On The Internet Nobody Knows
You're a Lambda

Opportunities and Challenges for Developers of
Lisp-Based Web Applications

Brent Benson
Oracle Corporation
On the Internet, nobody knows you're a dog.
Outline

• Traditional barriers to using Lisp for commercial application development
• Advantages and opportunities for developing web-based apps in Lisp
• Challenges and issues with using Lisp for web-based apps
• Quick mention of some tools, success stories and conclusion
Replies to “I want to use Lisp”

• “Lisp is not efficient enough”
• “We won’t be able to find programmers that know Lisp”
• “The runtime system for Lisp is too big and we can’t create standalone executables”
• “Lisp uses a garbage collector, and garbage collectors are less efficient than manual storage management”
On the Internet

• Application delivery mechanism: URL
• Application delivery tool: Web browser
• Things that are important to users
  – Features, performance, ease of use
• Things that the user doesn’t care about
  – Operating system
  – Programming language
Web-Based Apps: You’re In Control

- Application resides on the server
- Operating system is up to you
- Programming language is up to you
- Runtime library size and small executables a non-issue
- Free to choose based on productivity, reliability, cost
- Competitors have this same freedom, so choose wisely
Interactivity

• Interactive natures of Lisp development environments a key advantage
• Read-eval-print loop allows for quick prototyping and testing
• Integrated debugging allows for quick pinpointing of problems and immediately testing new code
• Delivery of fixes without bringing down server
Better Languages Attract Better Programmers

• A project doesn’t need thousands of programmers, just a few good programmers
• People who know Lisp are often thoughtful programmers who are driven to the best tools
• Application development is expensive, better programmers pay off
Closures and Continuations

- Handling arbitrary user navigation in a web browser is a difficult problem.
- Back, forward, refresh, and history can take the user to arbitrary application points.
- Application state can be saved in a closure or continuation and mapped to page views.
- Clean model leads to ease of programming and better user experience.
Macros

- Web-based applications have a great deal of redundancy and cry out for abstraction layers
- Declarative languages are a natural approach
- Main-stream declarative approaches lead to multi-level systems with different tools (CSS, JSP, templates)
- Macros provide a clean way to construct declarative abstraction layers using same tools, debuggers, etc.
Challenges

• Perception
  – Often need buy in from investors
  – Decide whether to be open about direction and explain advantages
  – …or gloss over and stress web standards

• Integration
  – With existing applications or libraries
  – Use web-services approach
  – Use Apache extensions or Java-based Lisp system
Challenges (continued)

• Support
  – Commercial software
    • Choices are fairly limited
  – Open Source software
    • More choices, some companies and individuals will provide paid support

• Not Invented Here
  – Using non-main-stream doesn’t mean that you shouldn’t continue to monitor new technologies and/or rethink approach
Tools

• Common Lisp
  – CL-HTTP
  – AllegroServe
  – Araneida
• Apache: mod_lisp
• Scheme
  – SISCWeb
  – PLT Scheme
  – JScheme, Kawa
Conclusion

• Web-based application users don’t care what technology is used on the server
• Lisp-based tools and environments have some real advantages for web-based apps
• Success stories: Yahoo! Store, Orbitz
• Ignore high productivity tools at your own peril!