipulation in browser

Architecture Overview

System Architecture:

- Core: TopicManager (Singleton, pub/sub hub)
- Agents: Reactive + stateless/derived logic
- HTTP Server: Custom (Sockets, threads, minimal parser)
- Servlet Layer: Clean separation (Controller role)
- View: Static templates + dynamic HTML/SVG generation

Design Highlights:

- Layered design (Model–Controller–View)
- Demonstrates low-level protocol handling

Visual: Layered stack diagram: Agents/Topics → TopicManager → Servlets → Browser (frames)

KEY COMPONENTS

1

TopicManager

- Routing + last-value cache
- Singleton pattern implementation
- Thread-safe topic management

2

Reactive Agents

- PlusAgent, MulAgent, IncAgent, DivAgent, SubAgent
- Reactive recalculation on input change
- State management for multi-input operations

3

Message & Topics

- Observer backbone pattern
- Type-safe message handling
- Pub/Sub architecture

4

Configuration System

- Config → GenericConfig → Graph (nodes & edges)
- Graph introspection utilities (structure extraction)

HTTP & VIEW LAYER

WEB INTERFACE ARCHITECTURE

MULTI-FRAME LAYOUT

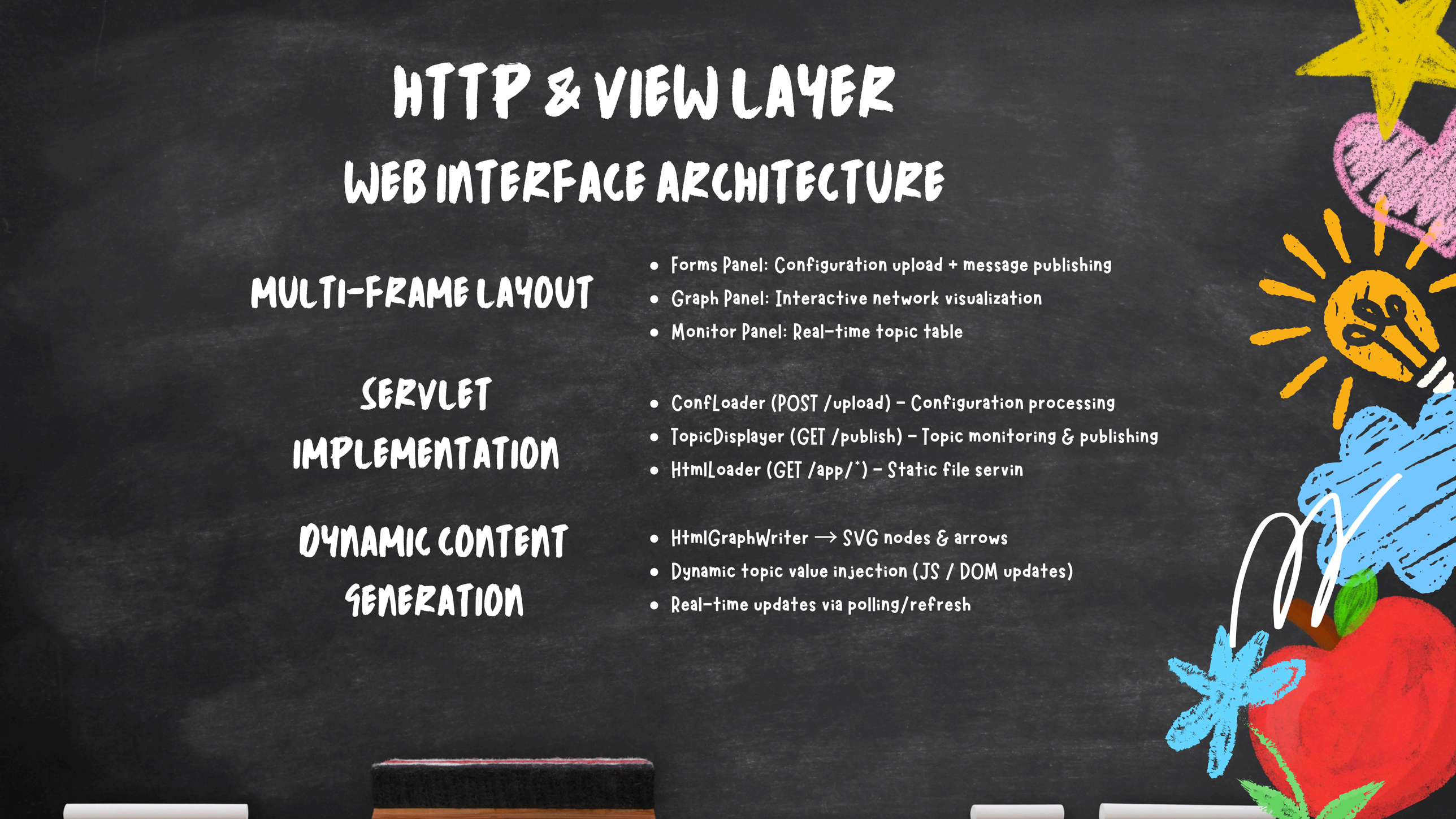
- Forms Panel: Configuration upload + message publishing
- Graph Panel: Interactive network visualization
- Monitor Panel: Real-time topic table

SERVLET IMPLEMENTATION

- ConfLoader (POST /upload) – Configuration processing
- TopicDisplayer (GET /publish) – Topic monitoring & publishing
- HtmlLoader (GET /app/*) – Static file servin

DYNAMIC CONTENT GENERATION

- HtmlGraphWriter → SVG nodes & arrows
- Dynamic topic value injection (JS / DOM updates)
- Real-time updates via polling/refresh



EXTRA FEATURES

1. Cycle Detection
    - DFS-based guard against infinite loops
    - Graph validation during configuration load
2. Multi-Format Configuration
    - Text + JSON configuration support
    - Robust parsing and validation
3. Reflection-Based Agent Instantiation
    - Dynamic class loading from configuration
    - Extensible agent system
4. Interactive Graph Layout
    - Force/spring layout algorithm
    - Collision reduction and auto-positioning
    - Live value overlays on graph nodes
5. Comprehensive Logging
    - Deploy, publish, compute trace
    - Debug and monitoring capabilities

ENGINEERING PRINCIPLES & LEARNINGS

1

SOLID PRINCIPLES APPLIED:

- SINGLE RESPONSIBILITY: EACH SERVLET HANDLES ONE CONCERN
- OPEN/CLOSED: EXTENSIBLE AGENT SYSTEM VIA REFLECTION
- DEPENDENCY INVERSION: INTERFACE-BASED AGENT DESIGN

2

TECHNICAL INSIGHTS:

- REACTIVE DESIGN > BATCH FOR UX & CORRECTNESS
- TEMPLATES + GENERATION = FAST ITERATION
- PUB/SUB DECOUPLING ENABLES SCALABILITY
- DEEPER APPRECIATION FOR FRAMEWORKS' HIDDEN COMPLEXITY

3

FUTURE ENHANCEMENTS:

- PERSISTENCE LAYER FOR CONFIGURATION STORAGE
- AUTHENTICATION & AUTHORIZATION
- RICHER ANALYTICS AND PERFORMANCE METRICS
- WEBSOCKET SUPPORT FOR TRUE REAL-TIME UPDATES


THANK YOU
AMIT CARMON