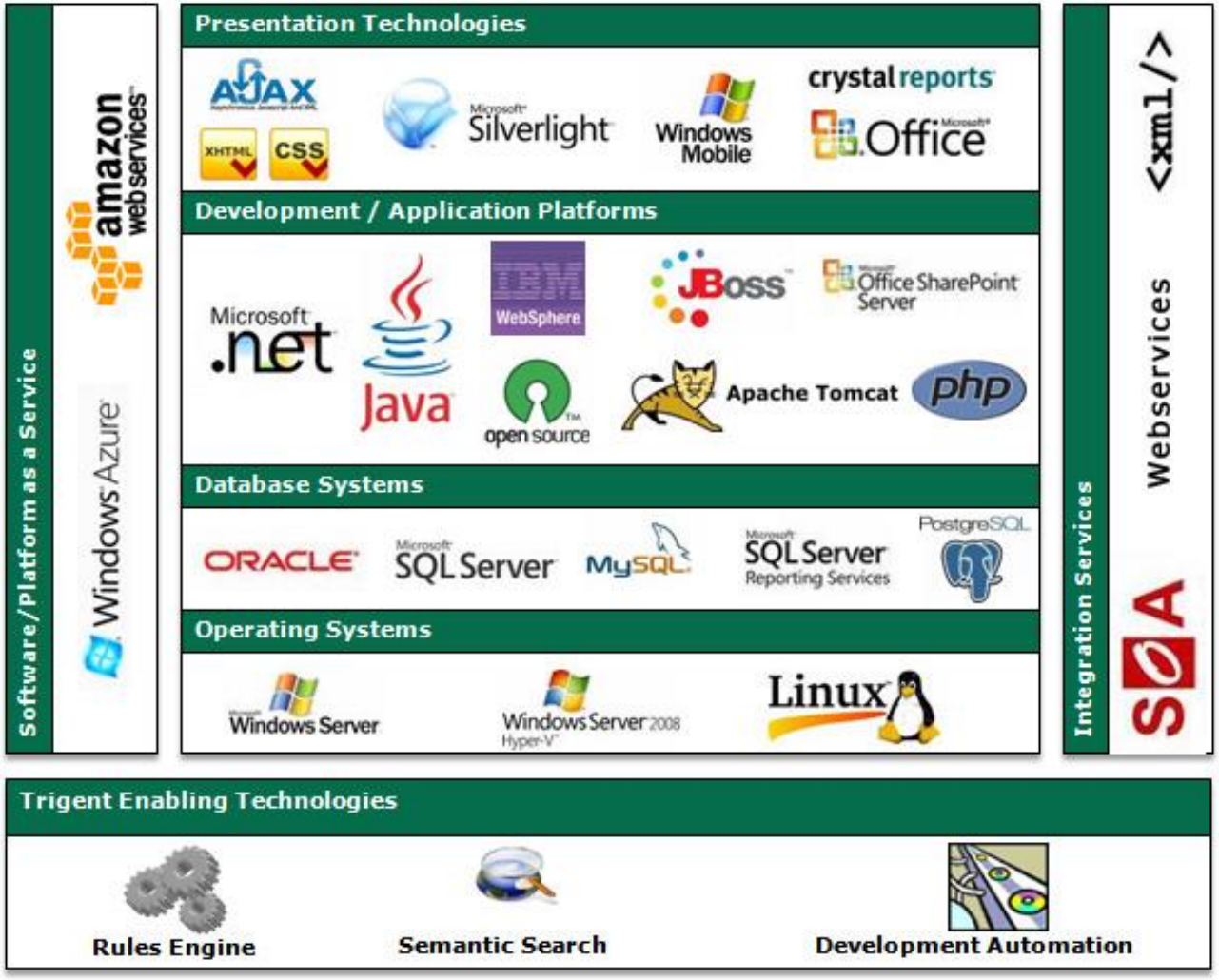
