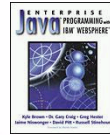


J2EE -- What's in it for me?

J2EE architectures, benefits and problems solved

A bit about me...

- Kyle Brown is a Senior Technical Staff Member for the IBM Software Services for WebSphere group (see www.kyle-brown.com)
 - ▶ My group is a consulting group that is part of IBM Software Group with the mandate to "Make our products stick" with our customers
 - ▶ I perform Architecture Reviews, Design and Code Reviews for WebSphere customers to discover how well their systems conform to accepted best practices and patterns
 - ▶ I also trouble-shoot critical customer problems with our tools
- I've been in the OO consulting field for about 12 years now
 - ▶ First with Knowledge Systems Corporation (7 years), then with IBM SSW (5 years)
- I've lectured at both NCSU and UNC-CH
- I've written three books and parts of three others
- The one YOU care about is the one we're using for the text of this course
 - ▶ We used the previous edition as the text for the last version of this course



A bit about the book

- Enterprise Java Programming with IBM WebSphere (2nd Edition)
 - ▶ Kyle Brown, Gary Craig, Greg Hester, Dave Pitt, Russ Stinehour, Jim Amsden, Peter Jakab, Mark Weizel
- Teaches the layering principles we've touched on in the lecture
- Explains the technologies in each layer
- Gives examples of how to use each technology in isolation, and in a full system with all the layers

IBM Software Group



Overview

- What are the kind of issues that real-world web sites face?
- How do you address those issues?
- What choices does J2EE give you?
- How do you decide which architectural choices to make?
 - ▶ E-bay
 - ▶ Schwab
 - ▶ LL Bean
- Where do we go from here?

IBM Software Group

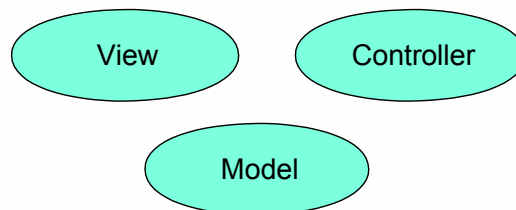
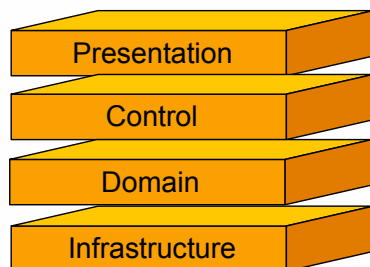


What's the problem?

- In theory, the web is simple
 - ▶ Anybody can put up a web site, right?
- In reality, it can get really complicated
 - ▶ Millions of hits, thousands of users, ever-changing requirements.
 - ▶ How do you get scale, security and reliability?
 - ▶ While the hottest web technologies may be cool for the front end the REALITY is that all of the key corporate data (the crown jewels) are kept in large, often very old back end systems.
 - ▶ You've got to integrate with other technologies. CRM, Supply chain, Departmental apps, all fit into the picture.
- How do you get a handle on how to deal with all of these issues?

Layering and MVC

- The basic principle we have to assume all applications follow is layering
 - ▶ Separate concerns in order to keep your code from becoming a hopeless pile of spaghetti
 - ▶ Apply standard OO principles and patterns to ensure that your code has a recognizable design and architecture
- The MVC Pattern is a standard application of layering
 - ▶ Model -- Represent the business logic
 - ▶ View -- Represent a way of interacting with the model
 - ▶ Controller -- Mediate between the two, and manage application flow
- Cleanly separate presentation (View) code from business (Model) code

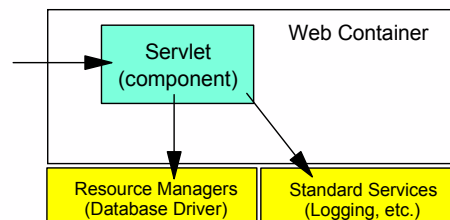


What is J2EE?

- A set of APIs for Server-side Java Programming
- Comprehensive support for
 - ▶ Web Development
 - ▶ Transactions
 - ▶ Distributed Objects
- Java 2 Enterprise Edition (J2EE) defines a collection of APIs for implementing Enterprise Java Applications
 - ▶ Enterprise JavaBeans (EJB), Servlets, JavaServer Pages (JSP), Java Database Connectivity (JDBC), ...

