What is J2EE?

- J2EE was designed for Enterprise Applications... *not for general purpose*
  - it is complex because EA are complex
  - for many applications it is an overkill
  - EA can benefit a lot from using J2EE
- In order to appreciate J2EE and use it properly it is important to understand what are EA

What are Enterprise Apps?

- These are the applications that drive banks, insurance companies, online stores, market places, B2B, B2C, Supply chain, ...

They’re what makes the world go round...

What is J2EE?

- J2EE is a framework for writing enterprise applications
- An industry standard coordinated by Sun Microsystems together with industry leaders – IBM, Oracle, BEA, SAP, Compaq, ... (JCP see JSR151)

What are Enterprise Apps?

- EA can be very different from one another, however they have common characteristics (not all exhibit all characteristics)
- Think about the EA behind

...
EA characteristics
High volume, scalable, distributed
• Some EA serve thousands of users concurrently
• For new systems you would like a scalable architecture that scale with the business

EA characteristics
Transactional
• The core of EA is processing transactions
  – They must be able to process transactions consistently and efficiently
  – Often distributed transactions involving different databases and other type of resources

Scalability Exemplified
• eBay (Data from 2002)
  – 69 Million Registered users
  – 380 Million dynamic hits / day! (assume 12h ⇒ >8,000 per second)
  – System need to scale with business:

Transaction Example
• Consider a reservation of tickets for a show
• After to specify the details of the reservation, the following tasks should be performed:
  – A place should be reserved for you
  – You should be charged for the reservation
  – A ticket (receipt) should be issued to you
• All these tasks must succeed or not occur at all

EA characteristics
Heterogeneous
• An EA is rarely a single software designed by a common team
  – It is a collection of interoperating software systems designed over years by deferent teams with different technologies
  – New components are added due to merger/acquisitions, business trends, introduction of new technologies...

Transaction Example
• Checkout
  – Reserve chair otherwise, customer won’t find a place
  – Process CC otherwise, theater owner unhappy
  – Send receipt otherwise, customer won’t be let in
EA characteristics

Robust, non-stop, fault-tolerant, secure
- Often should be available 24/7
- Automatic recovery from crashes – fast, no data loss or integrity loss
- Manage sensitive data, different levels of access, adhere to federal regulations

J2EE Container

- In order to attack these challenges J2EE introduces the concept of a J2EE Container
- The ‘application’ consist of a set of components that live inside a container
- The container itself is distributed and can span several machines

Direct Downtime Costs (hour)

<table>
<thead>
<tr>
<th>Source: InternetWeek 4/3/2000</th>
<th>Est. Loss of Millions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Brokerage operations</td>
<td>$6,450,000</td>
</tr>
<tr>
<td>Credit card authorization</td>
<td>$2,600,000</td>
</tr>
<tr>
<td>Ebay</td>
<td>$225,000</td>
</tr>
<tr>
<td>Amazon, 22 hour outage, June 1999 =&gt; $180,000</td>
<td></td>
</tr>
<tr>
<td>Package shipping</td>
<td>$350,000</td>
</tr>
<tr>
<td>Home shopping channel</td>
<td>$113,000</td>
</tr>
<tr>
<td>Catalog sales center</td>
<td>$90,000</td>
</tr>
<tr>
<td>Airline reservation center</td>
<td>$89,000</td>
</tr>
<tr>
<td>Cellular service activation</td>
<td>$41,000</td>
</tr>
<tr>
<td>On-line network fees</td>
<td>$25,000</td>
</tr>
<tr>
<td>ATM service fees</td>
<td>$14,000</td>
</tr>
</tbody>
</table>

Separation of concerns

- The container allows for separation of concerns:
  - the container provide a set of services such as transaction management, distributed communication, etc.
  - the components use these services in a manner that allows them to concentrate on the business logic they perform

Example – online catalog

- Suppose your application is an online store and you want to implement a product catalog
- Many users access your catalog concurrently each issuing a query
- Consider the implementation of such a catalog...

