SOEN 387

Web-based Enterprise Application Design

Credit to Dr. Chalin

- These notes are based on his originals

Good Design

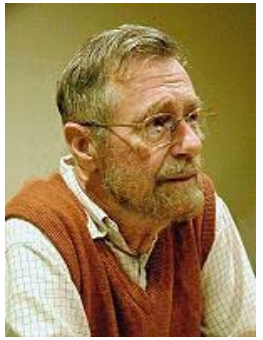
- It should be simple
  - Less it more
- It should show intent
- It should meet user requirements
  - Testing?
- It should be easily maintainable

Fundamental S/W Design Principles

The following are very interrelated:

- Information Hiding (Parnas)
- Protected Variations (Larman)
- **Separation of Concerns** (Dijkstra?, Parnas)
  - High cohesion.
  - Low coupling.

We will apply these principles often.



OO Design Principles

- Design based on *Responsibility Assignment*.
  - Assigned to types, classes or objects.
- Approaches:
  - GRASP – by Larman.
  - Responsibility-Driven Design (RDD) – by Wirfs-Brock et. al.

OO Design Principles: GRASP (Larman, SOEN 343)

- Information Expert.
- Creator.
- High Cohesion.
- Low Coupling.
- Controller.

Design Principles: Expectations

Fundamental (general) principles:

- Consistently applied throughout the course.
- I will point out when, and you should be able to identify these situations too.

OO:

- Will not be explicitly taught, but rather applied.
- Review Larman, or other sources.

Architectural Styles

- Which do you know?
- Have you heard of any?

EA Architectural Styles

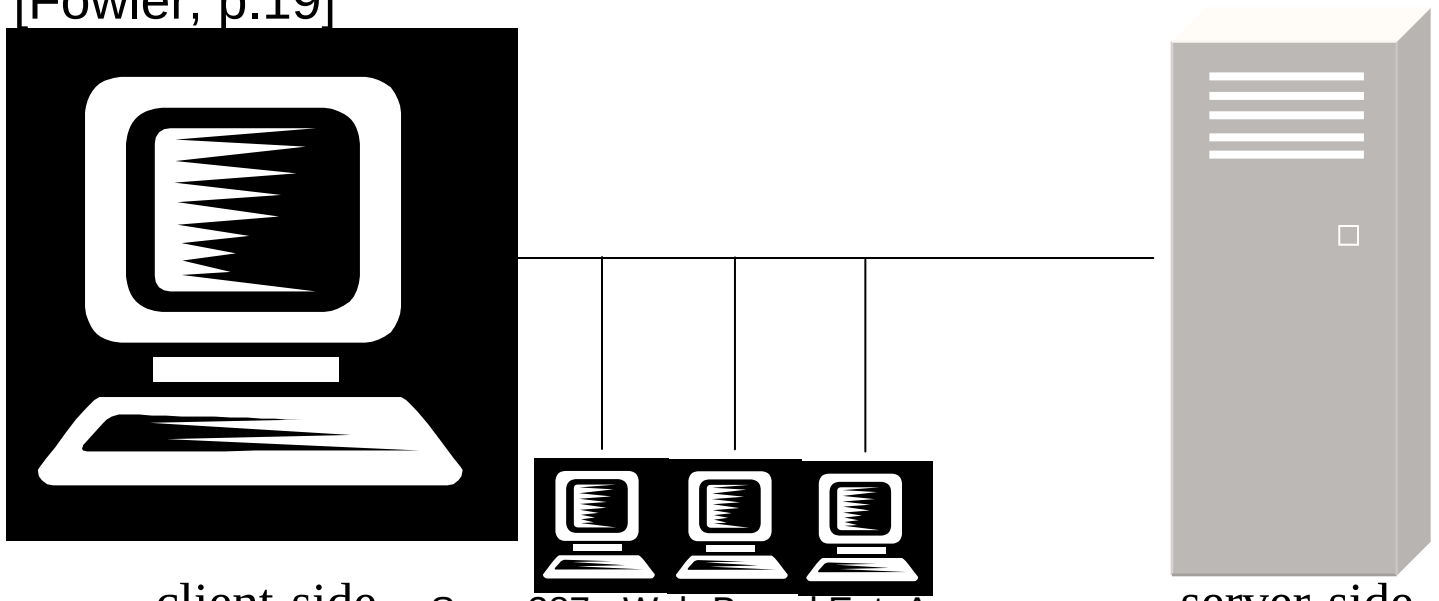
- Are there any styles that are applicable in the context of WEA?

