EMPOWERING THE DEVELOPER

How to Build an E-Commerce Application using *J2EE*™

Carol McDonald

Code Camp Engineer

Sun microsystems

Code Camp **Agenda**

- **J2EE™ & Blueprints**
  - Application Architecture and J2EE
  - Blueprints E-Commerce Application Design
- **Enterprise JavaBeans™ (EJB™)**
  - Designing and implementing the Business Model:
    - **Entity** Beans, **Stateless** and **Stateful Session** Beans
- **Servlets , Java ServerPages™ (JSP™)**
  - Designing and architecting the "View" or Presentation logic
- **Assembling and Deploying** the J2EE™ Application:
  - **Transactions** JTS™ and JTA™
    - Describe how the sample application uses the transaction capabilities of the J2EE™ platform to simplify component development.
  - **Security** How to use J2EE™ security to safeguard your e-commerce application

EMPOWERING THE DEVELOPER

Download Code Camps

- **All slides with notes are on your CD**
- **You can download this and other Code Camps :**
  - EJB, Java Performance, J2ME, JINI, JMS, JSP, XML

http://www.sun.com/developer/evangcentral

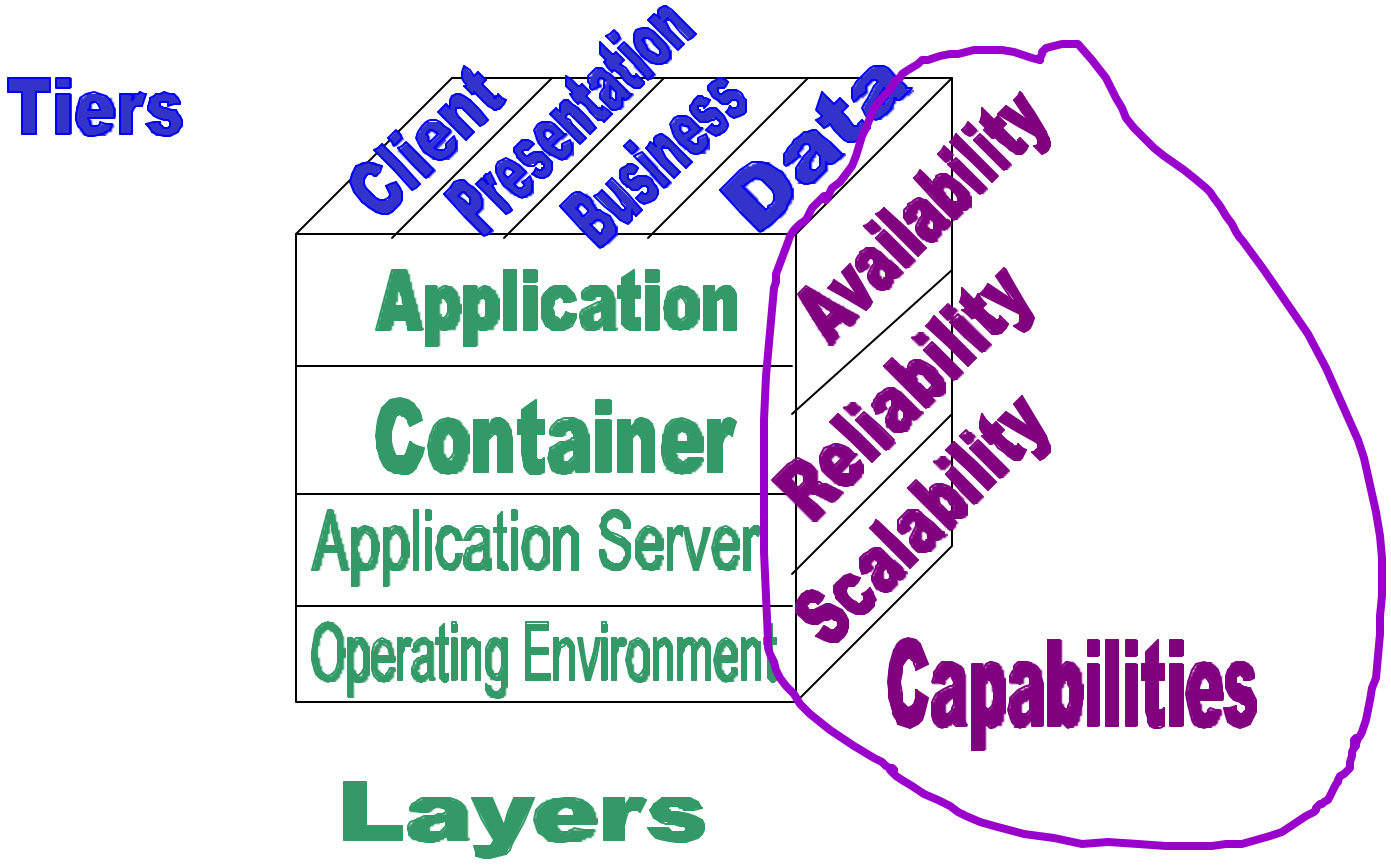
  - under Events: Developer Code Camps
- **ALSO**
  - You can also ask questions there
  - Listen to audiocasts

J2EE™ & Blueprints **Agenda**

- *Application Architecture and J2EE*
- Sun BluePrints™ Design Guidelines
- Architecture of the sample application

EMPOWERING
THE DEVELOPER

Architecture and the Cube



Architecting your application Using **Tiers** and **Layers affect** the application **Capabilities**