1 To simplify matters we assume that the data in the catalog doesn’t change
Example – online catalog

• For good response you must process multiple request concurrently
  – this requires multithreading
  – each request processing requires resources (socket, db connection), you need to limit the number of concurrent request processing to avoid server crash
  – you want to cache results for similar requests

Example – online catalog

• You want to achieve scale with several machines
  – you need to balance the load between the machines
  – if a machine fails (crash, network..) you would like to rerun the requests it was handling on a different machine

Example – online catalog

• With J2EE you write a component with a single method that process a single request. The Container will:
  – manage several copies of your component on each machine
  – load balance request processing
  – rerun requests in case a machine fails
  – pool database connections and bound number of threads/sockets...

Container Services

• Concurrency, load balancing, failover, resource pooling
• Transactions
• Persistence
• Distributed Objects
• Messaging
• Naming
• Security

Container – component contract

• In the previous example we used the load-balancing service of the container implicitly – there was no need for a load-balancing API
• However, we do need to denote to the container that the request processing should be subject to load-balancing, failover, etc.

Container component contract

• We denote our use of a service in several ways
  – Extending certain class/interface
  – Use annotations
  – Follow naming conventions
  – Use descriptors files
  – in some cases programmatically
Annotations

```java
@Entity
@Table(name="USERS")
public class User implements Serializable {
    private Long id;
    private String username;
    private String password;

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    public Long getId() {
        return id;
    }
}
```

Application Servers

- In the same way the JVM provide an abstraction for Java that allows it to be implemented on different platforms, the J2EE Container provide an abstraction that allows different application servers to host J2EE applications.

```java
@Retention(RetentionPolicy.RUNTIME)
public @interface Monitored {
    String description();
    double lowerLimit() default 0;
    double upperLimit() default 0;
}
```

```java
public class MachineA {
    ...
    @Monitored(description="productionRate", lowerLimit=20, ...)
    public double getProductionRate() {
        return currentRate;
    }
}
```

Container component contract

- The rules that summarize the interaction between the component and the container (around the use of the container services) is referred to as the container component contract.

Common Application Servers

- BEA WebLogic
- IBM WebSphere
- JBoss
- Sun Microsystems SunOne
- Oracle
J2EE Specification

- J2EE is a standard defined by Sun Microsystems jointly with industry leaders
- The J2EE Specifications specify
  - container services
  - component types
  - included APIs
  - naming conventions
  - ...

Multi-tier architecture

- J2EE applications has a multi-tier architecture
- The system is divided into 3 or more layers
  - presentation layer
  - business logic layer(s)
  - enterprise information systems layer

.. in J2EE terminology
J2EE technologies

- J2EE include the following elements
  - Web technologies – Servlets, JSP, web-services
  - EJB (Enterprise Java Beans)
  - Additional APIs

Servlets & JSP

- Most EA have a Web-interface
- In J2EE Servlets & JSP are used for implementing Web-interfaces
- There are other technologies some build top of JSP and/or Servlets
  - Struts, Velocity, Tapestry, ...

Enterprise Java Beans

- EJB is a server-side component architecture targeted at business logic and persistence
- It defines components called Enterprise Beans
- There are several types of Enterprise Beans – Session Beans (stateless/stateful), Entity Beans, Message Driven Beans

J2EE APIs

- EJB
- JDBC
- Servlets & JSP
- JMS
- JNDI
- JTA
- Java IDL
- JavaMail & JAF
- JAXP
- JAX-RPC, JAXM
- JCA
- JAAS

Web Services

- Some EA expose some of their functionality to be accessed in an automated way to allow B2B or custom clients
- This can be done via web-services
  - e.g. Amazon, eBay, Google, ...
- An evolving technology, expected to play central role in the future

Sample applications

- We present two examples of J2EE applications to make the previous concepts more concrete
- Please bear in mind that these are sterile ‘classroom’ examples that cannot manifest the complex nature of most enterprise applications

-
Java Pet Store

Pet Store Design

Course Project