  → Client-server.

  → Layered …

Client-Server (Two-tiered System)

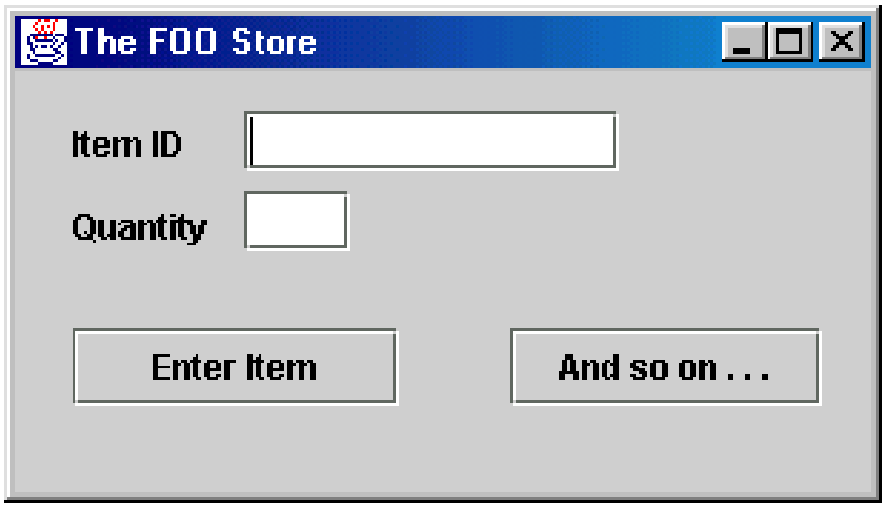
- "… most people see *tier* as implying a physical separation. Client-server systems are often described as two-tier systems …" [Fowler, p.19]