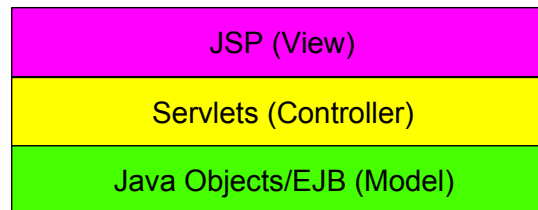
Containers and Components

- The most basic idea behind J2EE is the division of labor between containers and components
- Containers provide the environment in which components execute
 - ▶ Containers provide the implementation of the J2EE API's
 - All containers look the same to components
 - Give vendors the freedom to innovate internally and to differentiate themselves by providing different tools, quality of service, etc.
 - ▶ Provide control by providing the calling framework
 - ▶ Containers handle
 - Concurrency, Consistency, Security, Availability, Scalability, Administration, Integration, Distribution, Data Access
- Components handle
 - ▶ Presentation, Business Logic, Data Access



J2EE and MVC

- The MVC Pattern is at the heart of J2EE
 - ▶ Model - Represents the business logic
 - ▶ View - Represents a way of interacting with the model
 - ▶ Controller - Mediates between the two, and manage application flow
- **MVC is just a specific example of application layering**
 - ▶ Each layer has a specialized purpose; additional layering within the application is possible
 - ▶ Layers effectively encapsulate bits of the application's behavior
 - ▶ Different J2EE technologies fit best within particular layers



Java and the Web

- Java Servlets are small, server-side Java programs that respond (typically) to client HTTP requests
 - ▶ similar to a CGI program
- Java Server Pages are a way for embedding Java code into server-side HTML pages
 - ▶ A template language for Java Beans and server-side HTML processing

Java Servlets

- A Java class that represents a single URL to the client
 - ▶ Defines a `service()` method that handles HTTP requests
 - ▶ `HttpServletRequest` -- access request data
 - ▶ `HttpServletResponse` -- reply to the client
- An instance of each class is a shared resource used by multiple threads
 - ▶ Each thread handles an HTTP request

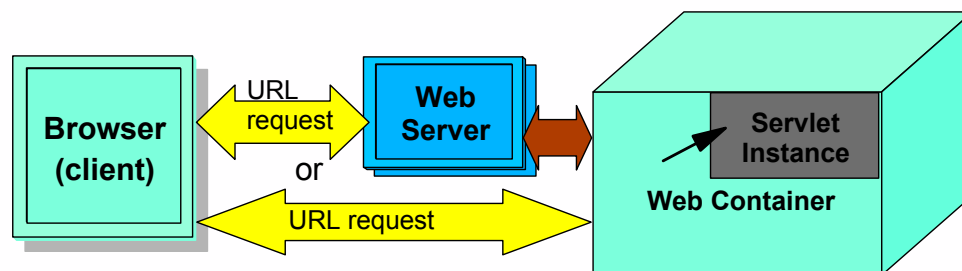
```
public class MyServlet extends HttpServlet {  
    public void service(  
        HttpServletRequest request,  
        HttpServletResponse response)  
        throws ServletException, IOException {  
        response.setContentType("text/html");  
        // get stream to output HTML on!  
        PrintWriter out = response.getWriter();  
        // send out a simple banner  
        out.println("<HTML><BODY>");  
        out.println("<h1>Hello World!</h1>");  
        out.println("</BODY></HTML>");  
    }  
}
```

IBM S}



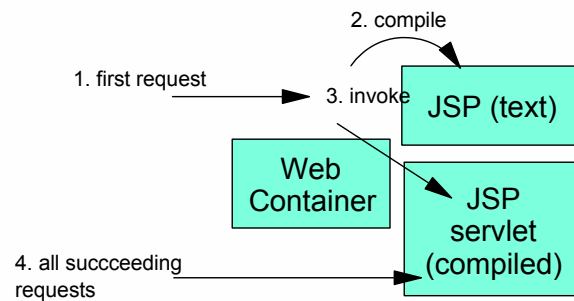
Generic Servlet Invocation

- Client makes a request of a Servlet through an URL
- (Optional) Web Server forwards request to Web container
- Web Container locates an instance of a Servlet class
- Servlet engine calls Servlet's **service** method



