Real World J2EE

Design Patterns and Architecture behind *TheServerSide.com*

Presented by:

Floyd Marinescu
Senior Architect

info@middlewarecompany.com / http://www.middlewarecompany.com

Talk outline

- A Walk through of TheServerSide.com project lifecycle
  - YAP!? – Yet Another Portal!?
  - Requirements / Design Phase
  - Implementation in J2EE
  - Deployment, Scalability Tests and Bugs

YAP!?

Yet Another Portal!?

info@middlewarecompany.com / http://www.middlewarecompany.com

Yet Another Portal?

- TheServerSide.com
  - Initially conceived of by Ed Roman in October, 1999.
- Existing Portals were vendor specific or not communities
  - www.ejbPortal.com - Persistence
  - developerWorks – IBM
  - www.ejbNow.com
- Why did The Middleware Company build theserverside.com?
    - Wanted to contribute to the community
    - Education is our business ☺

What is *theserverside.com*

- A free, vendor-neutral community portal about J2EE, implemented using J2EE.
- Features
  - Middleware News
  - Discussion Forums
  - Patterns Repository
  - User-run application server reviews
  - Articles / Tutorials, etc.

Planning for TheServerSide.com

- Buy vs. build
  - Wanted a java based forum messaging system
- J2EE promise of component based technology
  - Still too young.
  - Market places:
    - www.flashline.com
    - www.componentsource.com
- Non-Java alternatives:
  - Ultimate Bulletin Board
    - Not focused enough for our needs

Designing TheServerSide.com

info@middlewarecompany.com / http://www.middlewarecompany.com

Designing TheServerSide.com

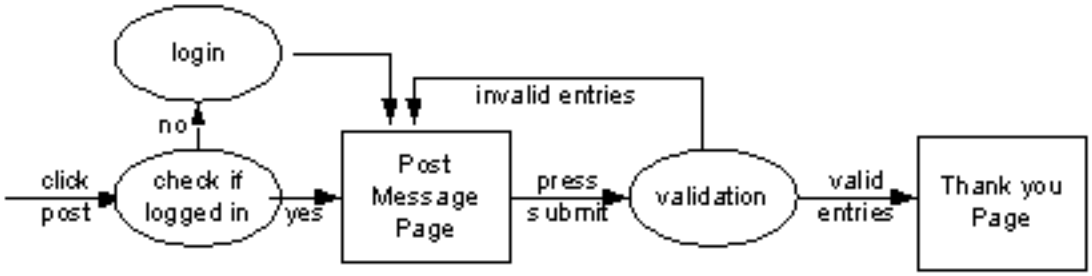
- Requirements analysis / derive usecases
- Design domain model from usecases
  - Domain model persistence mechanism
- Design business "processes" from usecase

Requirements / Usecase:

- Def'n: Usecase
  - A user interaction with the application
    - Ie: "Placing a Bid" on eBay.
- Allow you to specify the functionality of an application before you build.
- Defining your usecases are the first step in application design.
- Usecases are written down in a "Requirements Document".

Usecases on *theserverside.com*

- Post messages, reply to messages
- Browse forums, threads.
- Login, logout, signup
- Admin usecases
- Example of the PostMessage usecase, from our actual requirements doc:

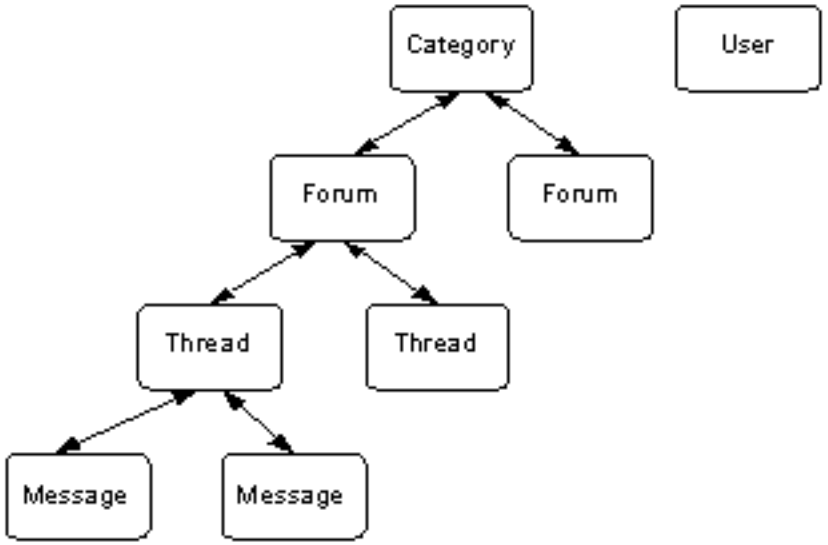


Domain Model

- The domain model
  - classes/objects that model real world aspects of your business problem.
  - Domain objects model the nouns (people/places/things) in your business problem.
  - Also known as: Data model, business objects, object model.
  - Eg. A domain model for eBay:
    - Item, Buyer, Seller, Bid, Auction.
  - Usually map from your usecases
    - Ie: Bid object, Place Bid usecase.

theserverside.com domain model

- User
  - models a person using the site
  - Has a name, password, address, email, etc.
- Message
  - Encapsulates a message with a subject and a body