client-side software

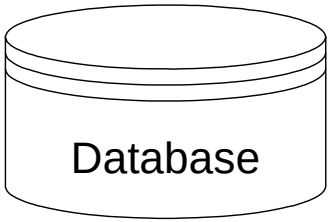
server-side soft-ware

Enterprise Application Layers



Calculate taxes

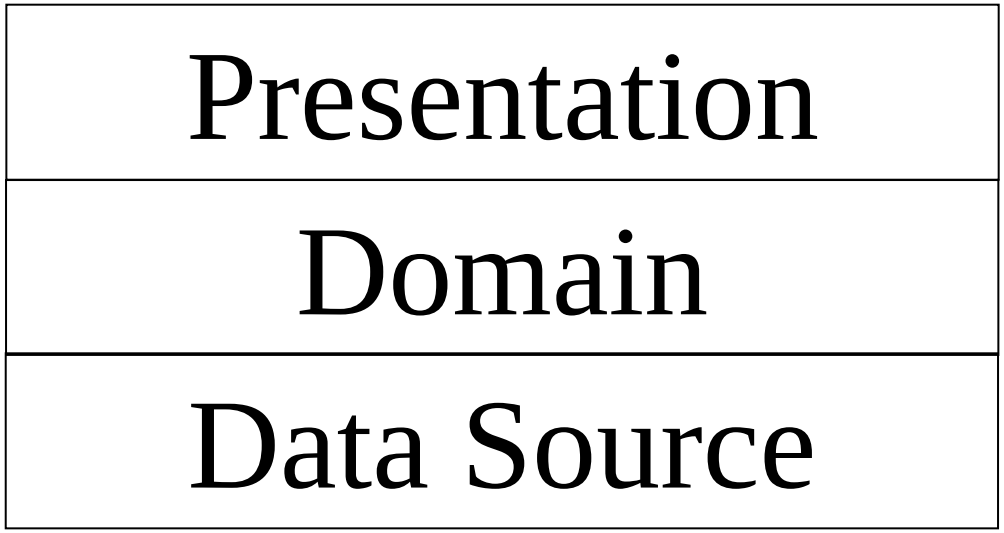
Authorize payments



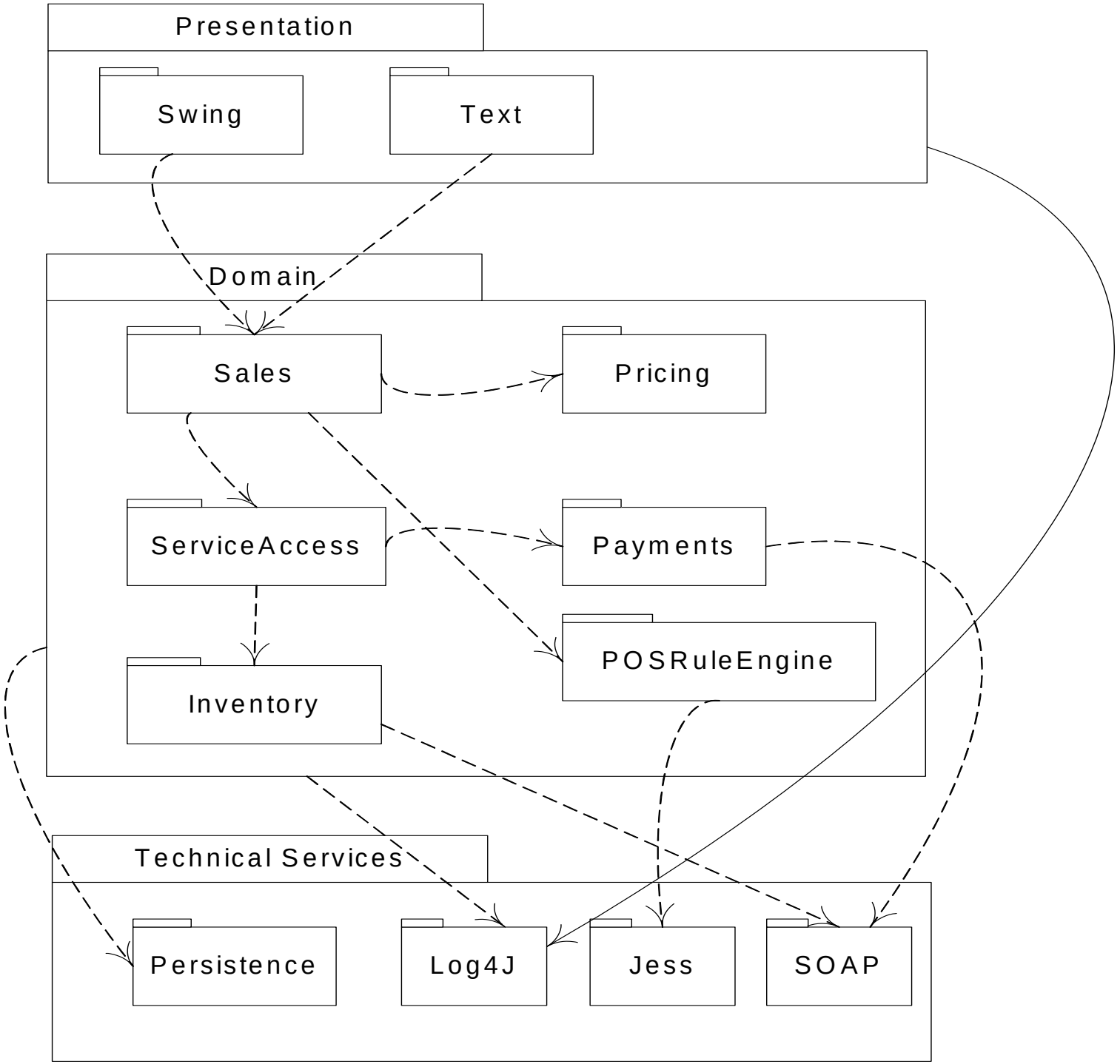
Layering – General Scheme

- Presentation / Application.
  - UI.
  - Generally “thin”.
  - (Term “application” can be misleading. It does not mean …)
- Domain / Business Logic.
  - Core system functionality.
- Technical Services.