JavaServer Pages

- JavaServer Pages (JSP) is a standard HTML "template" language
 - ▶ Embed standard JSP tags into an HTML page
 - ▶ Embed Java code (scriptlets)
- JSPs are converted to servlets at runtime
 - ▶ Page compilation triggered by changes in the JSP, or on first invocation
 - ▶ JSP Source is parsed
 - ▶ Java servlet code is generated
 - ▶ This "JSP Servlet" is compiled, loaded and run



IBM Software Group



XML and XSLT

- Another way to produce HTML output is by translating XML into HTML
- XSLT (eXtensible Stylesheet Language Translation)
 - ▶ A rules-based language for defining how to translate XML elements into another form of output (often HTML)
- A Servlet (or similar controller) generates XML which is transformed (using XSL Stylesheet transformation) into output HTML
 - ▶ Customized HTML, WML or XML output styles
 - ▶ Often useful when you have multiple possible text transformations (WAP, HTML, etc)

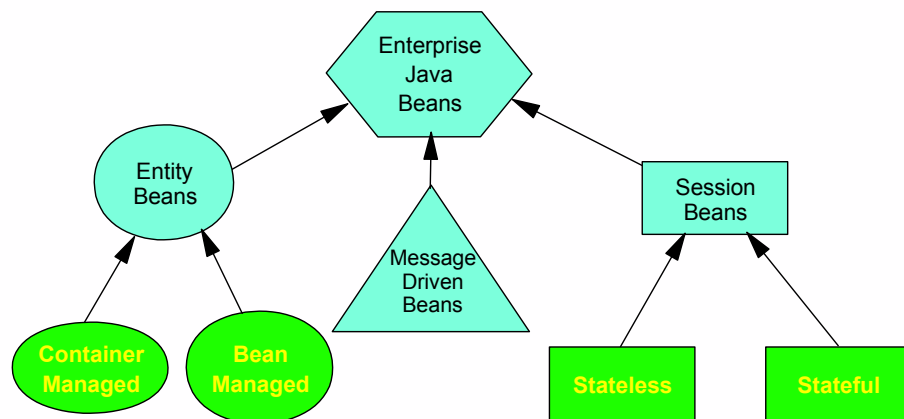
IBM Software Group



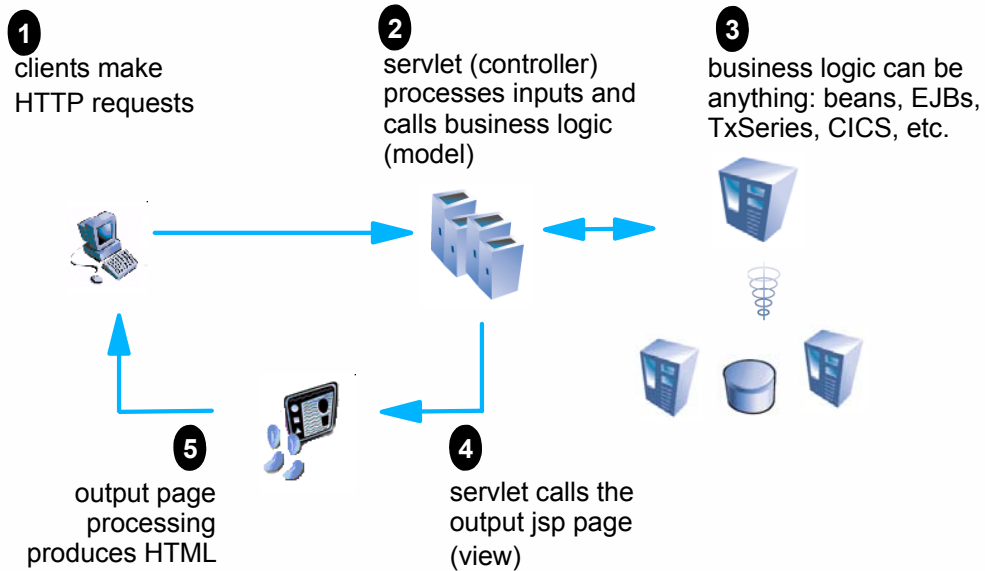
Enterprise Java Beans