EMPOWERING THE DEVELOPER

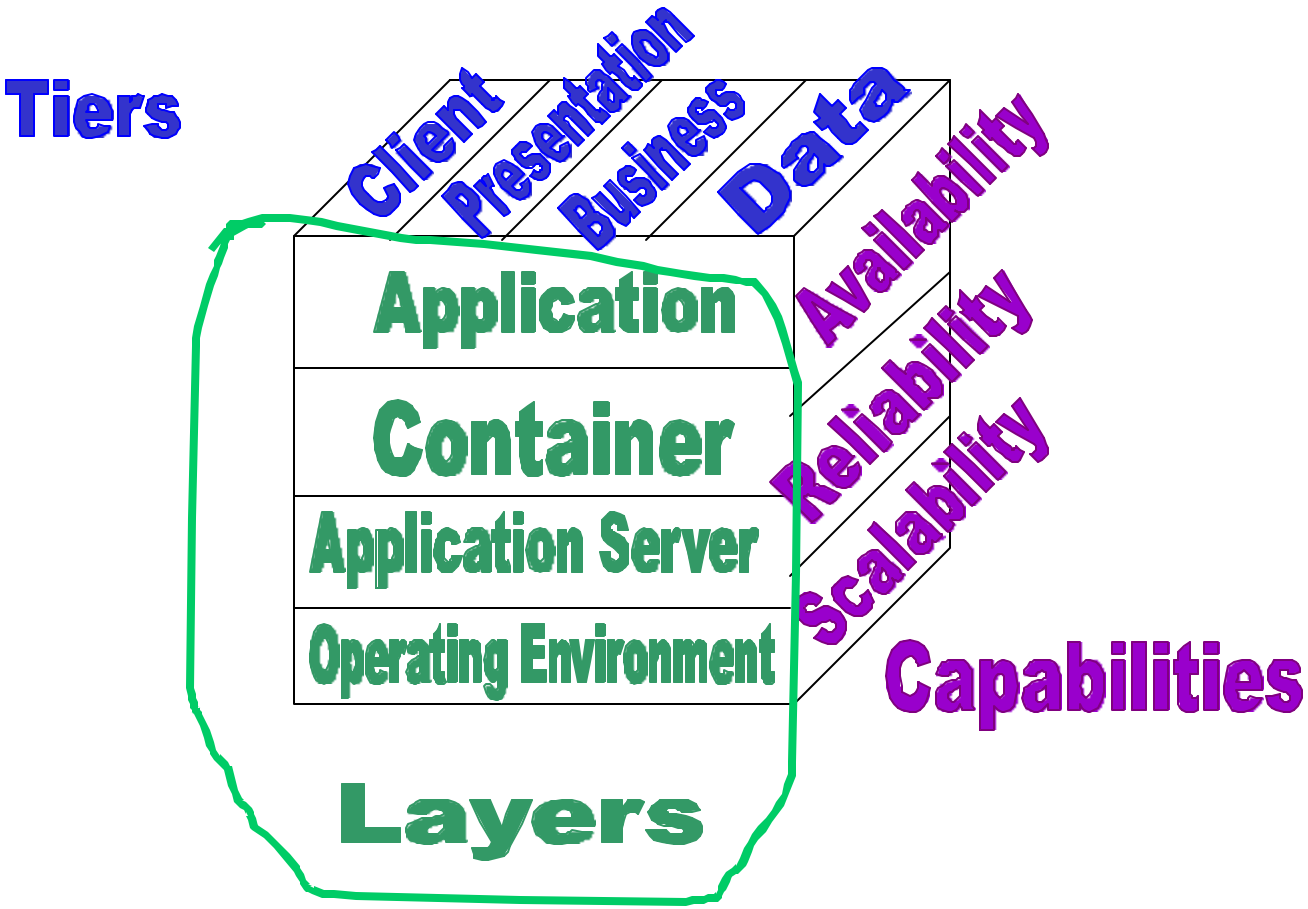
Capabilities = Non-Functional System Requirements

- **Availability**: service is **accessible**
- **Reliability**: **consistency** of application and transactions
- **Capacity**: can serve **# users**
- **Extensibility**: can **add functionality**
- **Flexibility**: can **support changes**
- **Performance**: good **response time**
- **Scalability: can** support **increased load**