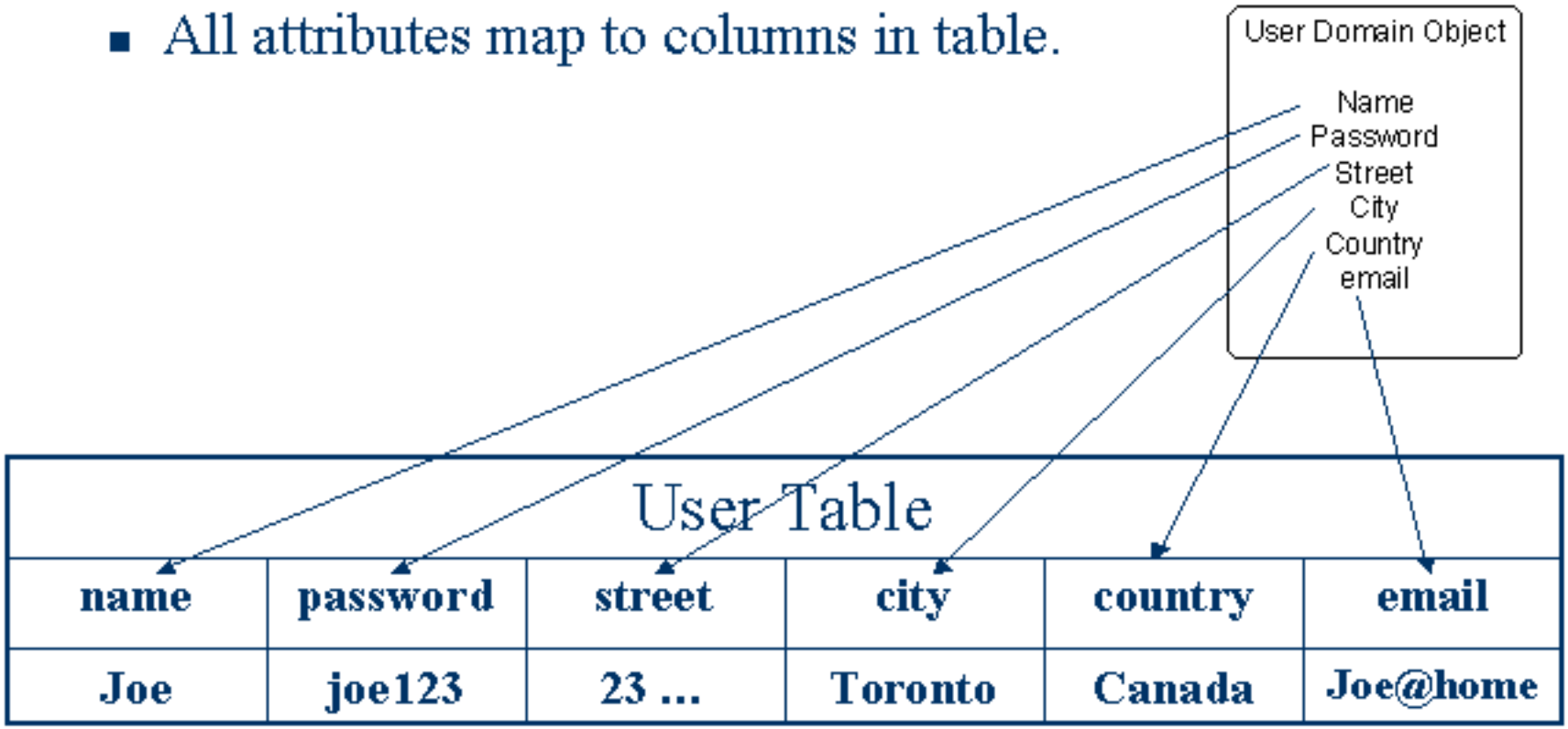
Domain Model Persistence

- Persistence
  - The saving of information to some permanent store
    - Database, files, etc.
  - Persistent objects survive system failure.
- Problem: How we do save the state of our domain model?
    - Eg: Joe signs up on theserverside.com, we need to save is information so he can login at a later time.
- Answer: We need to persist Joe's information to some permanent store.

- How do we persist our data?
  - Most common format: SQL database
    - Classes in our domain model map to tables in our database.
    - How? Attributes of the class map to columns on the table