- A component model for developing and deploying object-oriented, multi-tier Java applications
- Based on three foundational technologies
 - ▶ OMG's CORBA
 - ▶ RMI (Java Remote Message Invocation)
 - ▶ X/Open XA Transaction architecture
- So what's an EJB?
 - ▶ An EJB is a distributed, transactional, (possibly) persistent software component
 - ▶ Programmer provides a specification and implementation for its business methods
 - ▶ Transaction attributes and persistence can be provided at Deployment time

EJB Types (EJB 2.0)



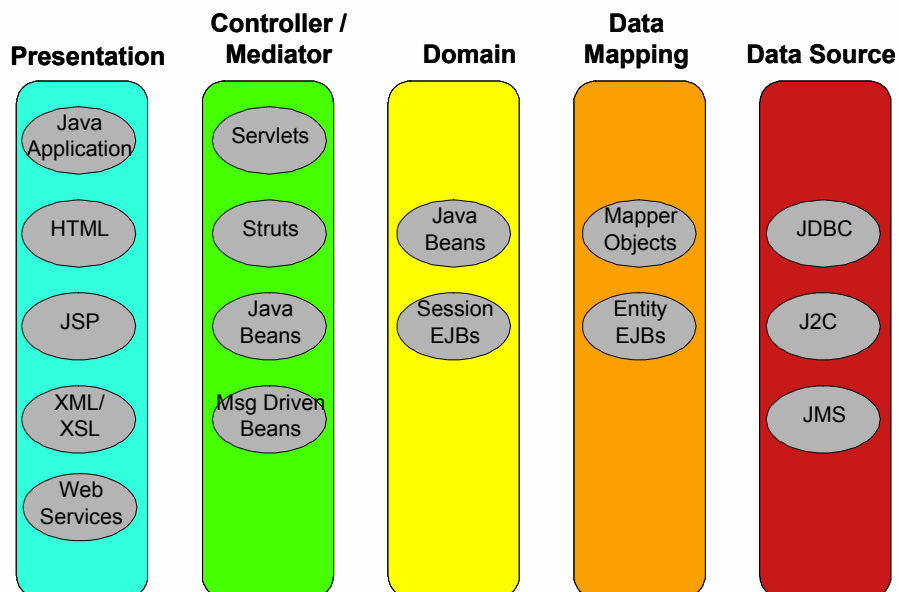
MVC Program Flow



IBM Software Group



J2EE Technologies in Layers

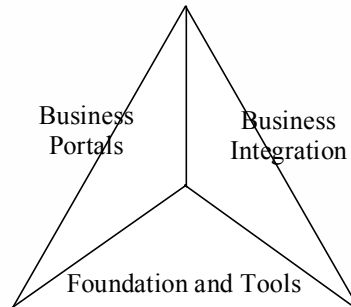


IBM Software Group



What's a WebSphere?

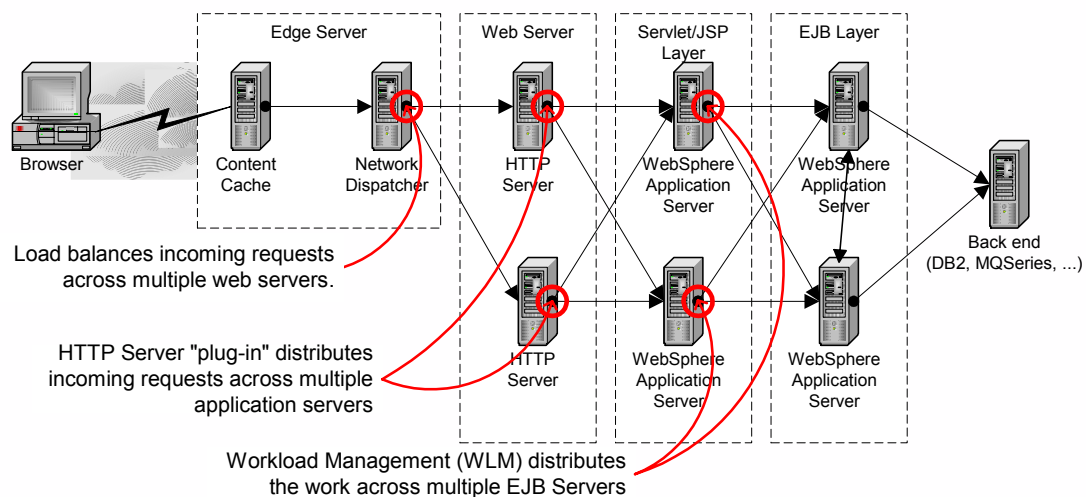
- WebSphere Application Server implements the set of J2EE technologies we've been discussing
- WebSphere is a family of products from IBM including the Application Server, development tools (WebSphere Studio) and tools and frameworks for business integration and user reach and experience
- You'll learn about the WebSphere foundation and tools in this course



