THE UNCERTAIN WORLD OF APPLICATION ARCHITECTURE

Dr. Paul Dorsey - Dulcian, Inc.

Michael Olin – Systematic Solutions, Inc.

NYOUG - June 7, 2011

Which Church is True?

"…I asked the Personages who stood above me in the light, which of all the sects was right (for at this time it had never entered into my heart that all were wrong)—and which I should join. I was answered that I must join none of them, for they were all wrong." – Joseph Smith

"Old School" Application Architecture

- Easy decision – not much to choose from
- You worked with what the platform supported

Punch Cards

3270 Terminal

Making Progress…

SQL*Forms

Too much of a good thing…

JavaEE Open Source

From left to right: The controls for a cable box, DVR, DVD, television, audio amplifier and VCR

Getting better…

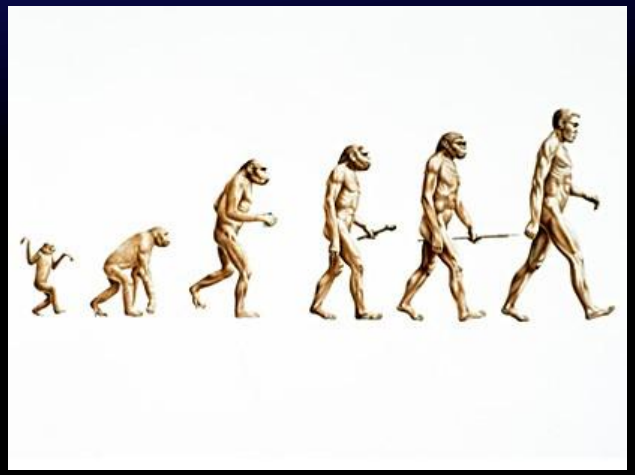
Oracle's ADF

SQL*Forms / Oracle Forms

- Oracle Forms 4.5 released in 1993
  - First “GUI” version that worked
  - Built in IDE
  - PL/SQL
  - Client / Server
- Huge numbers of Forms applications built and deployed
  - Many sites deploy thousands of Forms applications internally
  - Oracle Applications (Financials) built in Forms

Forms Evolution

- Forms 6i
  - Shift to the web
- Forms 9
  - Desupport of C/S
- Forms 10 & 11
  - Some minor modifications