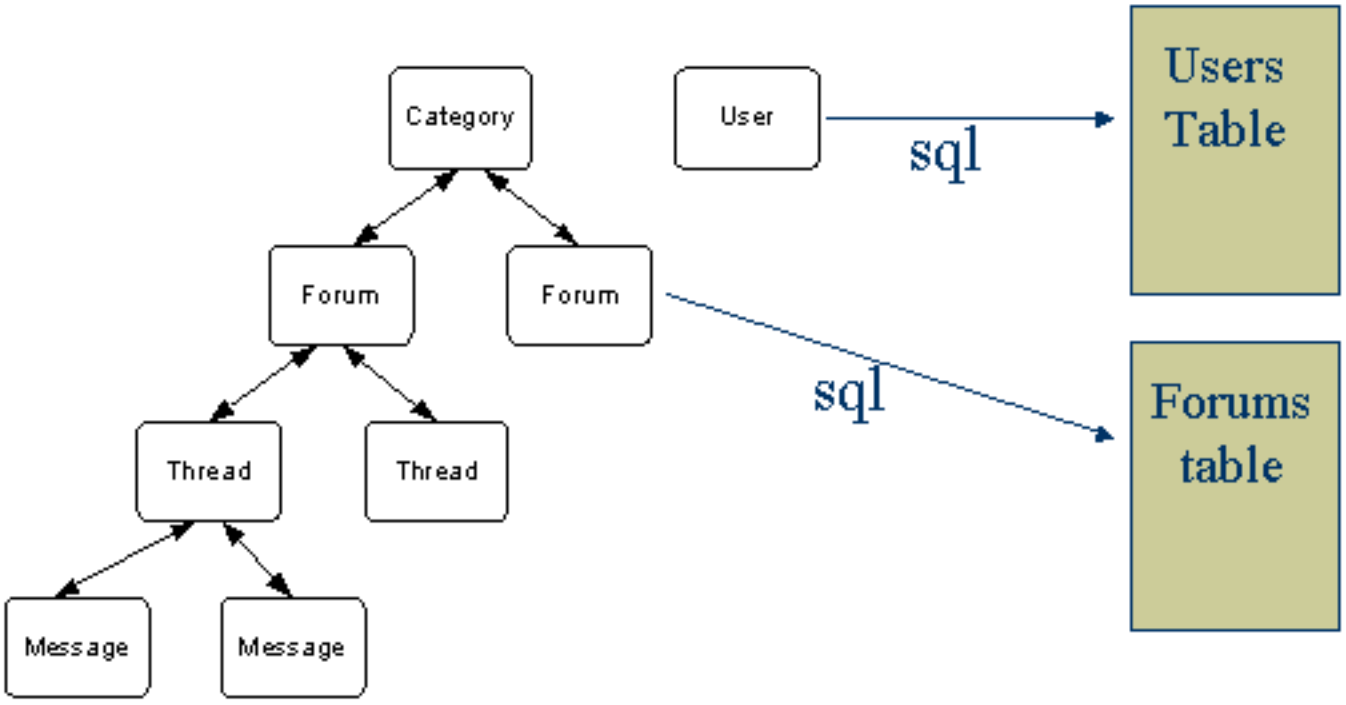
Persistence Mechanisms

- Mapping Objects to Tables
  - Object maps directly to table
  - All attributes map to columns in table.



Persistence on *theserverside.com*

- Domain model objects map to their own tables in the PostgreSQL database.



Business Processes

- The units of work in an application
- Model the "verbs" in your application
  - Eg: "run credit check", "login", etc.
- Business processes are your usecases and services used by the usecases
  - Eg: Placing a Bid is a business process & usecase (user interaction) on eBay
    - Placing a bid may also require a credit history check (which is not a user interaction)