Functionality / Dependency



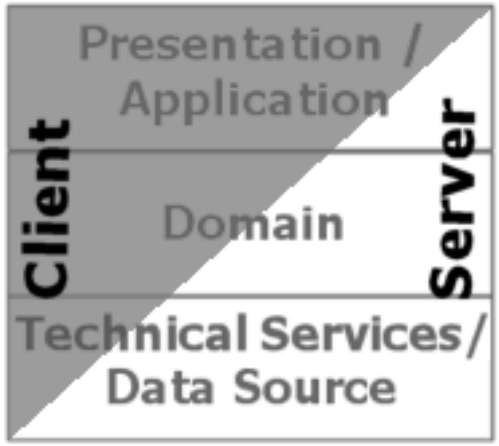
Layer Dependencies Example



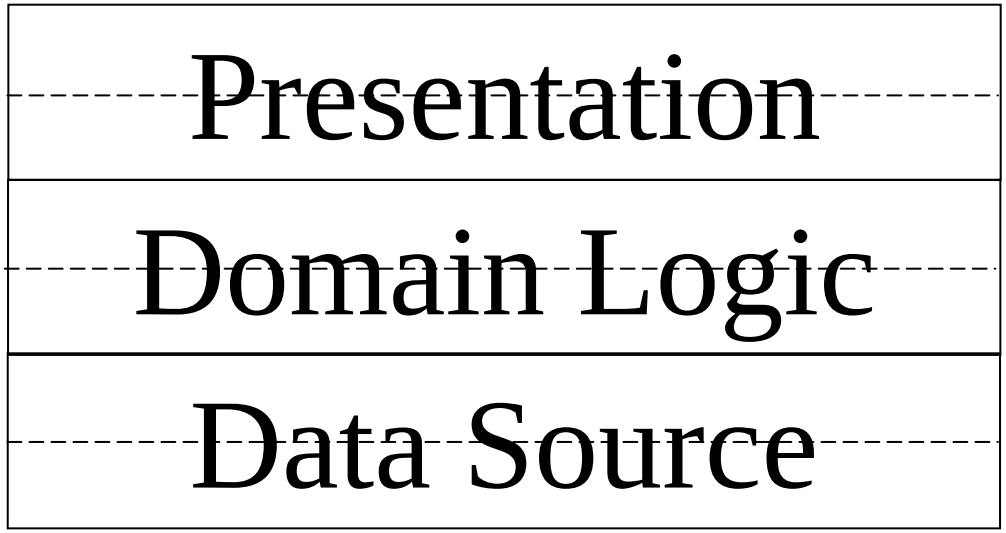
Layers

- What are pure layers?
- Where do we run our layers?

Where to Run Layers



Layers Refined



- GUI windows
- reports
- speech interface
- HTML, XML, XSLT, JSP, Javascript, ...

Presentation
(AKA Interface, UI, View)

- handles presentation layer requests
- workflow
- session state
- window/page transitions
- consolidation/transformation of disparate data for presentation

Application
(AKA Workflow, Process, Mediation, App Controller)

- handles application layer requests
- implementation of domain rules
- domain services (*POS*, *Inventory*)
  - services may be used by just one application, but there is also the possibility of multi-application services

Domain(s)
(AKA Business, Business Services, Model)

- very general low-level business services used in many business domains
- *CurrencyConverter*

Business Infrastructure
(AKA Low-level Business Services)

- (relatively) high-level technical services and frameworks
- *Persistence*, *Security*

Technical Services
(AKA Technical Infrastructure, High-level Technical Services)

- low-level technical services, utilities, and frameworks
- *data structures, threads, math, file, DB, and network I/O*