MINIMAL New Features since 4.5

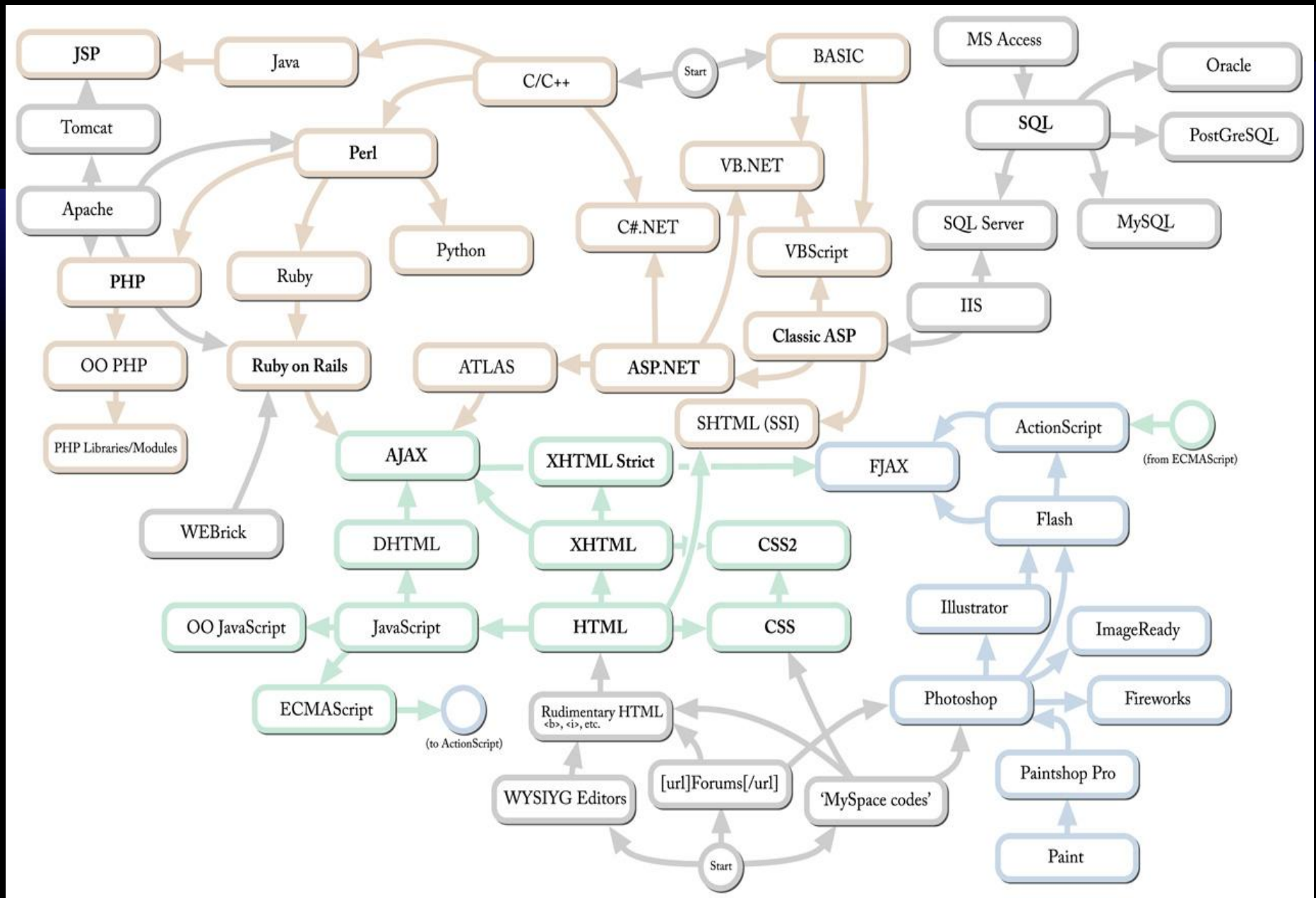
Oracle Application Express (APEX)

- Grew out of “WebDB”
- Free
- In the database, built by database group (not application development team)
- Thick database approach
- Popular (more than JDeveloper/ADF)
- Limited in functionality

JavaEE

- Thousands of “products”
- 10 projects => 10 architectures
- Last year’s best practices != this year’s
- Still evolving
- Developers are database-hostile

One JavaEE Architecture



Oracle's Application Development Framework (ADF)

- JavaEE-based
- LOTS of moving parts
- High risk, high complexity
- Still evolving
- Better than anything else in JavaEE space
  - But still not easy or nice
- 10 projects => 8 architectures