Implementing TheServerSide.com

With Java 2 Enterprise Edition

info@middlewarecompany.com / http://www.middlewarecompany.com

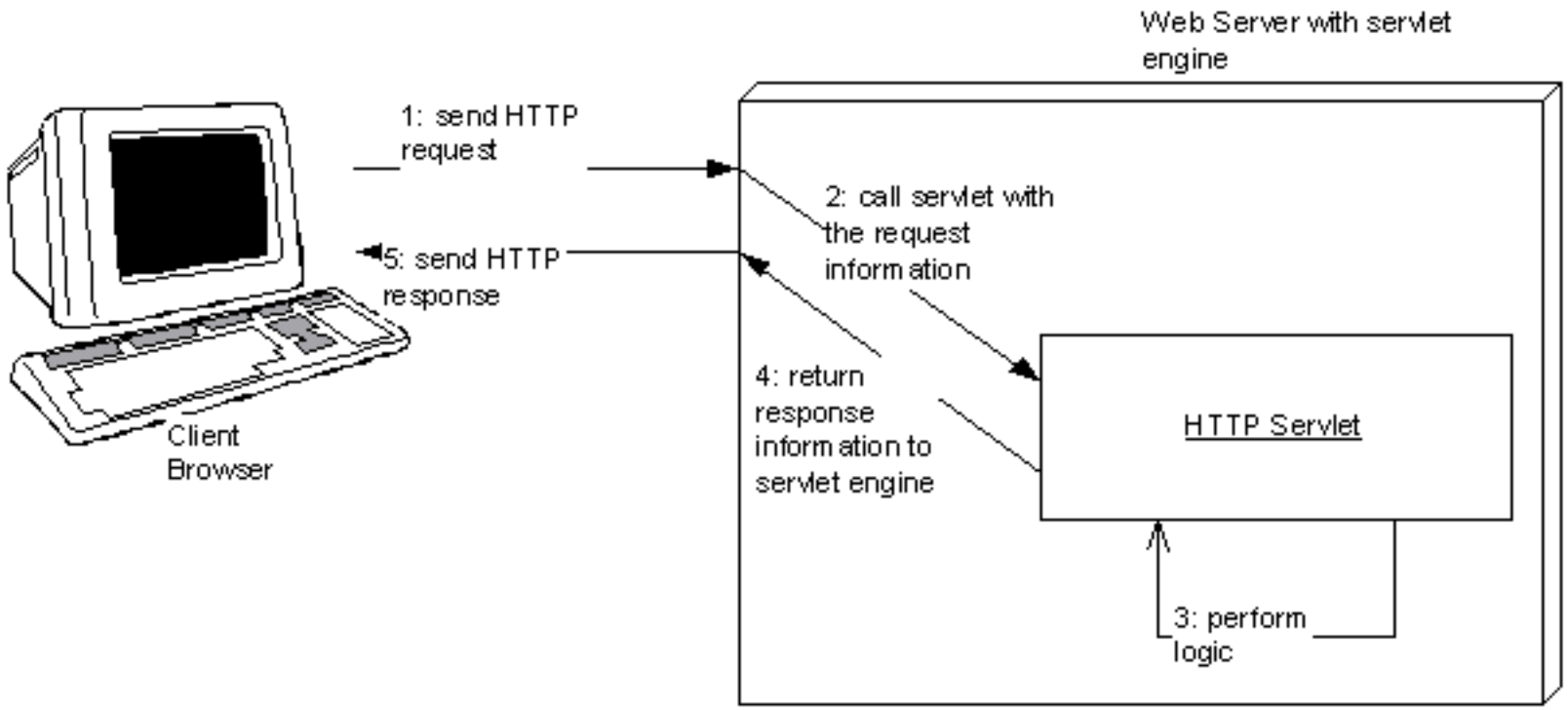
Implementing TheServerSide.com

- Technologies
  - Servlets
  - Java Server Pages
  - Java DataBase Connectivity (JDBC)
  - Enterprise Java Beans
- Development Environment
  - Visual Age for Java
  - VI

Servlets

- Servlets
  - Plain Java classes that implement a particular interface
  - The Java alternative to CGI
  - Enables dynamic websites
  - Acts on Web requests and provides a response
    - Response is typically a generated web page.

Servlets

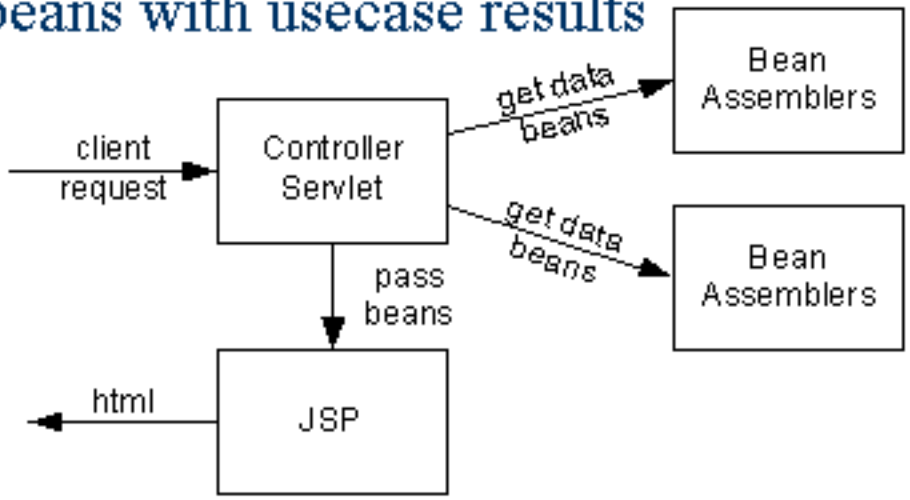


Java Server Pages

- plain html files with either embedded java code or jsp specific, XML-like tags.
- Used with servlets
  - business logic embedded in the servlets
  - JSP contains only plain HTML and JSP-tags that render pregenerated content.
  - Advantage: presentation logic is not mixed with the presentation view
  - An html designer can work on the JSP page while the programmers work on the java code in the servlets.

Servlet / JSP portal architecture

- Single Servlet Usecase Controller
  - Later known as Presentation Controller (in J2EE blue prints)
- One Controller Servlet receives all web requests
- Many java objects called assemblers.
  - Assemblers encapsulate presentation logic of a particular usecase
- Servlet delegates all requests to particular assemblers
- Assembler "assembles" beans with usecase results
- Controller passes beans to a jsp.