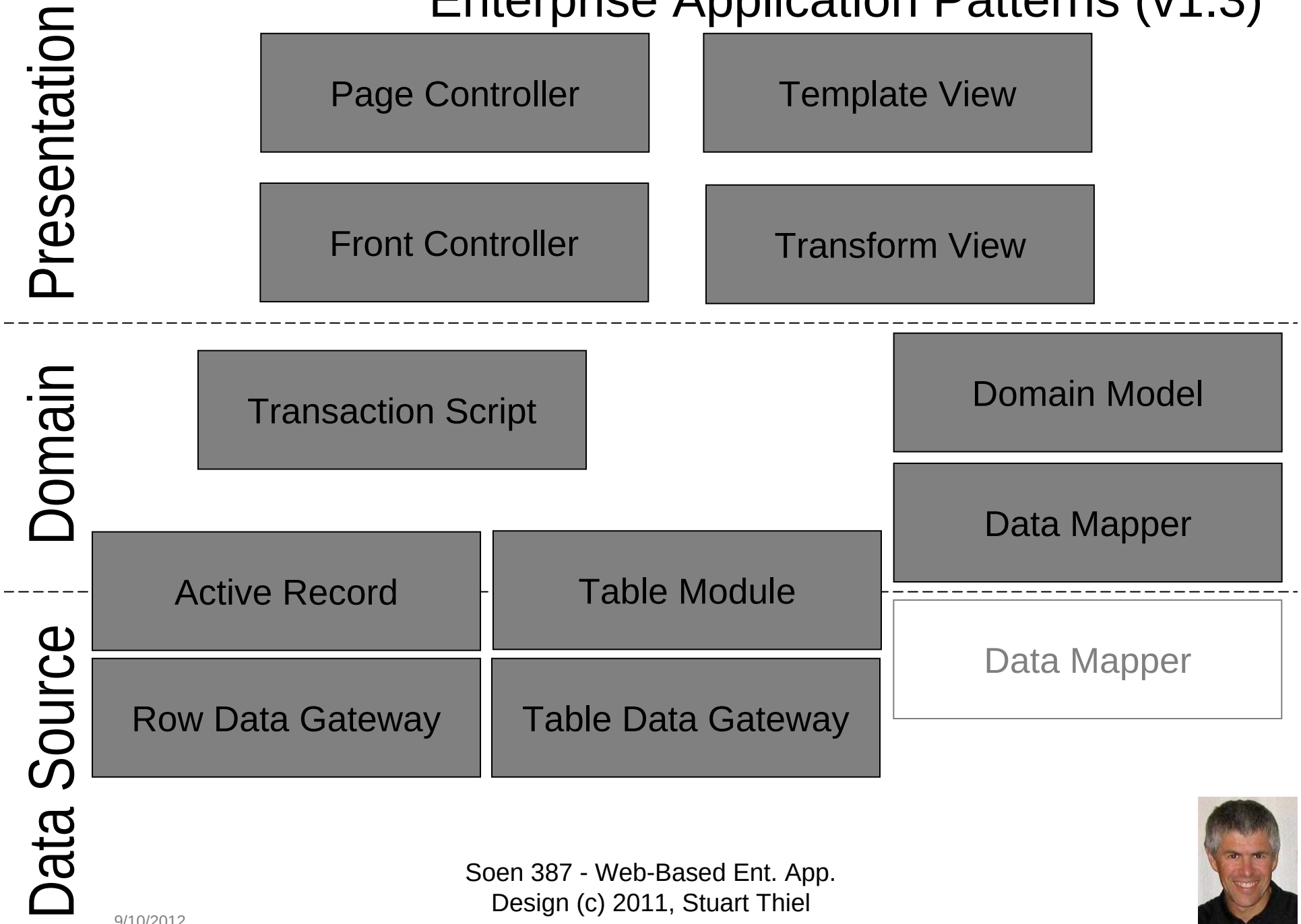
Foundation
(AKA Core Services, Base Services, Low-level Technical Services/Infrastructure)