Example Architecture Requirements

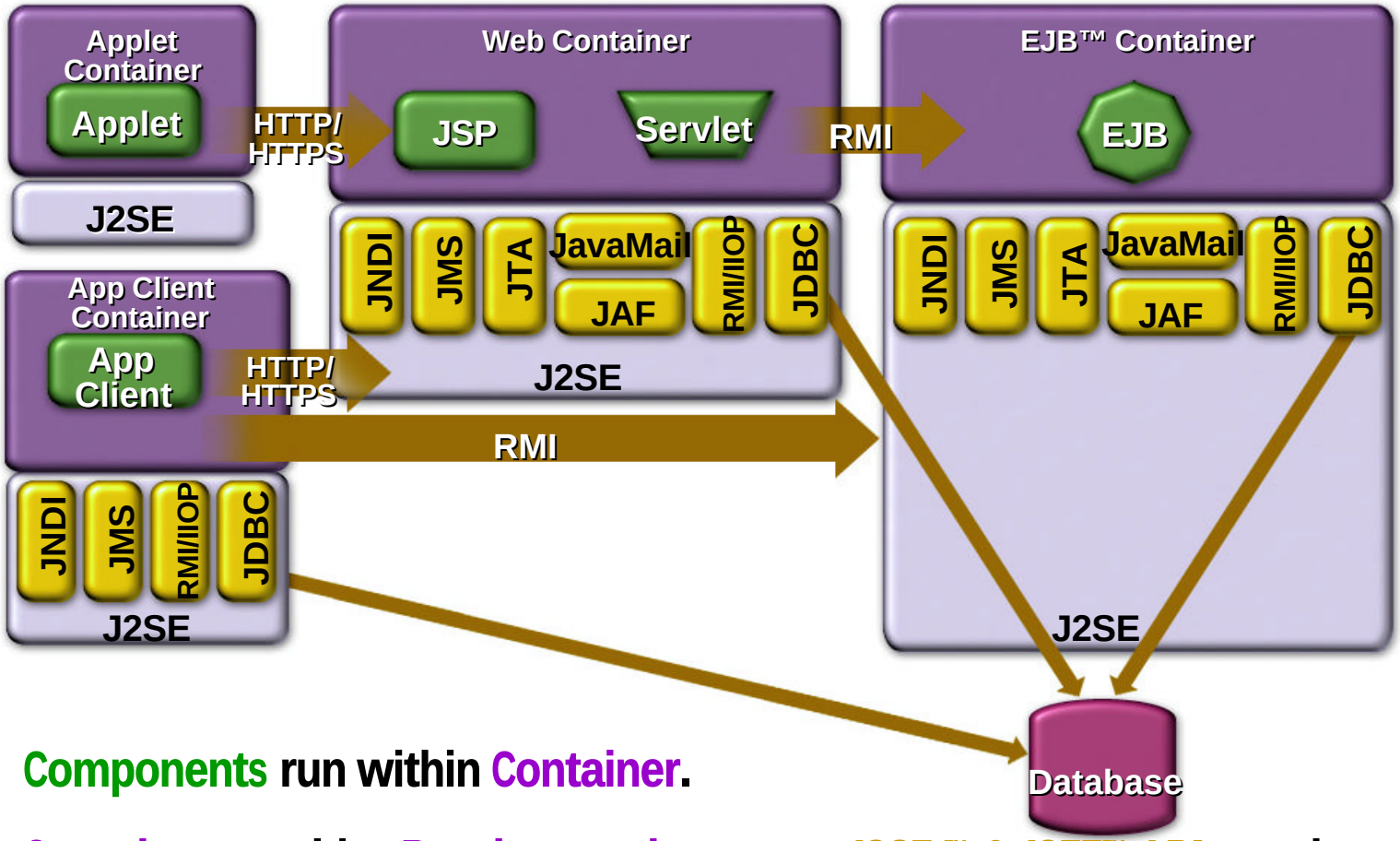
- **E-Commerce Apps:**
  - Many Clients,
  - **lots of reading**,
  - **less updating**,
  - low contention
  - **Need High performance and scalability for Reading data**
- **Insurance Intranet App:**
  - Fewer Clients, **Low concurrency**
  - High transaction isolation level, **Need Higher Consistency**
  - **Reliability of Updates** to DB **more important** than performance for Reading data

The Relationships between the Elements of Architecture



Architecting your application Using **Tiers** and **Layers affect** the application **Capabilities**

J2EE™ Platform Specification

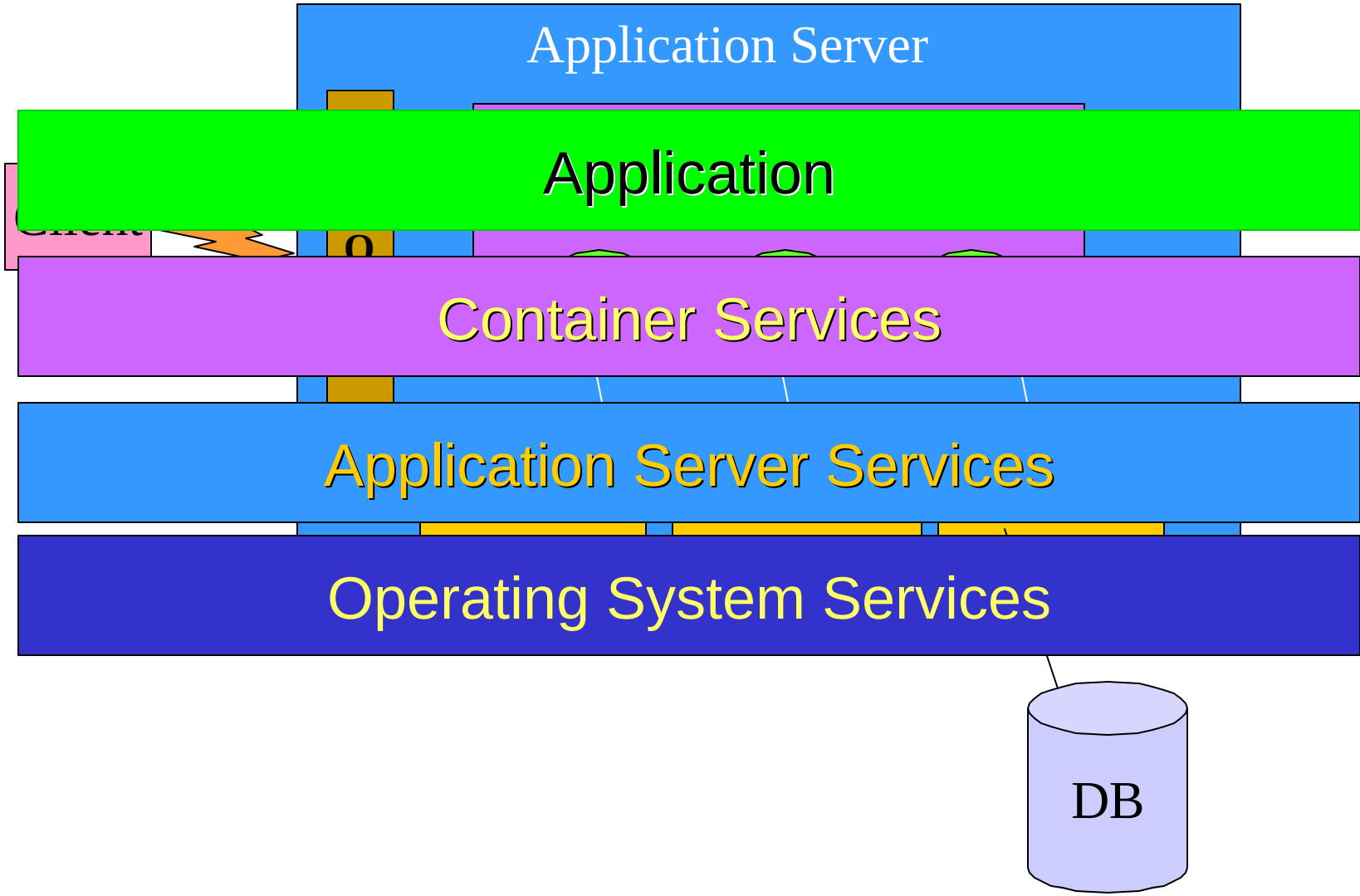


Components run within Container.