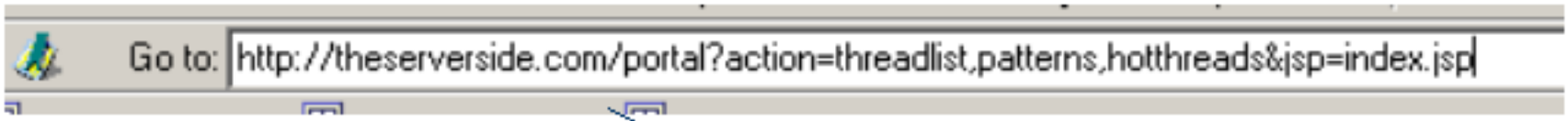


Single Servlet Usecase Controller

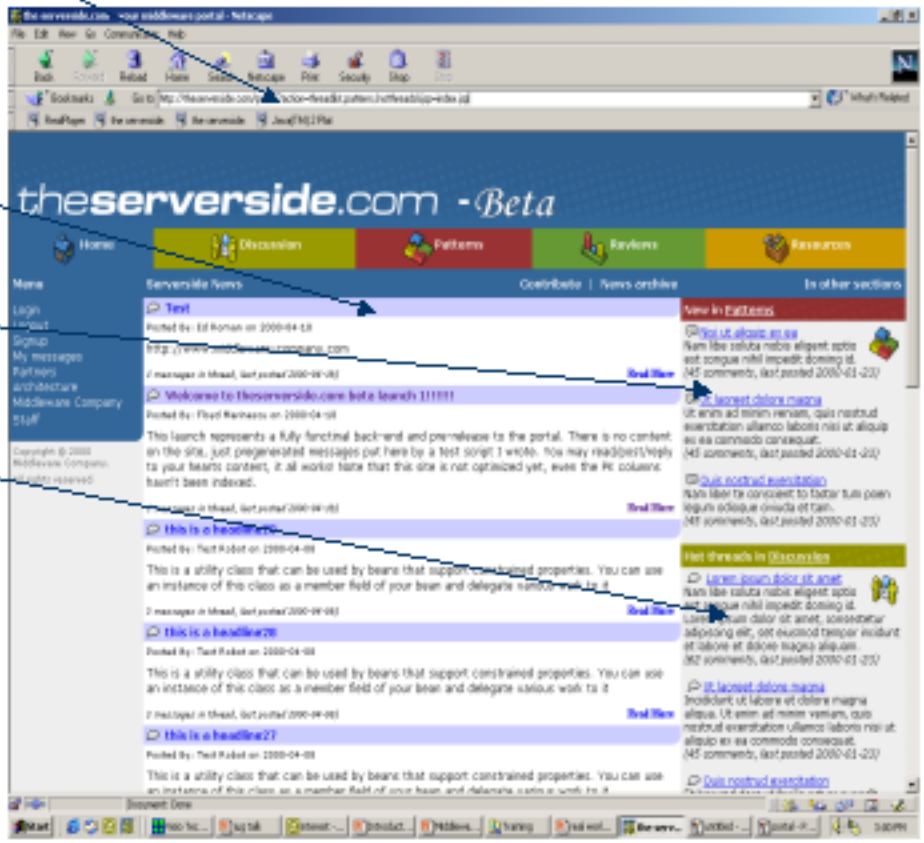
1. One controller servlet performs reflection on the "assembler" parameter in a web request
   http://www.theserverside.com/portal?assembler=ThreadList&jsp=forum.jsp
2. Take the returned beans and add it to the servlet Request Object.
3. Forward the request to the JSP specified in the web request

- Advantages
  - One servlet, less maintenance.
  - Usecases map directly to assemblers
  - Single point of entry for entire application
  - Business logic is totally hidden from view (JSPs)
  - Multiple JSP's can be a view to the same usecase (assembler)
- Disadvatanges
  - Views are not "encapsulated", links in site must be hardcoded with specific assemblers
  - All page resources (images, files) must be referenced with absolute paths

theserverside.com



- Three usecases on this page
  - Our news list(thread list)
  - New patterns
  - Hot threads



Single Servlet Usecase Controller

- A Better variation
  - Put controller calls in JSPs
  - Clients now request the JSPs, which internally query the controller servlet for "components" via JSP:INCLUDES
  - Allows building web-pages component-wise

Eg: embedding:

```
<jsp:include page=/portal?assembler=listOfThreads&jsp=threads.jsp…" />
```

Will show a thread list at this point in the JSP page. This line of JSP has made the list of threads into a re-useable component.

Enterprise Java Beans (EJB)

- Plain java classes that implement the EJB interfaces.
- Implement the business processes and the domain model for your application.
- EJB's make it easy to write applications that can scale up to thousands of concurrent users (eBay, etc).
- EJB is standard API and programming model not an implementation.
  - This differs from Microsofts Windows DNA, which is a technology.
  - EJB's are currently supported by 30+vendors, including IBM, Oracle, Sun, Hewlett-Packard, Netscape, etc (all but Micro$oft).