- Ref.: Larman

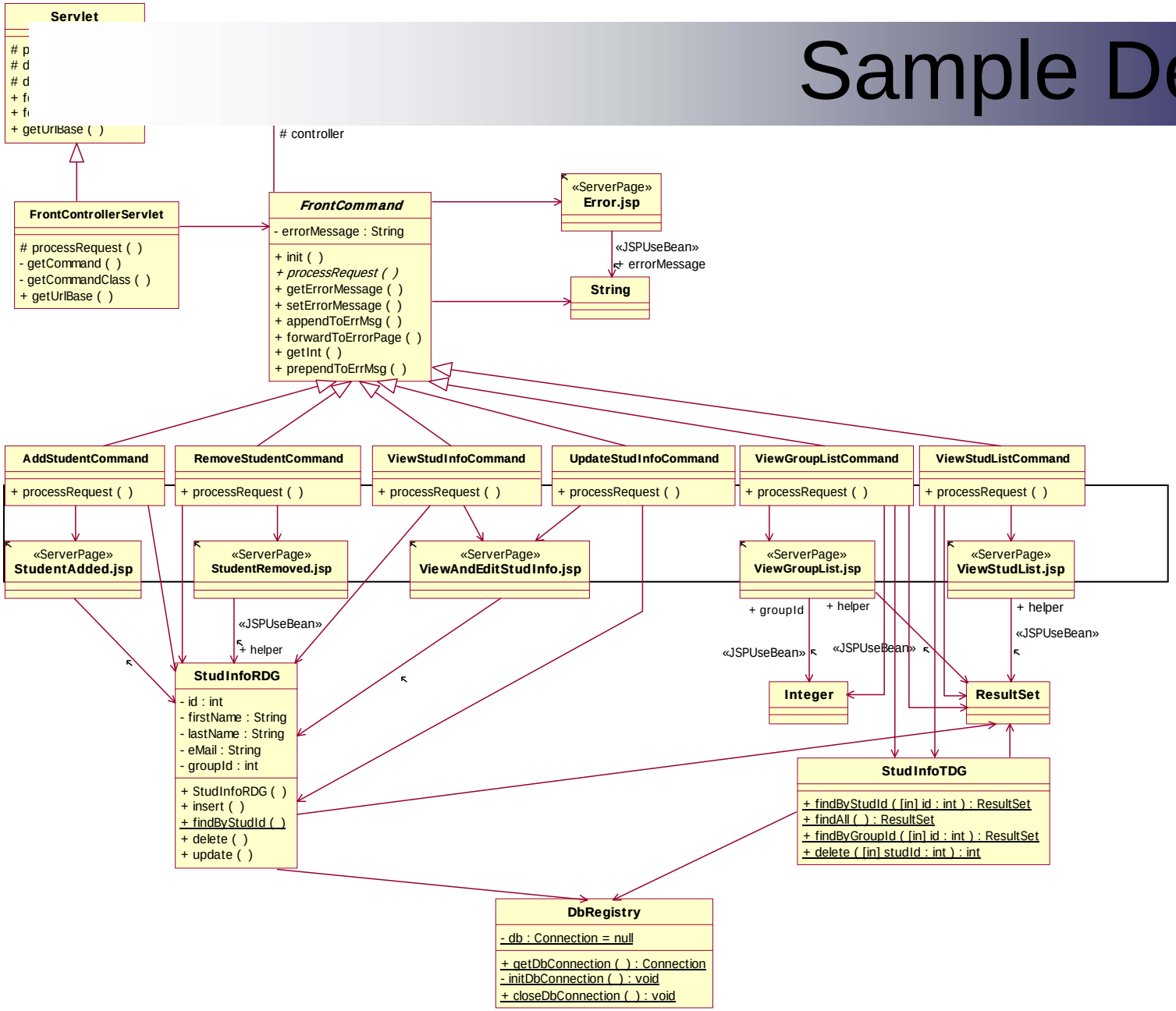
Presentation in EAs: Larman

- JSP pages
- I like this distinction
- What about taglibs or template languages?
- Layers within layers?

Enterprise Application Patterns (v1.3)