IBM Software Group



WebSphere J2EE Network Topology



IBM Software Group



Case Study: eBay



eBay

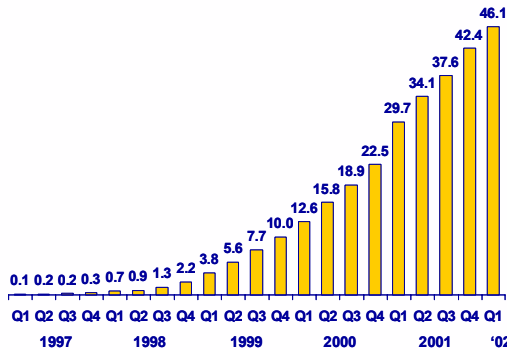
- eBay was founded in 1995 and now has more than 68.8 Million registered users
 - ▶ 31.1 Million active users – at least one visit per month
 - ▶ more than 1500 concurrent transactions per second
- Gross Merchandise Sales 1Q03 = \$5.31 Billion
- # of items listed in 1st Q 2003 = 220 Million
- In the beginning of year 2000, the management team at eBay realized their existing software architecture would not be able to meet their expected growth in the near future

eBay growth



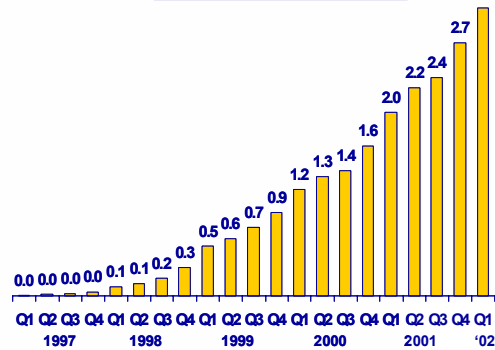
Registered Users (Millions)

46M Registered users
40K New users per day
1 new user every 2 seconds



Gross Merchandise Sales (\$Billions)