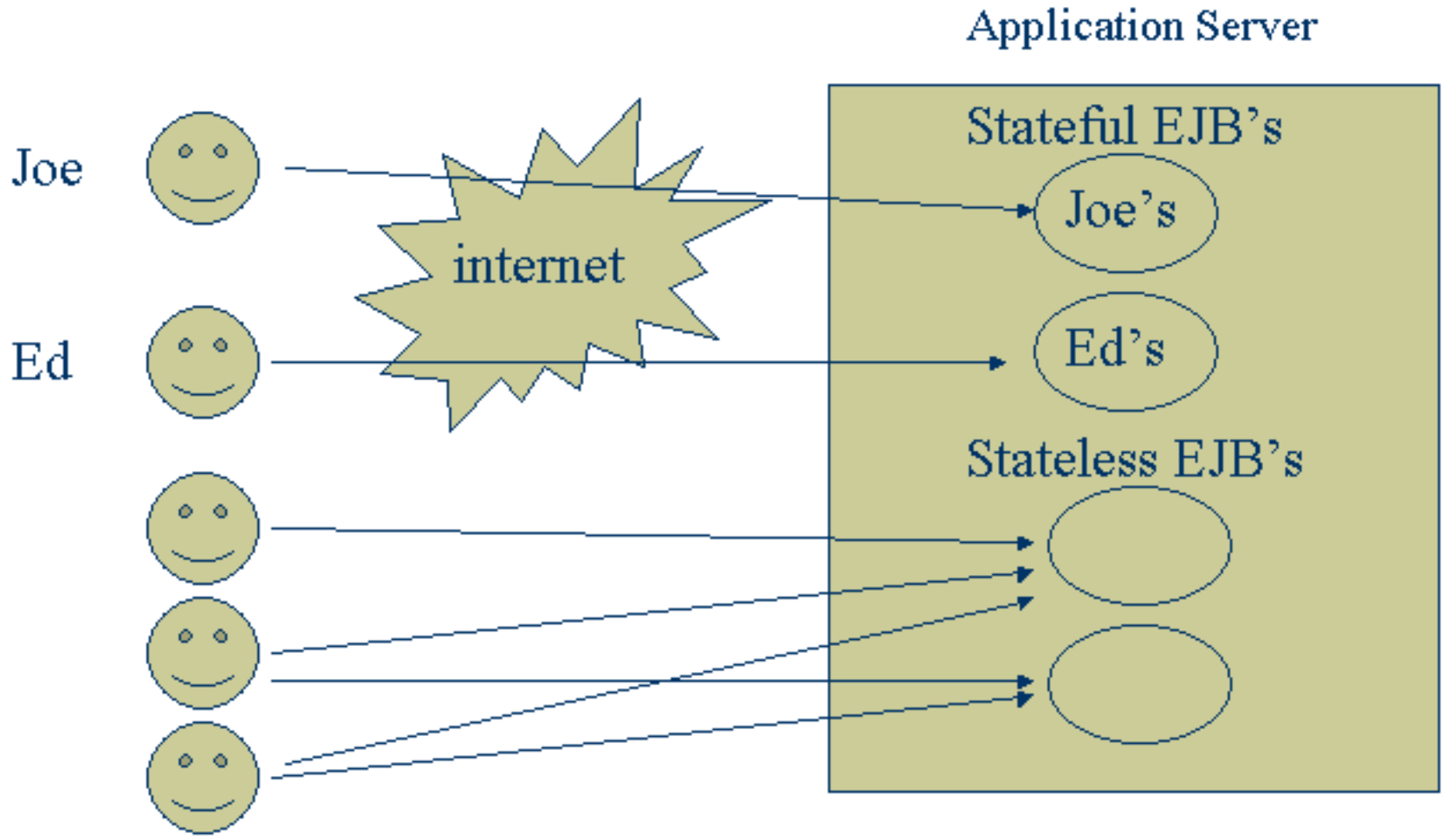
Enterprise Java Beans

- Two types of EJBs:
  - Session Beans
    - Implement business processes and workflow of your app.
      - Eg: the "process" of placing a bid on an item is a usecase that would be implemented as normal method on a session bean.
  - Entity Beans
    - Provide the domain model for your application.
      - Eg: An item, user, bid, etc are data objects that model the real world.
    - Used by Session beans
      - Eg: when placing a bid, you create a "bid" entity bean and may add it to a list of bids for an "item" entity bean.

Session beans

- Hold business processes
- Two types
  - Stateful
    - Hold per-client state
    - One stateful EJB for each concurrent user of your app.
    - Eg: shopping carts.
      - Each user needs their own shopping cart
  - Stateless
    - No per-client state
    - Typically hold information retrieval
      - Eg: listing products at Amazon.com
        - Does not require any information about the client

Session beans



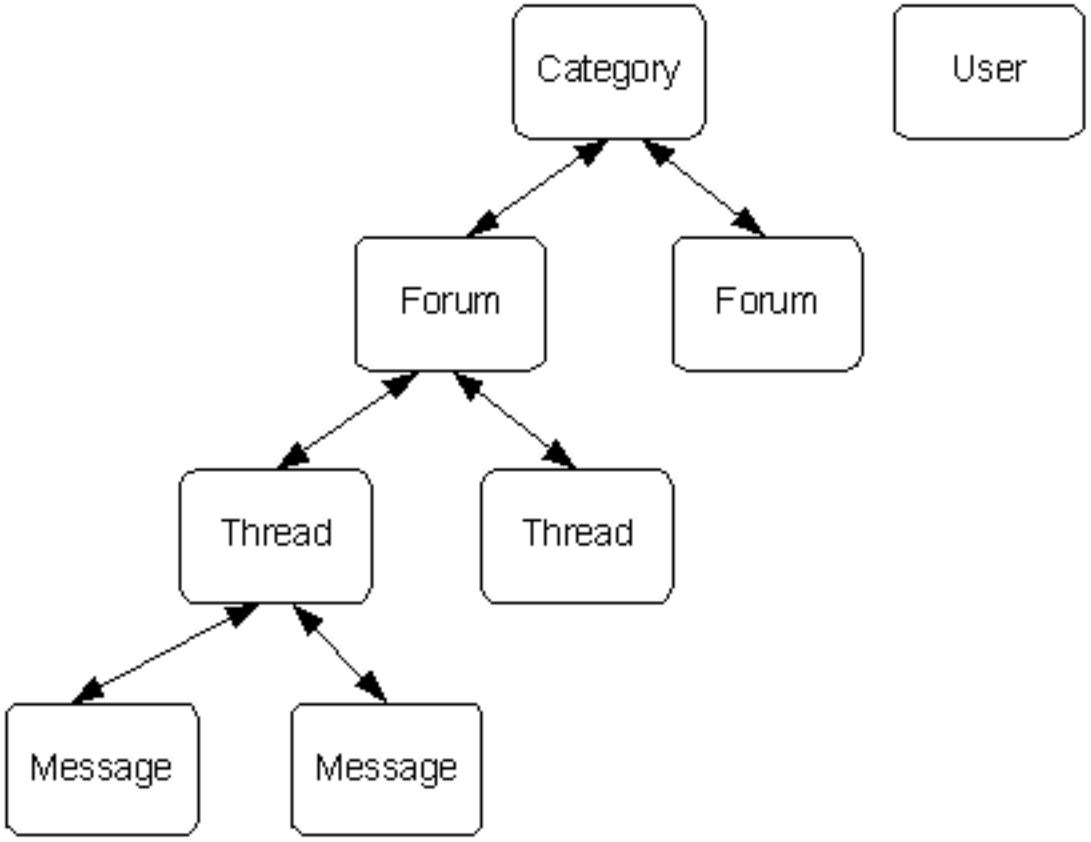
Session beans on *theserverside.com*

- Two session beans on our portal
  - One for stateful operations
    - Stores user's login status, and any usecases that require user login
      - Posting a Message – need to check if logged in
      - Login – sets login flag in the EJB
      - Logout – clears login flag in the EJB
  - One for stateless operations
    - All querying operations
      - Retrieving threads in a forum