Container provides Runtime environment, J2SE ™ & J2EE™ APIs, and remote communication

EMPOWERING THE DEVELOPER

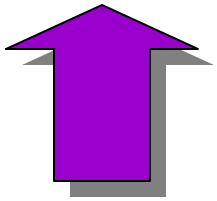
Layers of Services: The Application Server provides services to the Container, which in turn provides services to the EJB.



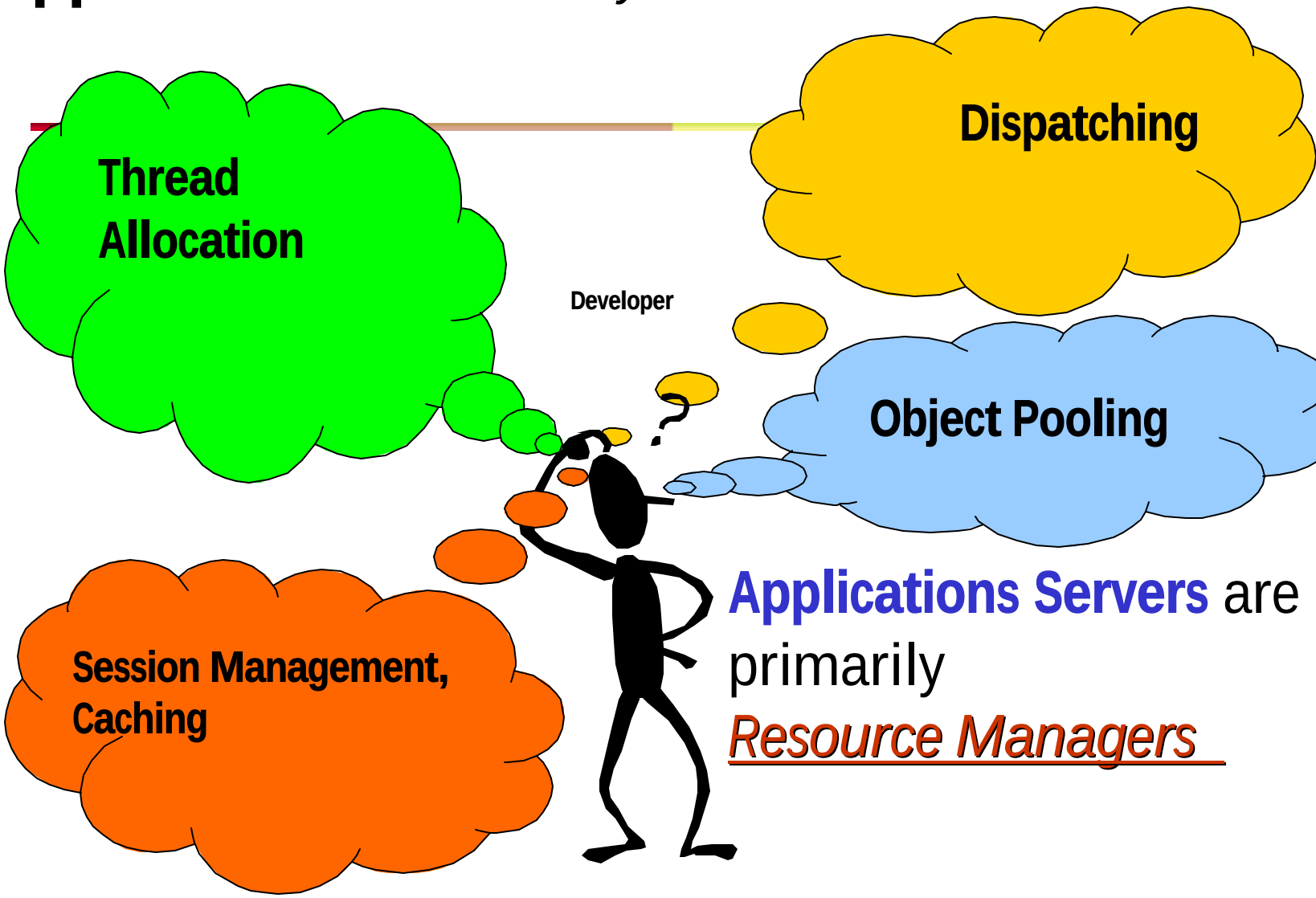
EMPOWERING THE DEVELOPER

Layers

- Layers are abstractions of underlying implementation
- Each Layer hides implementation details of layer below
- The Container provides components services of the layers below
- Advantages:
  - Easier development, maintenance for high performance, scalability
  - Does not require developer to know complex details of distributed protocols, concurrency, transactions, load balancing, security, resource pooling and sharing…