\$12.4B Annualized run rate
\$35 Million per day
\$400 per second



IBM Software Group



eBay Technology Drivers

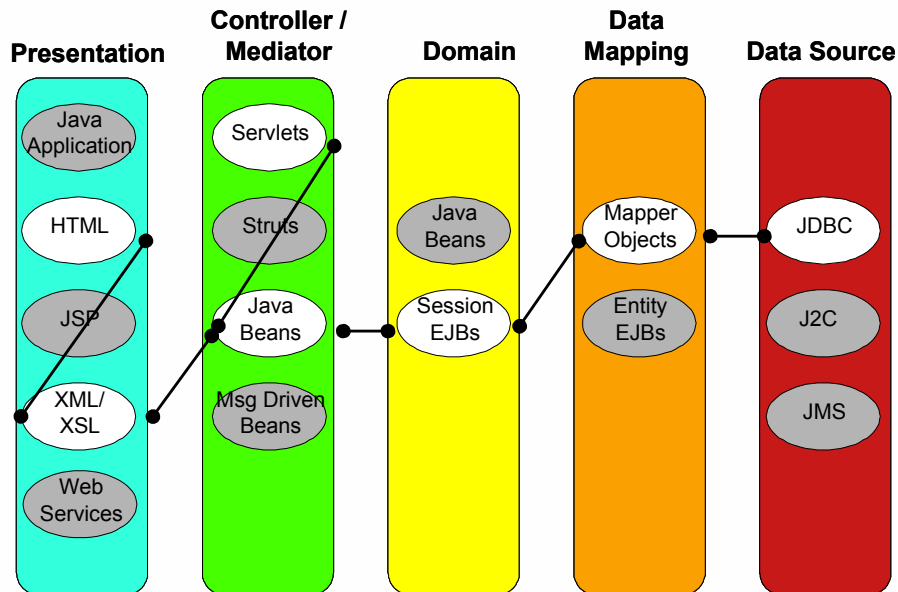


- Availability
 - ▶ The system should be available all the time (24 X 7)
 - ▶ No downtime for software updates or code releases
- Reliability
 - ▶ The system must not have single point of failure
 - ▶ It should support more than 99.9% uptime
- Performance
 - ▶ The system should have better response time and transaction throughput than the old system which is written in C++ and ran inside IIS as an ISAPI extension
- Scalability
 - ▶ The system should be able to handle future business growth seamlessly

IBM Software Group



J2EE Technologies at eBay



IBM Software Group



eBay Technology Choices



- So why did eBay choose the technologies that it did?
- XML/XSLT
 - ▶ eBay has a standard set of pages that are displayed in a lot of slightly different ways
 - ▶ business partners, foreign languages, etc.
 - ▶ Most pages are rather similar because they work from a common set of objects (items, auction, buyer, seller...)
- EJB
 - ▶ needed secure, centralized domain logic
- JDBC and Mapper objects
 - ▶ Performance is absolutely CRITICAL to eBay.
 - ▶ Highly customized and tuned database tables and database queries

IBM Software Group





Initial Topology Design - 20-Node Clusters

- 3 WebSphere domains with 20 nodes in each domain
 - ▶ Avoid single point of failure in case the admin repository is inconsistent
 - ▶ Each domain can be taken down for software upgrade or code release individually, ensuring 66% capability at all time
- Oracle database is used as the admin repository
- IP Sprayers are used to load balance users requests between HTTP servers
- HTTP servers are separated into two categories, core and international
 - ▶ Historically core and international sites were served by different HTTP servers
 - ▶ Simplify virtual hosts configuration in WebSphere plugin-cfg.xml
- Hardware load balancers (F5 Big IP) are used to load balance between app servers
 - ▶ Provide central point of control and ease of management
 - ▶ Provide more intelligent load balancing schemes
- Both the core and international EAR files co-exist on the same machine
 - ▶ Core site is www.ebay.com
 - ▶ International sites are: fr.ebay.com or www.ebay.fr, de.ebay.com or www.ebay.de, etc.
 - ▶ Reduce hardware cost
 - ▶ Each EAR file has its own JVM and virtual host

IBM Software Group



Software and Hardware Used

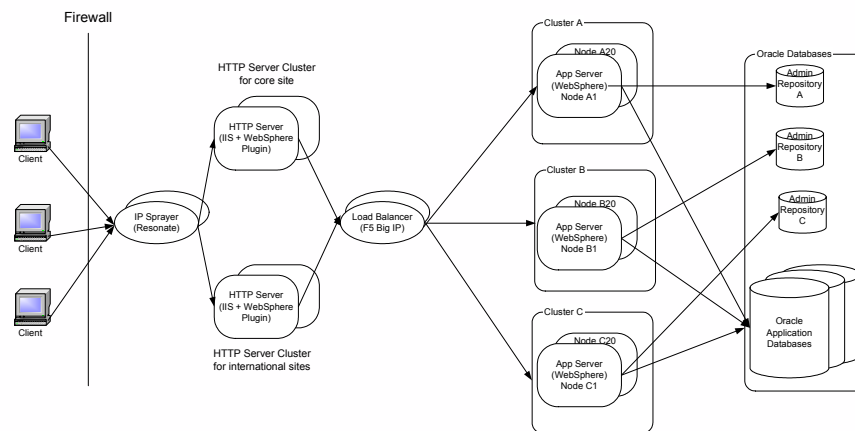
- App Server
 - ▶ Software: WebSphere AE 4.0.x running on Windows 2000
 - ▶ Hardware: IBM xSeries 330 2-way 1.2 GHz dual CPU
- HTTP Server
 - ▶ Software: IIS 5.0 running on Windows 2000
 - ▶ Hardware: IBM xSeries 330 2-way 1.2 GHz dual CPU
- Database Server
 - ▶ Software: Oracle 8.1.6 and Oracle 9i running on SunOS 5.8
 - ▶ Hardware: SUN 420 4-way
- IP Sprayer between HTTP servers
 - ▶ Resonate 3.3x on Windows NT 4.0
 - ▶ Hardware: IBM xSeries 330 2-way 1.2 GHz dual CPU
- Load balancer between app servers
 - ▶ Hardware: Big IP by F5