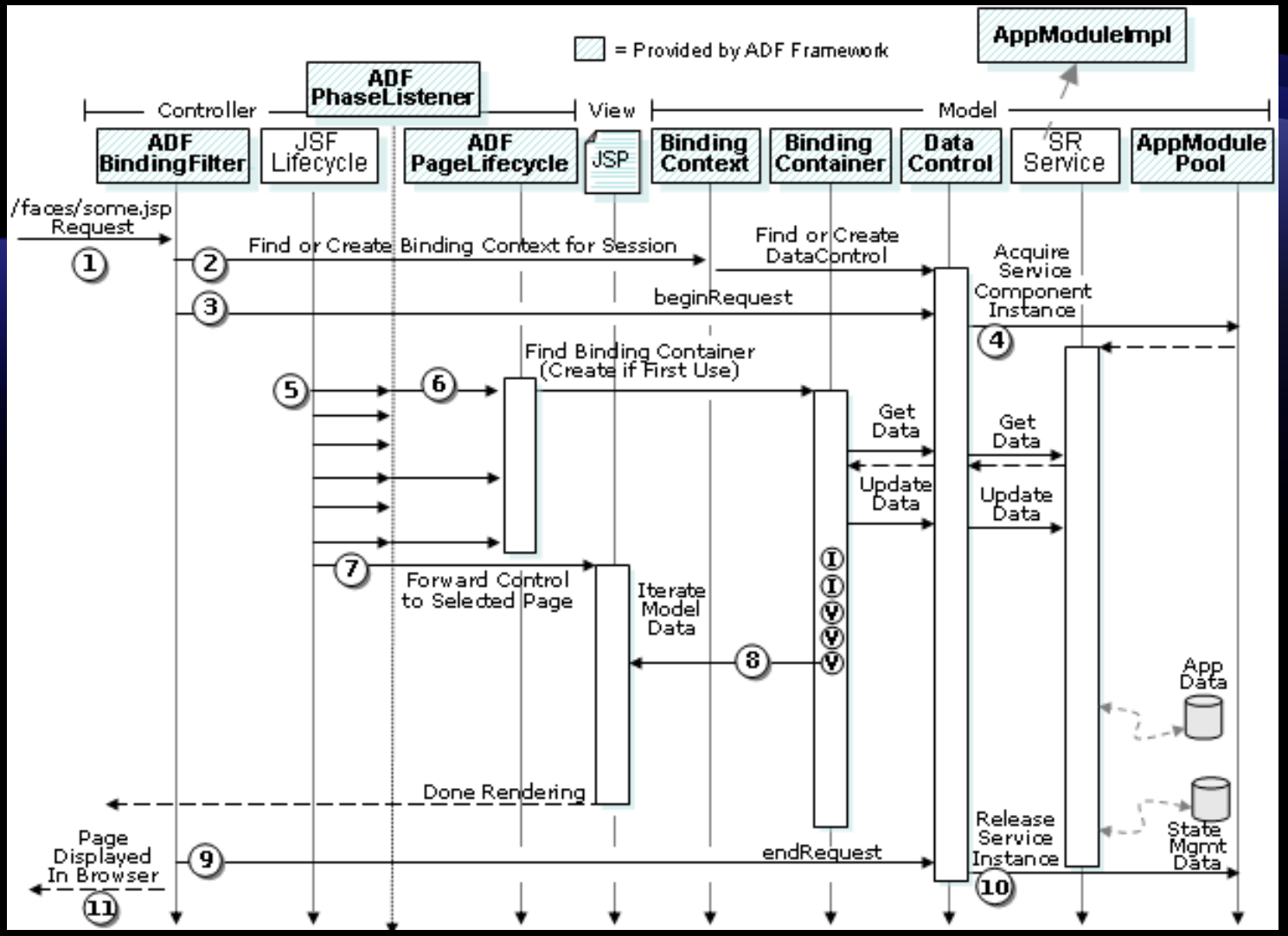
:block.item


```
<af:selectOneChoice

binding="#{ProjectSearchBean.cityNr}"
</af:selectOneChoice>

public class ProjectSearchBean extends ....
{
private Number cityOid = new Number(0);
...
...
public void setCityOid(Number cityOid)
  {
    this.cityOid = cityOid;
  }

  public Number getCityOid()
  {
    return cityOid;
  }
...
...
...
}
```



The Cycle of Life of an ADF Application

.Net

- Not as hard to learn as JavaEE
- Not as big a world as JavaEE
- Developers are database-clueless (and intend to stay that way)

What are we doing?

- Michael:
  - Thick database
    - Concentrating on back-end in PL/SQL
  - Still supporting front ends in Oracle Forms
    - …or client/server PowerBuilder
    - and thick-client .Net (no middle tier)

What are we doing?

- Paul:
  - Lived for years in Forms
  - Tried to make ADF work
  - Looked carefully at APEX
  - Wrote his own framework
    - “Forms-like”
    - Will be presented this afternoon
  - Thick Database approach

What is everyone else doing?

1) JavaEE and SOA - and mostly failing
2) .Net and surviving but with bad applications
3) APEX and mostly happy with limited applications
4) Staying with Oracle Forms and praying for a miracle

What do we suggest?

"The average lifespan of a deployed application is 5 years" – NOT!!

- Select an architecture and go
  - Minimize # of moving parts
- Thick database
- Count on a LONG ramp up time
- When you are finished, the world will have moved on
- Or use Paul's stuff

What is a "Thick Database" Approach?

- Back-end database platform rarely changes
  - This has been a best practice for years.
- Build as much business logic as possible into the database.
  - Keep the logic close to the data.
  - Make a view for each screen.
- Limit "architecture" choices to selecting the front end for UI deployment.

Summary

- No clear answer
- No good answer
- World is still evolving.
- Risk of failure is higher than ever
- Thick database is your only defense.