Sample Design

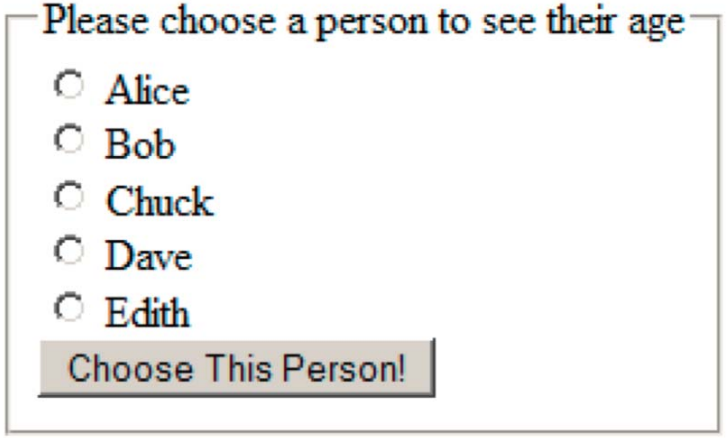


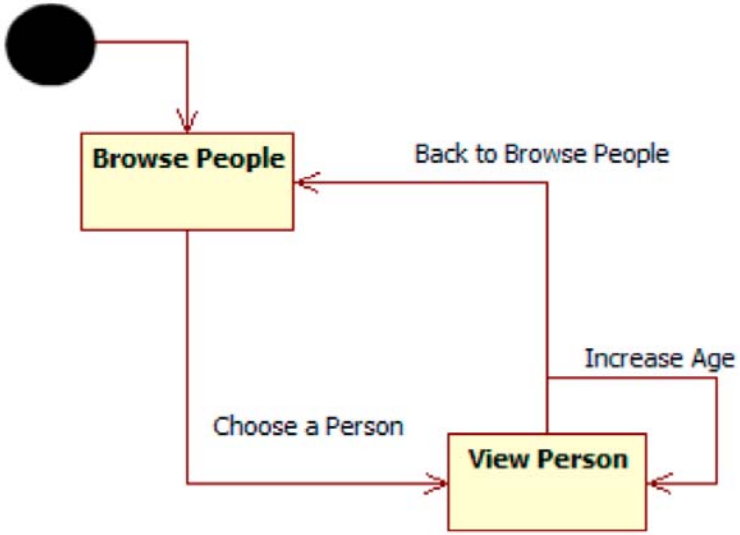
Course Examples

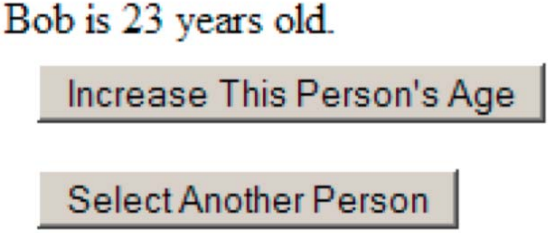
- We'll revisit some of the same examples a lot
  - Buddy Age
  - Hello Web

Buddy-Age Application: Features

- Browse buddy list.
- View person (age).
- Increase age.







Hello Web / Greeting Application

- Will also be used in first few lectures as a running example.
- First version seen during tutorial.

What happens when you don't use the patterns?

- You end up with a …

Do-it-all Servlet

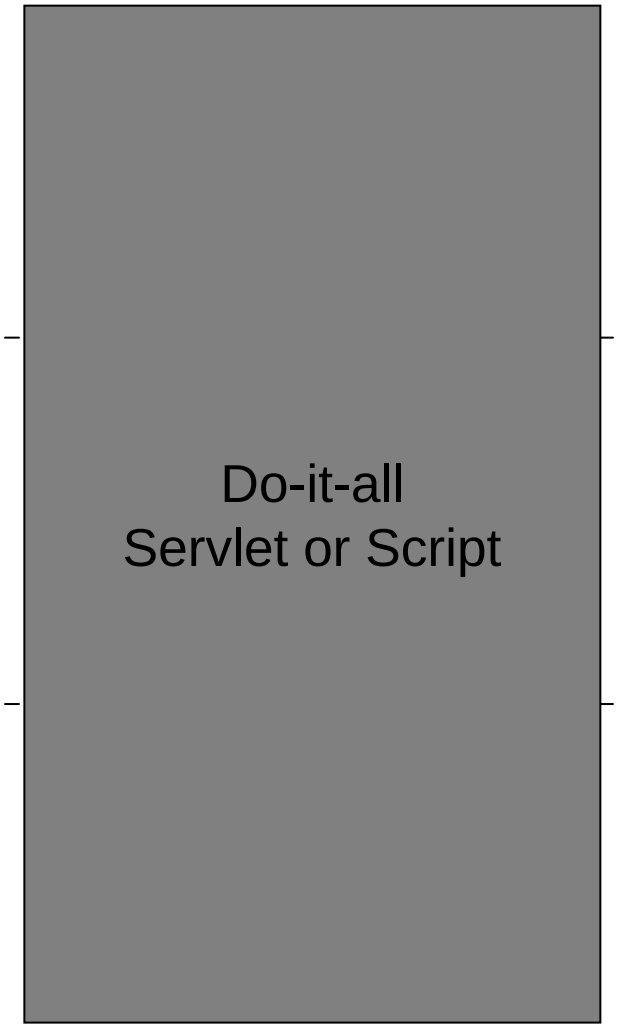
- … or, a Degenerate Transaction Script (TS).
- Responsibilities covers all three layers.
- For a Degenerate TS: roughly, each corresponds to a Use Case.