Domain model persistence

- Entity beans
  - encapsulate the persistence mechanism
    - Work with objects in your workflow, not sql.
  - Two types of persistence
    - Bean Managed Persistence (BMP)
      - You write all the database mapping code with JDBC
    - Container Managed Persistence (CMP)
      - Your app. Server handles the persistence transparently

Entity Beans on TheServerSide.com



Wrap Entity Beans with Session Beans

- Clients only call session beans, entity beans hidden behind session bean workflow.
- Advantages
  - Reduces network overhead
    - Each method call requires a network round-trip
    - Eg: 5 "getter" methods on an entity bean wrapped by one session bean method call
  - Encourages cleaner separation of presentation logic from business logic
    - Servlets just validate and pass beans, domain model is encapsulated by methods on a session bean

Details Object

- Marshall entity bean state into a plain java object (details bean) which
- Details object contains all entity bean attributes, + getters/setters
- Clients request the details bean to "get" entity bean state
- Advantages
  - Reduce network round-trips
    - Eg: perform "getters" on the local details object instead of the entity bean
  - Details object re-useable
    - details objects and be used in JSP's

Encapsulate validation logic in Details objects

- Problem: duplicated validation logic in servlets and EJB's.
  - Eg: User information on signup form
    - Duplicated in servlet form validation and User EJB.
- Sol'n: Place validation logic in "setters" on the details object.
  - Use details object to validate in servlets and EJB.
- Advantages
  - Cleaner code, less coupling / spaghetti code

Inherit from details objects

- Entity beans extend their detail objects
- Advantages
  - Cleaner code
    - Attributes and getters/setters moved out of the entity bean

isModified

- Problem: App. Servers will store entity bean state to database after every method call
  - If entity bean has not been modified, this database call is not necessary
- Solution: implement an isModified flag
  - Set isModified to true in methods that modify state
  - Place a isModified check at the top of ejbStore

isDBShared

- Problem: EJB's will reload state before every transaction, even if the underlying DB state hasn't changed.
  - Since multiple apps. could modify the DB.
- Solution
  - Weblogic Specific "isDBShared" deployment flag
    - Guarantee that only WL will be modifying the DB.
- Result
  - Beans are loaded ONCE upon activation
  - Along with isModified, we have created an in-memory cache of entity beans! FAST!!!!

Entity Bean Primary Key Generator

- **Problem**
  - How do you generate unique ID's for your entity beans in a simple, portable, high performing fashion?
- **Solution**
  - Have one Entity Bean generate primary keys for all your other entity beans, by simply incrementing a counter.
  - Since your "PK generator" is an entity bean, the current count will persist across system crashes and across VM's.
- **Implementation**
  - Make the PK of the PKGenerator a string, so each entity bean using it can "find" it by name.
    - Eg: Message Bean will do a PKGeneratorHome.findByPrimaryKey("MessageBean")
    - This allows each entity bean to have its own incremental counter of PK's, rather than one counter for the whole app.

Listing behaviour strategies

- The debate:
  - Query operations via JDBC in session beans, or via the entity beans?
- List via JDBC in session beans
  - Result sets/rowsets returned to client
  - Advantages
    - No transactional overhead for simple query operations
    - Takes advantage of DB built in caching
    - Rowset provides a common interface for all query operations
    - Retrieve the exact columns you are interested in
      - Entity beans must load every column in the table

List via JDBC in session beans

- Disadvantages
  - Violates persistence mechanism encapsulation
  - Not object oriented, violates entity bean bean layer
  - Less maintainable – copied sql code
  - No compile-time checking of query results
    - Bugs can easily arise without objects

Query operations via entity beans

- Use EJB finders and return details objects to clients
- Advantages
  - More maintainable
  - More encapsulated
  - Faster than the alternative if entity beans are cached
  - Compile time checking in client tier