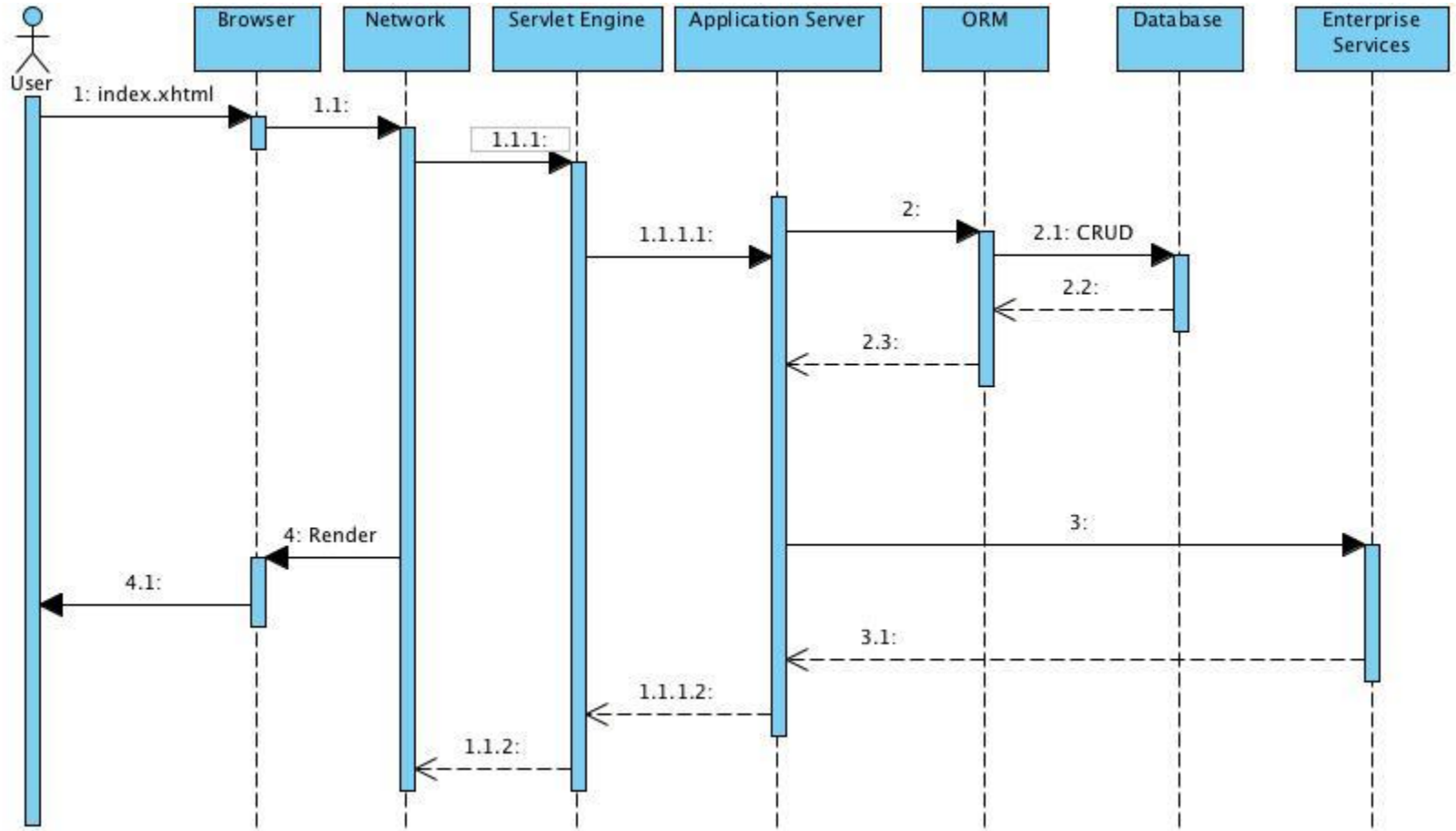
Julius Bär