IBM Software Group





Initial Topology Design - 20 Node Cluster



IBM Software Group



IBM Software Group



Case Study: Charles Schwab

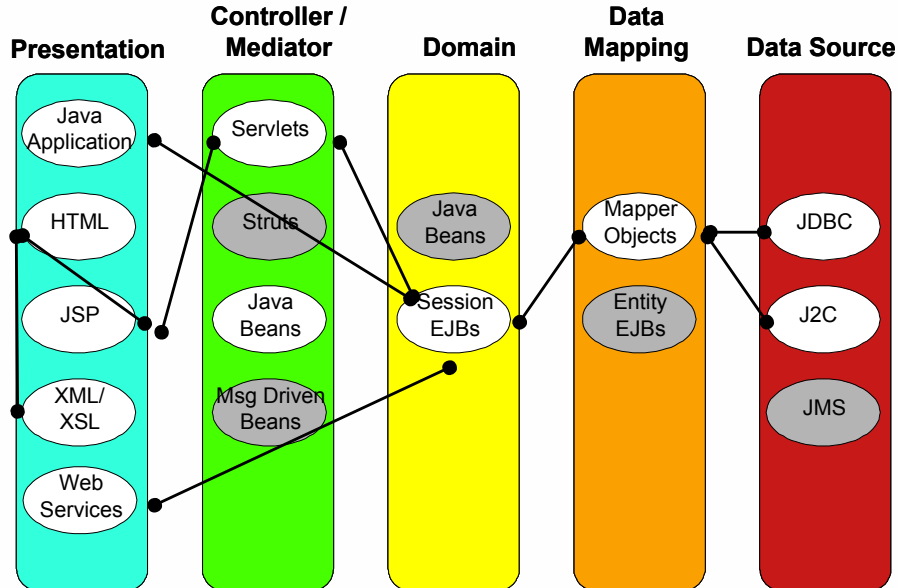
Charles Schwab

- One of the first discount brokerages, Charles Schwab was founded by Charles Schwab nearly 30 years ago
 - ▶ Today it serves nearly 8 million clients nationwide
 - ▶ Based on a philosophy of personalized advice, not driven by commission
 - ▶ Provide service through their web site and through 19,000 professionals across the country
- They provide an enormous range of investment services
 - ▶ Brokerage Services, Lending
 - ▶ Investment Advice, research and financial planning
 - ▶ Large portfolio investor services
- Their wide range of services makes for a lot of diverse requirements for their infrastructure
 - ▶ Support both their web site and their investor centers
 - ▶ Provide customized, highly targeted services for their customers

Charles Schwab Technology Drivers

- Schwab's different business drivers create lots of UI choices
 - ▶ High-net worth customers get a very high-performance, highly customized desktop UI
 - ▶ Different web customers have different needs
 - Active traders need different information than people mostly interested in retirement or college savings
 - ▶ The Web UI also has to be customized for the needs of the Investment Centers
- However, the back-end logic is all the same
 - ▶ The databases and CICS transactions that implement the back-end logic are shared by different applications
- They found early on they couldn't get effective reuse at the CICS transactions level CGI level, so they began to look at sharing higher-level abstractions
- Performance isn't everything but it is important...

