Wrap up

COSC 617

Rails vs. X

- Rails vs. X
  - JEE
  - .NET
  - PHP
  - Web2Py
  - Django
- How to address this?
  - platform/OS
  - Languages
  - Performance
  - Reliability
  - Future

My take

- .NET – if you're a Microsoft developer
  - alternative languages – C# , visual Basic.net, ASP .NET, etc have appeal
  - mature tools
- Java EE
  - old reputation for being painful and hard to configure
  - Good tools
  - Pro: seemingly millions of available APIs
  - Con: seemingly millions of available APIs
  - New, more dynamic languages on JVM: JRuby? Groovy?

Other platforms?

- PHP
  - “Just a templating engine on steroids”?
  - powerful, but maybe not as good as MVC Separation
- Sails – Rails for Java
- Grails – Groovy on Rails
  - produces JEE 5.0 deployable apps
- Web2Py
  - dynamic language
  - framework new, scalable?, reliable?
- Django
  - Another Python framework - http://www.djangoproject.com/
- Others: http://en.wikipedia.org/wiki/Web_application_framework

Rails

- powerful, slick and easy
- testing functionality is first-rate
- Good model of reasonable defaults with ability to customize
- But, some of it seems half-baked
  - SOAP/web services
- Ruby language is interesting but debugging can be painful (!)
  - Same for rails framework

ORM Differences

- J2EE
  - Hibernate – lots of configuration – specify what you want?
  - ActiveObjects – mirrors active record
    - Little configuration, migration support etc.
- Rails
  - easy to do simple things
  - May have to go to "bare bones" - SQL – to do many more complicated things.
- Is that what we want?
  - Easier to do the straightforward things but harder for the complex?

Other differences

- MVC
  - Java – in API, and in theory
  - Ruby on Rails – everywhere – hard to avoid
  - Good – enforces discipline
  - Bad – Straightjackets design
    - Some things don't fit quite so easily in

- Testing
  - Junit in Java
  - Expensive add-on tools in other frameworks
  - RoR – test-driven development for the rest of us.

Performance

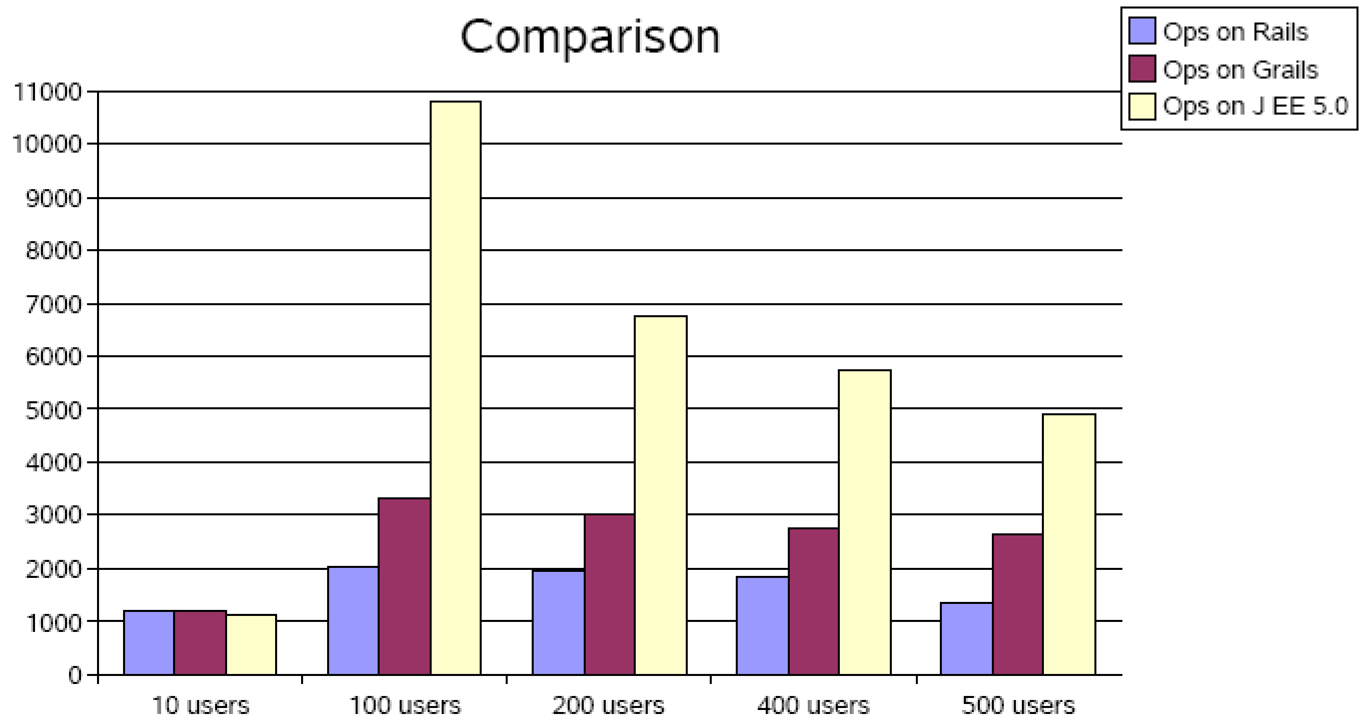
- Which platform is
  - Fastest
  - most robust
- Is platform X
  - fast enough
  - robust enough?.. for my project
- Quick & Dirty vs. Thorough and detailed
  - do you have time to set up shop to do benchmarking
  - When will my server crash?
  - when do I need more front-end servers?

Careful comparison

- Example from 2007 JavaOne conference
  - http://developers.sun.com/learning/javaoneonline/2007/pdf/TS-9535.pdf?
- Strategy
  - build an app in 3 platforms: Grails, JEE, Ruby on Rails
  - use test harness and server to simulate loads
  - Same hardware, etc.

Java EE5, Grails, Ruby on Rails analysis

http://developers.sun.com/learning/javaoneonline/2007/pdf/TS-9535.pdf?



Conclusions to be drawn from this

- Java can appear to be faster
  - by one set of measurements..
  - conducted by Sun employees

- Viewpoints
  - System you'll deploy will be run on next year's hardware
  - If you need to extract every bit of performance, you may need to do custom coding anyway.

- Otherwise, ease trumps performance, maybe?

Web Frameworks in the evolving context of software engineering

- 1980s
  - development methodologies and early tools
  - waterfall/spiral
  - Object-Oriented
  - CASE

- early 1990s
  - patterns and UML

- Late 1990s- Present
  - Deep and powerful APIs – Java – networks HTTP – xml, etc
  - AOP,
  - ORM
  - Agile, XP, Scrum

General Trend

- Tools pushed to higher-level
  - Support for AOP and other models from research
  - Increase support for understanding of how work gets done
- Agile vs. waterfall
  - Life-cycle considerations
- Plan for evolution
  - perhaps the biggest strength of Rails
  - Testing
  - Deployment

The future – what will change

- Tools will continue to get higher-level, shielding from more detail
- Without necessarily compromising on performance or reliability
- “Baked-in” support for testing and evolution will become more common-place
- Web standards will evolve
- Easier development for multiple devices - mobile

What won't?

- individual needs of specific customers/institutions will continue to differ enough to require custom development
- Support for legacy databases and integration with old tools will continue to be important
- Browsers & Javascript will continue to be buggy and insecure.