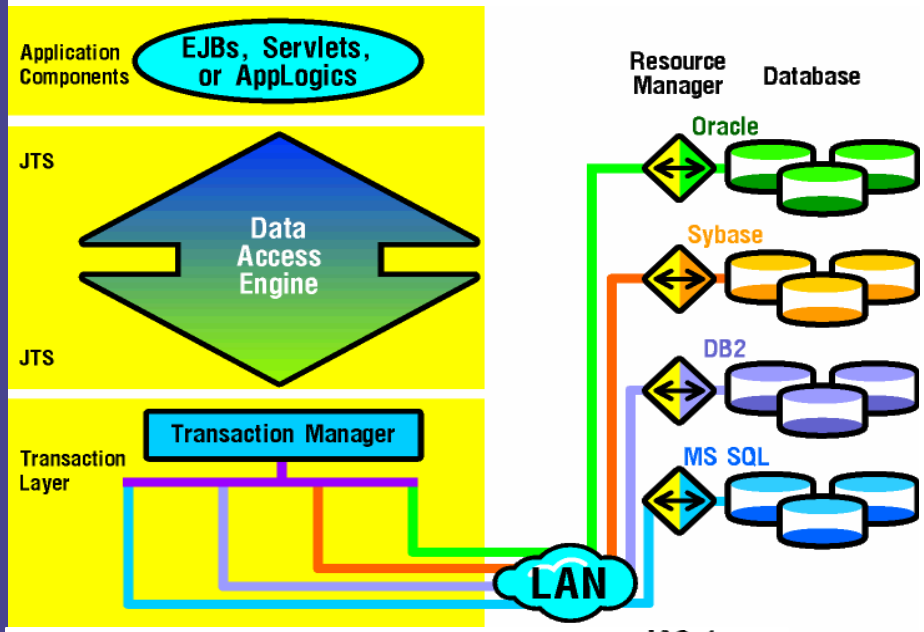
Application Server *: Key Services*

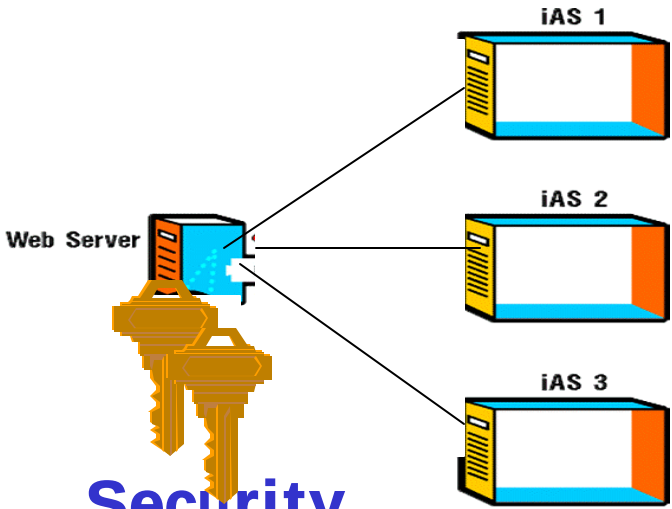


Applications Servers are primarily *Resource Managers*

Applications Servers Manage: dispatching, thread allocation, pools, caches, load balancing, clusters

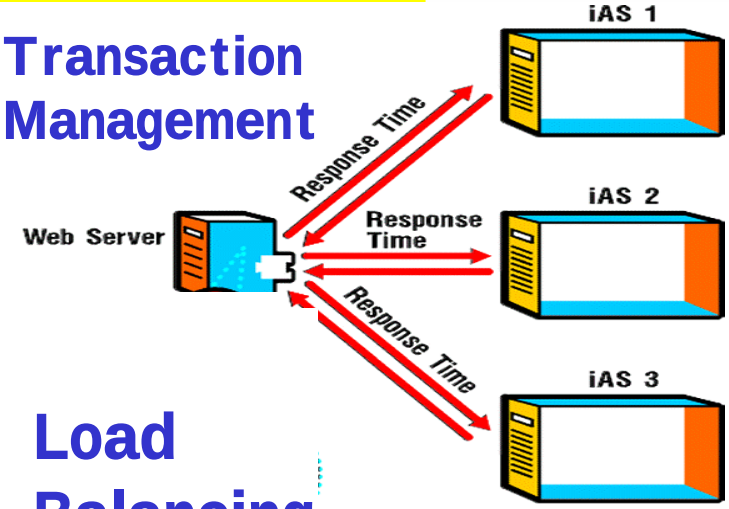
What features might you look for in an Application Server?