J2EE Technologies at Charles Schwab



IBM Software Group



Charles Schwab Technology Choices

- Rather than choose one UI technology, Schwab allowed a number of choices
 - ▶ High net-worth customers use a Desktop Java UI Application
 - Fast UI reaction times, ability to do lots of caching
 - ▶ Most investors use a portal
 - Different clients get to choose different tools to meet their needs
- Session EJBs have provided a good mechanism for shared services
 - ▶ JavaBeans are used for the shared domain objects
 - ▶ Schwab has put a lot of effort into developing frameworks and code generators for a very configurable services architecture

IBM Software Group



Charles Schwab Topology Choices

- Well over 400 servers (4-ways)
- Subsets of the servers are used for different speciality functions
- Core functions deployed on the majority of the servers
- Schwab took a multiple-tier approach
 - ▶ Separate the UI tier from the domain logic tier, from the data services tier

IBM Software Group



IBM Software Group



Case Study: LL Bean

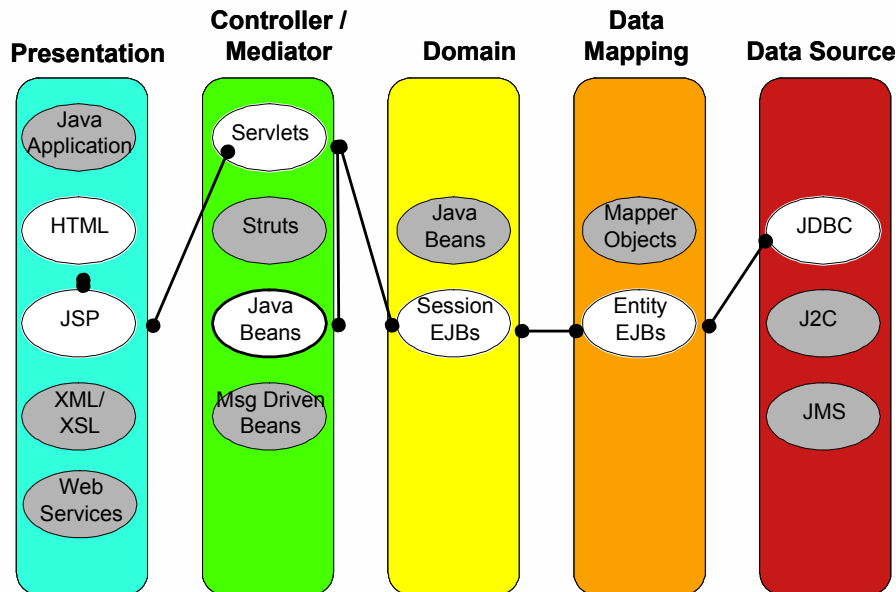
LL Bean

- Casual clothing and outdoor gear dealer founded in 1912 by Leon Leonwood (LL) Bean
 - ▶ Annual Sales of > \$1 Billion
 - ▶ Employs over 8000 people at peak holiday season, retail stores in four states, worldwide catalog sales
 - ▶ Carries over 17000 items
 - ▶ Ships over 12.7 million packages per year
 - ▶ Takes over 140,000 orders on the busiest day of the year
- A pioneer in web-based e-Commerce (llbean.com launched in 1995)

LL Bean Technology Drivers

- A key consideration in this instance was time.
 - ▶ Very short development cycles. Everything has GOT to be ready by the Christmas rush
- Technology is not LL Bean's core business
- They have similar requirements to other retailers
 - Catalog viewing
 - Shopping Carts
 - Coupon codes
 - Credit Authorization and Sales
- Considerations like this lead to a build vs. buy decision

J2EE Technologies at LL Bean



LL Bean Technology Choices

- In this case, the build vs. buy decision led to buy
 - ▶ WebSphere Commerce Suite
- WCS is a framework and reusable library of parts for building commerce sites
- Catalog functions implemented as a set of Entity beans and Services implemented as Session beans
- Sales functions customized by modifying template JSPs
- Performance is not as important as programmer productivity
 - ▶ Tune only where it's absolutely needed (core functions)



What You've Seen

- J2EE applications are built using any of a number of technologies
- That good J2EE Applications follow a distinct layered structure
- That J2EE can be used to develop scalable, high-performance web sites