HOME MADE PERFORMANCE PROBLEMS CAUSED BY FRAMEWORK USAGE

TECHNOLOGY STACK



TYPICAL WEB 2.0 APPLICATION



POTENTIAL PROBLEM FIELDS

Requirements

Only Functional and UI are defined, no Non-Functional

Design

User Experience driven only

Framework selection happened by accident

Configuration

Wrong detail configurations in frame work sections

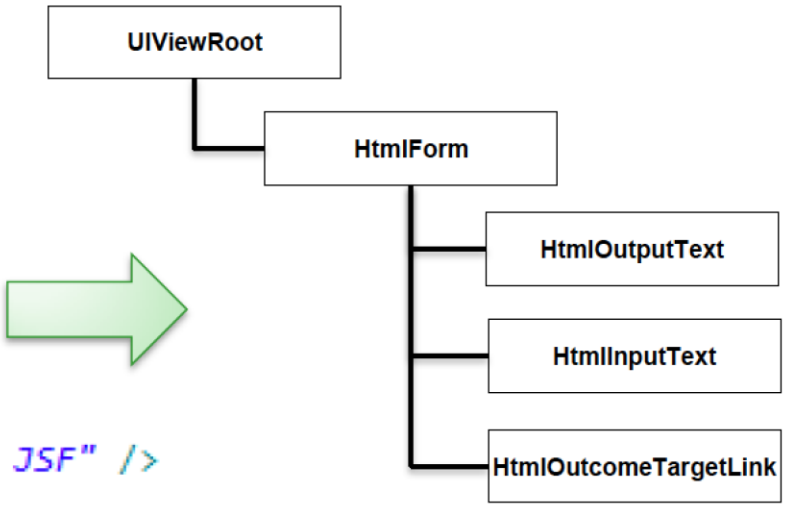
Deployment

Company Reality not respected : eg. World Wide distribution of the Organization

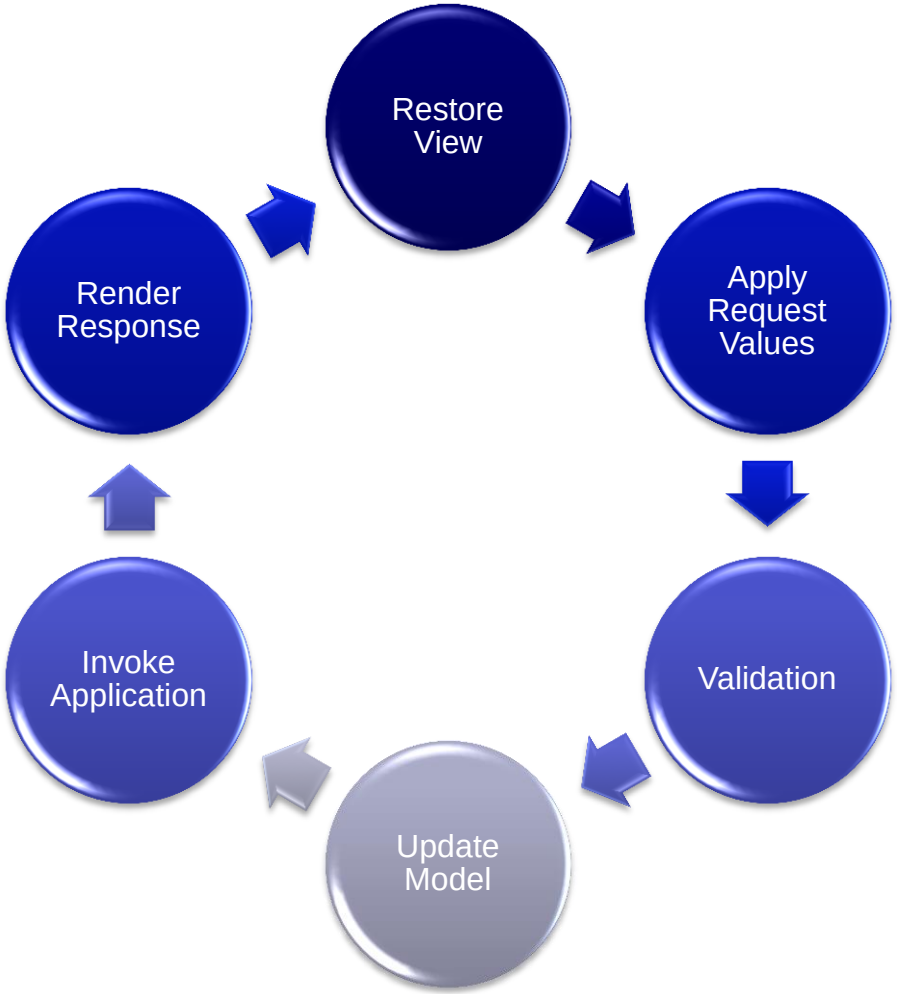
Security related infrastructure: DLP, Content Scanning etc.