Do-it-all Servlet or Script

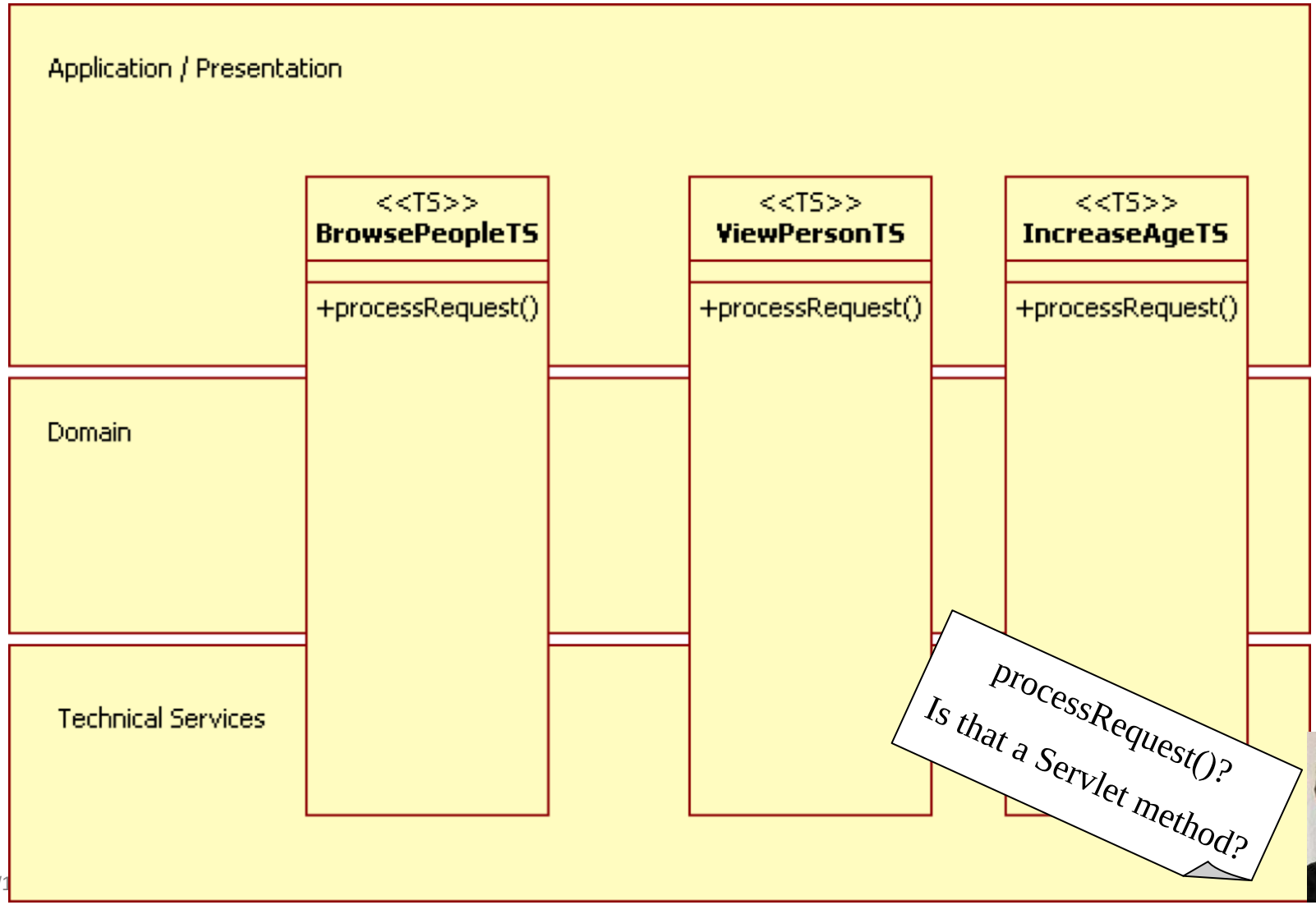
Presentation

Domain

Data Source



Do-it-all (Degenerate) Transaction Script (TS) for Buddy App



How do we feel about this

- How does it scale?

Hello.jsp

```
<%@ page
  contentType="text/html;
  charset=UTF-8" language="java" ...
  %>
<html>
<body>
Hello <%= request.getParameter("name") %>
</body>
</html>
```

Hello.jsp

```
<%@ page
  contentType="text/html;
  charset=UTF-8" language="java"
  ... %>
<html>
<body>
Hello ${param[name]}
</body>
</html>
```

Container processes JSPs …

- Web server (Tomcat) actually compiles JSP pages into Servlet like classes.
- Hence, JSPs are a convenience for developers.

Tomcat: At Most One Servlet Instance

- For any given Servlet class, S, there is at most one instance of S.
- This has important implications w.r.t. concurrent access to the services of S.
- (More on this later.)

Parameters and Attributes

- Which are which?
- What are contexts?
- One Servlet?