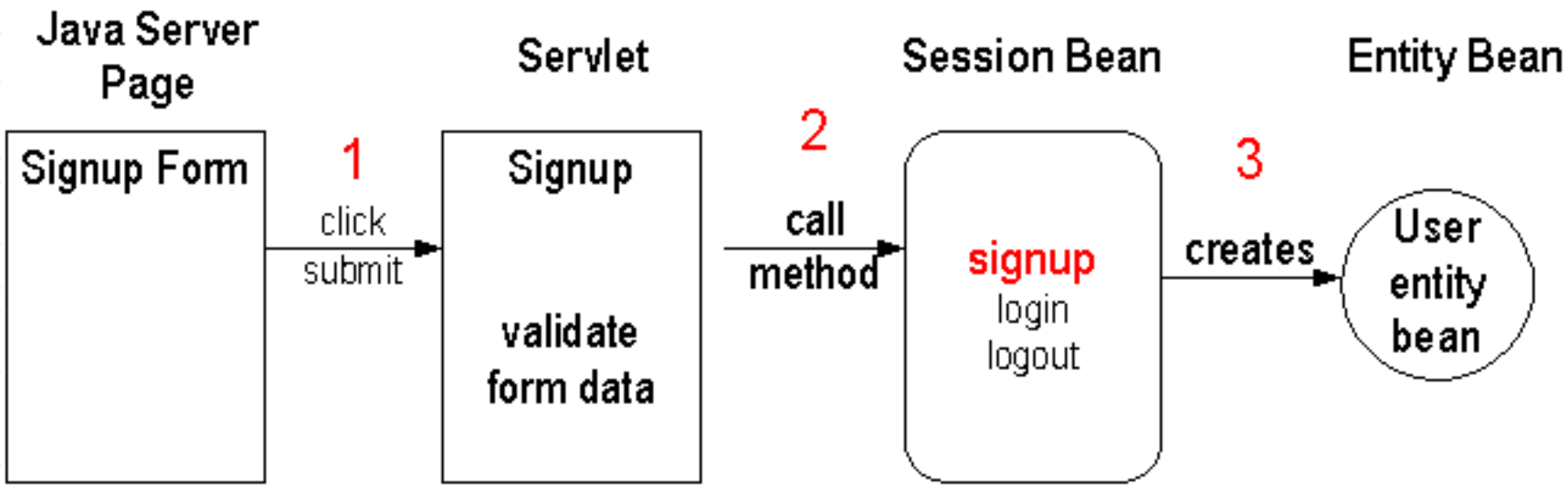
Query operations via entity beans

- Disadvantages
  - Transactional overhead with simple read-only queries
  - More awkward to perform joins

Usecase revisited

- You have designed your usecases, how do we translate this to a J2EE architecture?
  - Usecase divided into two parts
    - Presentation logic
      - Can map to one servlet and several JSP's
    - Business logic
      - Maps to one method on a session bean and many entity beans

A Signup usecase implementation



Deployment, Scalability Testing and Bugs

info@middlewarecompany.com / http://www.middlewarecompany.com

Deployment

- Initial Deployment:
  - Pentium 300, 256 megs RAM
  - Weblogic 4.51
    - Webserver, Servlet Engine, EJB Container
  - PostGreSQL 6.5.3
    - Data Base
  - Redhat Linux 6.0

Scalability Testing

- Test: Populated DB with 40,000 users and 16,000 messages
- Result
  - Front page went from 2 seconds to 12 seconds to load – single user. ☺
  - Solution 1
    - Throw more hardware at it.
    - New configuration: PIII 650, 512 megs RAM
  - Did it work? Slightly, brought response time down to 8 seconds.

Scalability Testing

- Actual problem:
  - How did we find it?
  - Bug with PostGreSQL
    - PostGres wasn't using indexes on int8 primary key columns (even though indexes were created).
    - Primary Keys were int8 due to use of System.getCurrentMillis to generate keys. ☺
- Solution 2 Upgrade to PostGres 7
  - Didn't work
- Solution 3 use "incrementing entity bean" to generate PK's.
  - PK's were small enough to use int4 columns, which PostGreSQL could index properly
- Result
  - Single User Response time brought down to about 3 – 4 seconds.

Load Testing

- WebLoad Eval
  - Give it a url, it will simulate 12 concurrent users
- Result
  - Not scalable to 12 concurrent users (wait time of 15 to 20 seconds).
    - Other Symtoms:
      - Memory usage was at maximum
      - Unsure as to Webload's reliability (it said that cnn.com took 6 seconds to load).

Optimize-It to the Rescue!

- Optimize-it, the Java Profiler
- Remotely profiled theserverside.com
- Result:
  - 30% of cpu time was being spent in SocketCommunications with DB.
  - Due to design blunder
    - Call to DB to count number of messages in a thread upon every invocation of getThreadDetailsObject()
- Fix
  - Maintain message counts manually in Entity Beans
- Result
  - 12 concurrent users response time down to 1 – 2 seconds.

Bugs

- Mega Bug
  - When an IE user posts a long message, next screen would always be "Page Cannot Be Displayed" white screen of death.
  - Spent days trying to debug this, trying everything from html verification to packet sniffing to praying.
- Cause of Bug
  - According to Weblogic tech support, a bug in the Linux kernel causes the server to Reset, rather than Close connections.
    - Their recommendations: use a different webserver or switch to Solaris ☺
- Solutions:
  - Tries RESIN open source servlet engine from caucho software
    - Bug still present! Wow it really was a Linux Bug!!
- Final solution
  - Used RESIN with its Apache Webserver plugin (WL doesn't have an Apache Plugin on Linux yet).

TheServerSide.com

- Final configuration of TheServerSide.com
  - PIII 650, 512 megs RAM
  - Weblogic EJB Container
  - RESIN Servlet Engine
  - Apache Web Server
  - PostGreSQL 7.0
  - Redhat 6.0

Theserverside.com

Come join the J2EE community!

info@middlewarecompany.com / http://www.middlewarecompany.com

Would you like to write your own Portal?

- The Middleware Company takes smart, ambitious people and turns them into experts by:
  - Encouraging employees to:
    - speak at conferences
    - write magazine articles
    - work with latest technologies
- www.middlewarecompany.com

For more information

- Jump start your project with expert training!
  - The Middleware Company provides onsite training from design to proof-of-concept
  - www.middlewarecompany.com
- Theserverside.com – your J2EE resource
  - www.theserverside.com
- EJB-INTEREST listserv
  - http://archives.java.sun.com