JSF COMPONENT TREE

```
<h:head>

</h:head>

<h:body>
    <h:form id="form">

        <h1>
            <h:outputText value="Hello JSF" />
        </h1>

        <h:inputText id="input" value="#{flash.input}" />
        <h:link outcome="page2.xhtml" value="Page 2" />

    </h:form>

</h:body>
```



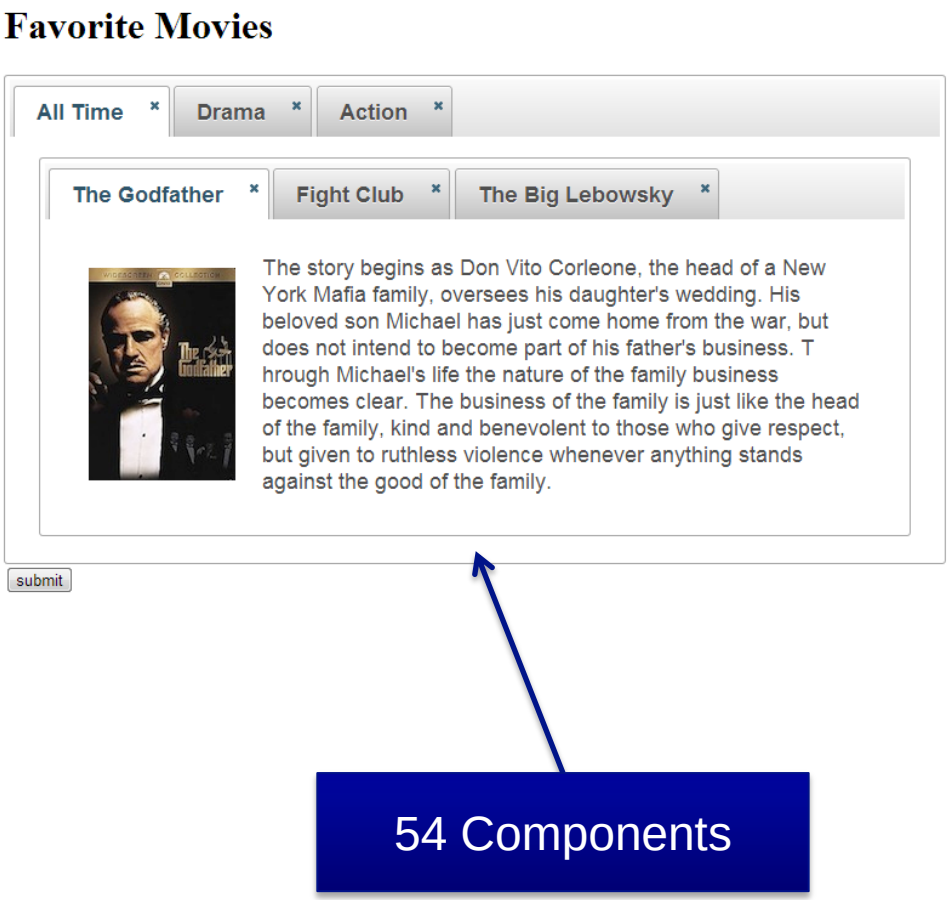
JSF LIFECYCLE



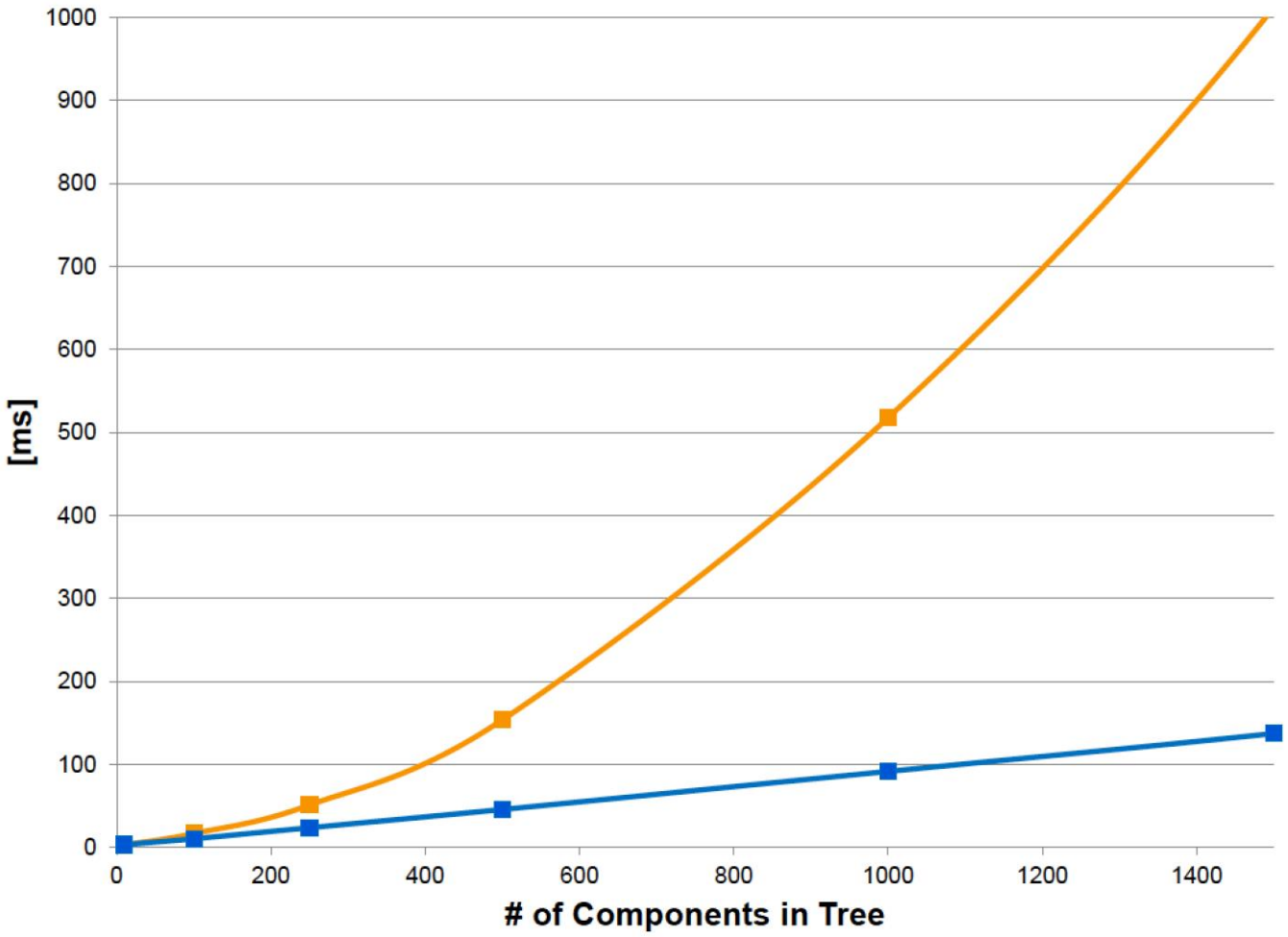
GENERATED DEADLY STUFF

Observed:
- < 50 rather rare
- Average amount : 250
- Exceptional Cases > 3000

Cause:
- Unneeded Use of JSF Components
  - Wrong usage of Composite Components
- “Dead Code” : rendered=“false”
  - Anyway part of the component tree
- Complexity
  - eg. nested Tabbed Panes



IMPACT OF FRAMEWORK SELECTION



SO WHERE'S THE BEEF THEN

Generated webpage content is massive bigger than source -> Network traffic is much higher than expected

The component tree is traversed several times and therefore passes all the Lifelines more than one time

Many frameworks use massive client side Javascript and AJAX interaction -> Rendering is time consuming -> Client side caching uses much memory on the client

Multi traverses of the Component trees can lead to multiple Database accesses

UI FUNNIES

Business likes User Experience:

Combo boxes that look up potential further entries after typing a first value

- Tons of Web and Database requests

Sorting and filtering Tables within the Browser

- Client Side Data Caching or Database Queries

Several Composites per Page

- Massive Component Trees

ORM ISSUES

- **Data model often driven by the Entity Classes**
  - Data model is not based on a logical Design
  - Data model is “unknown”, Data types might be to generic
  - Data model often slow if used for reporting
- **Table mapping**
  - Constraints and referential integrity might be ignored
  - Wrong join strategy used for master detail tables
- **ORM Performance**
  - We have seen Applications that use more than 50 % of the total End-2-End time
  - Often: no caching is used therefore all the objects will be retrieved again and again from the DB
  - Different Session Handling Implementation (JPA, Hibernate, Toplink etc..)
  - Extra Layer, extra time

THE WHOLE BUNCH