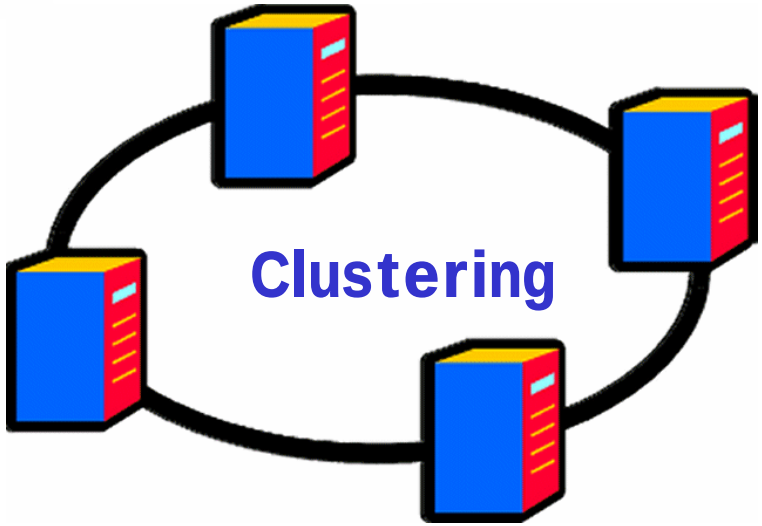


Security

Transaction Management



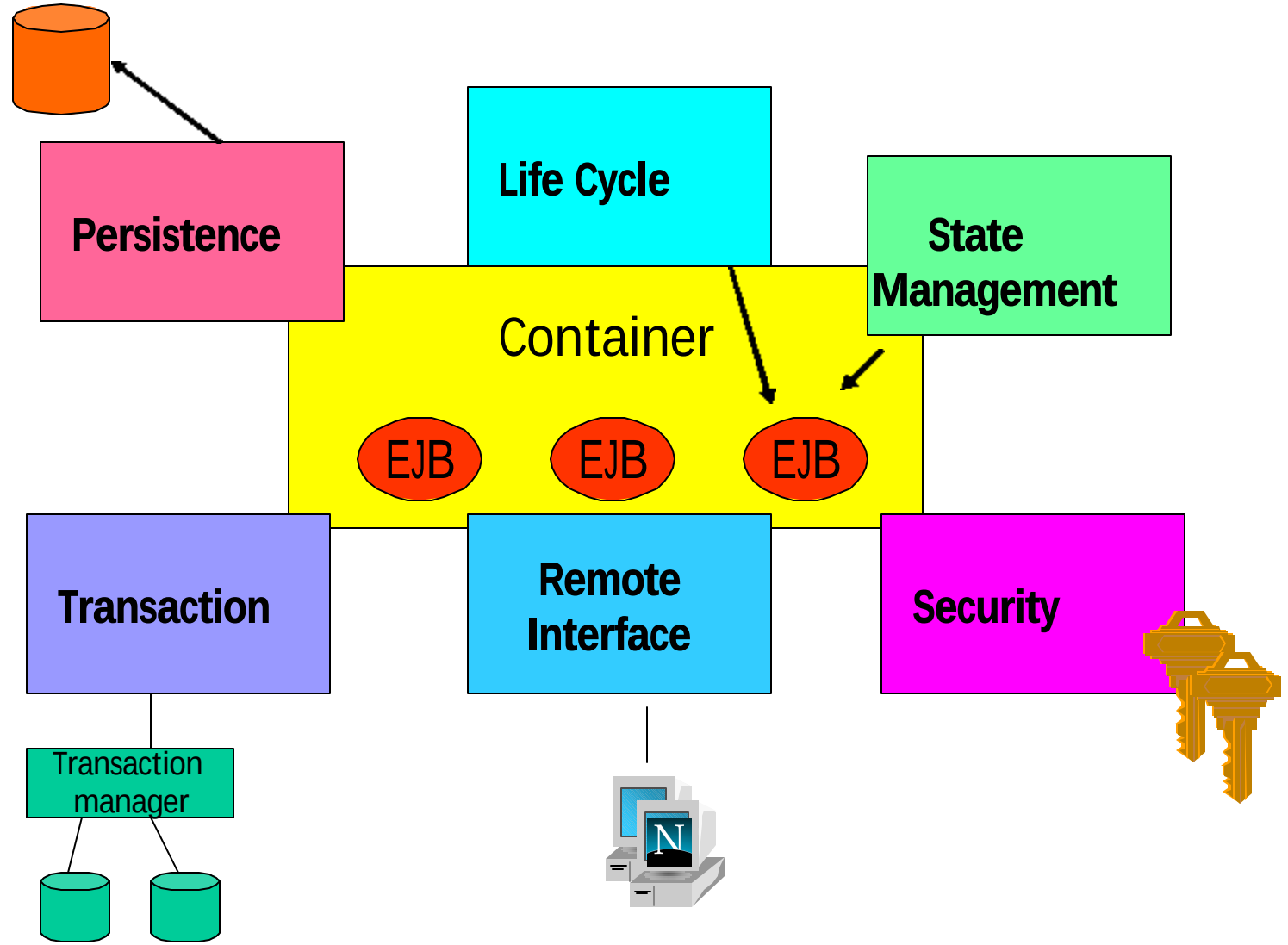
Load Balancing



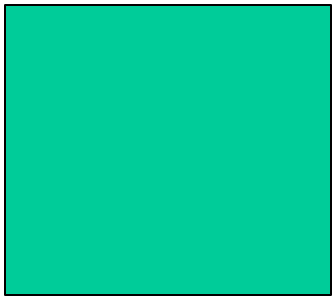
Clustering

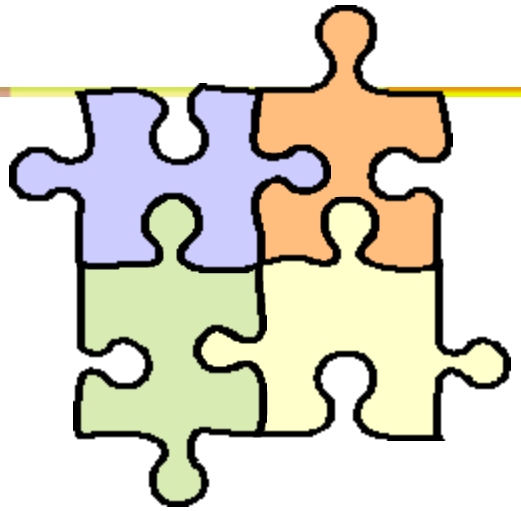
EMPOWERING THE DEVELOPER

EJB Container Services



Components





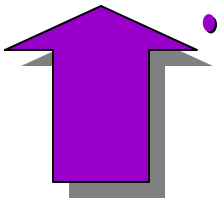
Monolithic:

1 binary file

Recompiled & linked for any changes

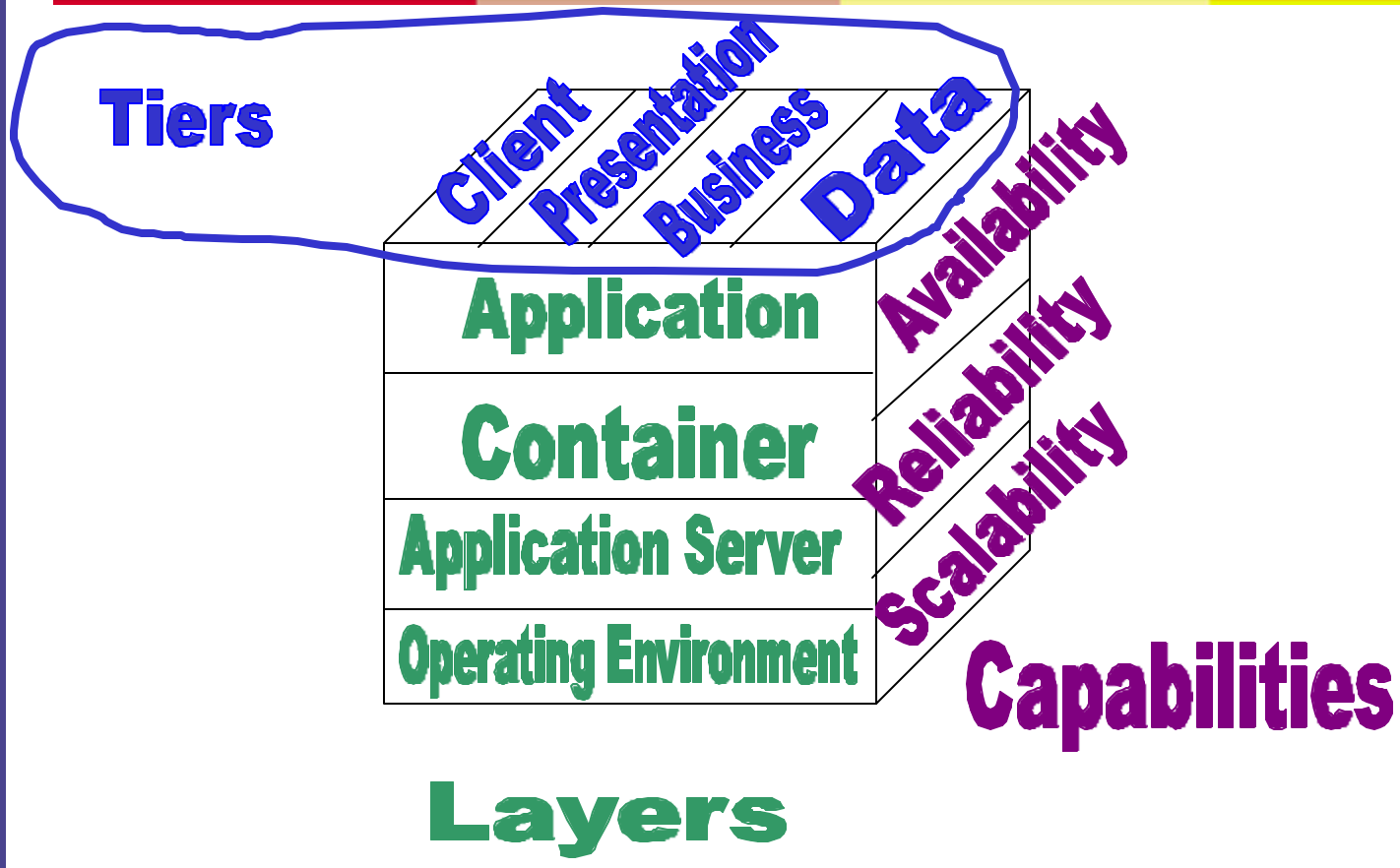
Components:

- **plug-gable Parts**.
- **Implementation separated from Interface**, only Interface published
- **hides implementation details**
- **Better design, easier updates**
- **Better Flexibility, Extensibility**

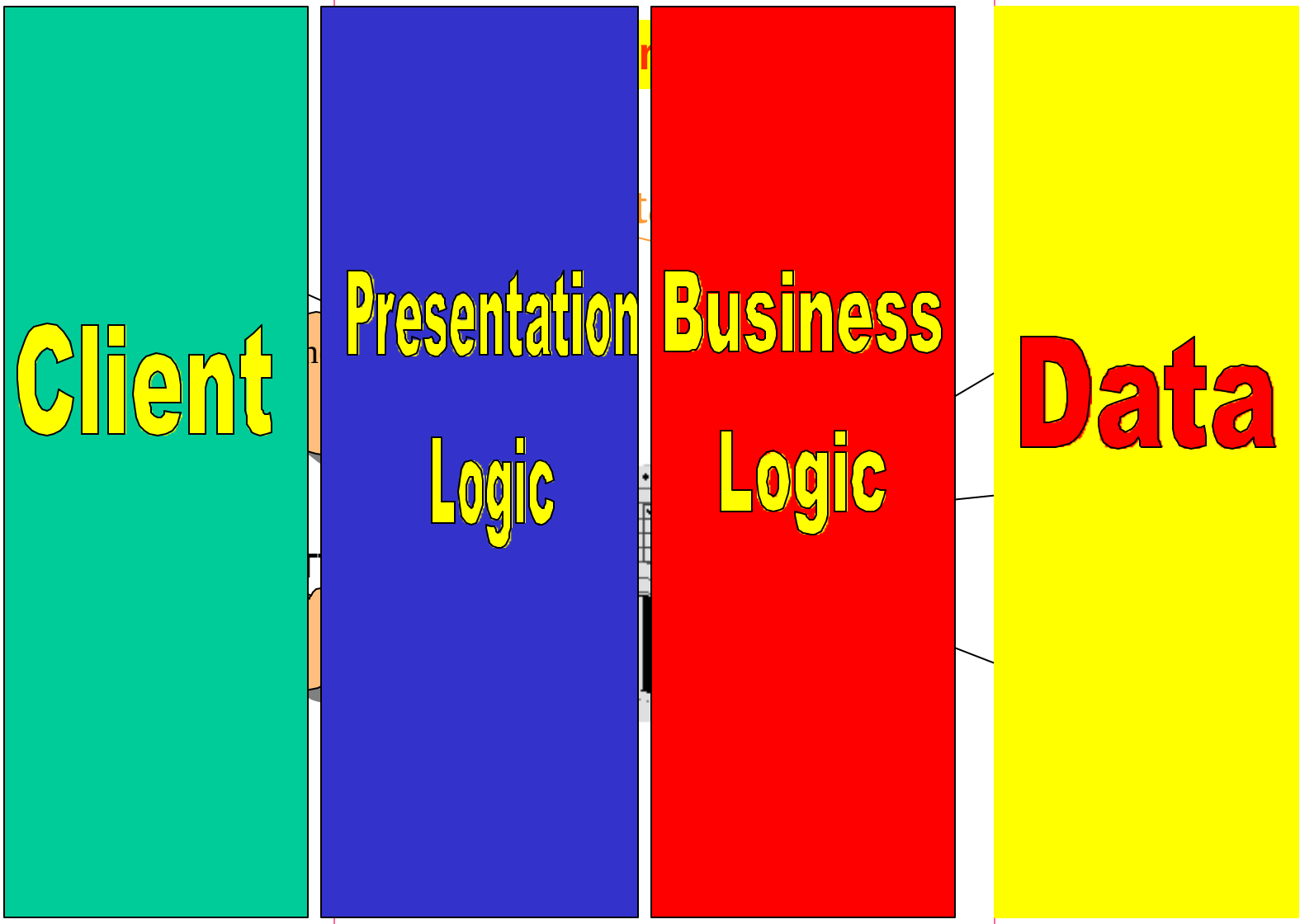


EMPOWERING THE DEVELOPER

Architecture and the Cube



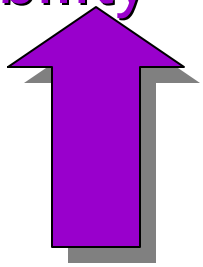
n-tier Architecture with J2EE™



EMPOWERING THE DEVELOPER

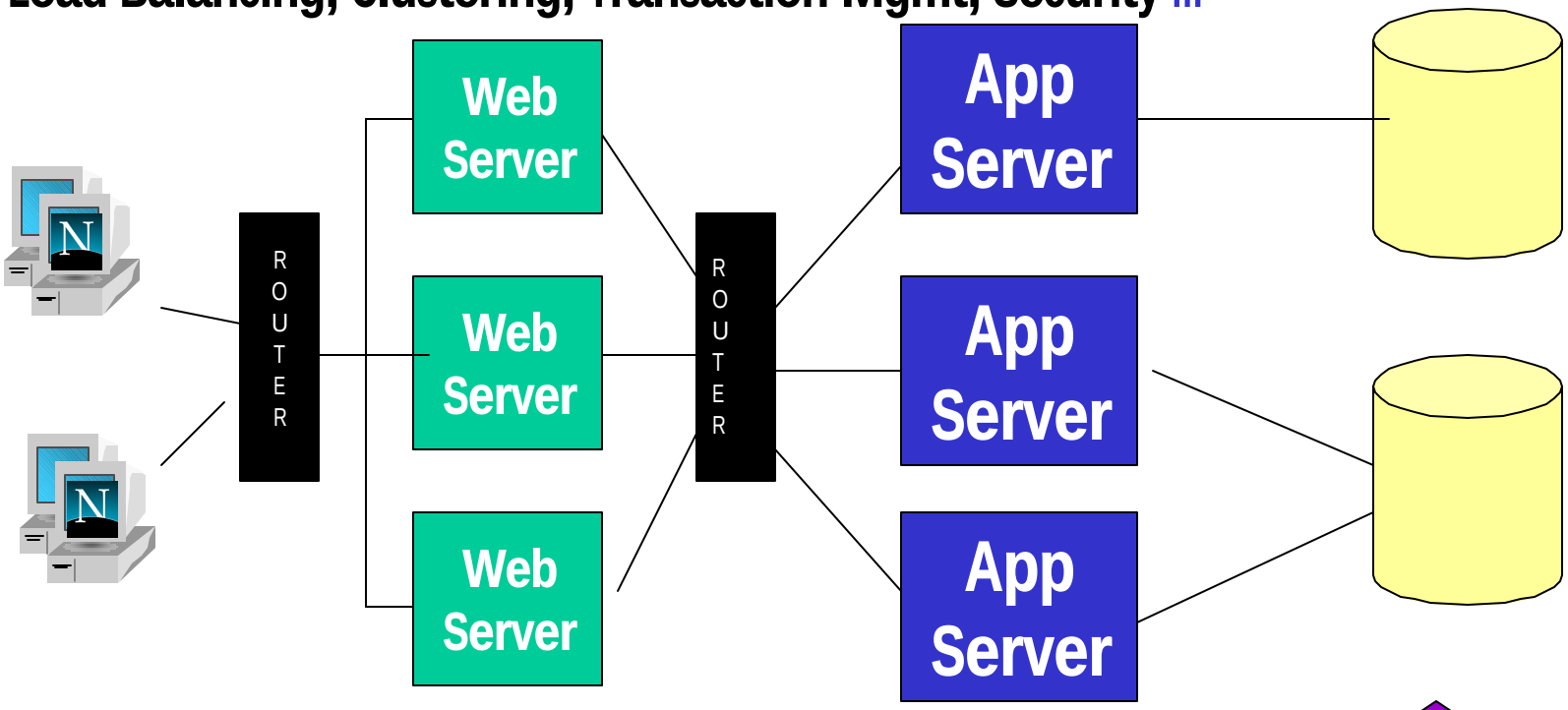
Tiers

- **Tiers separate functionality**
  - Presentation Logic, Business Logic, Data Schema
- **Advantages:**
  - One tier can be changed **without changing the rest= easier updates**
  - **Lower deployment** and **maintenance** costs.
  - Capabilities go up: Flexibility , Extensibility



Capabilities
Non-Functional Requirements

With Good Architecture, Layering, Tiers,
Load Balancing, Clustering, Transaction Mgmt, Security ...



Reliability, Availability, Extensibility,
Flexibility, Scalability, Goes up

Why J2EE™?

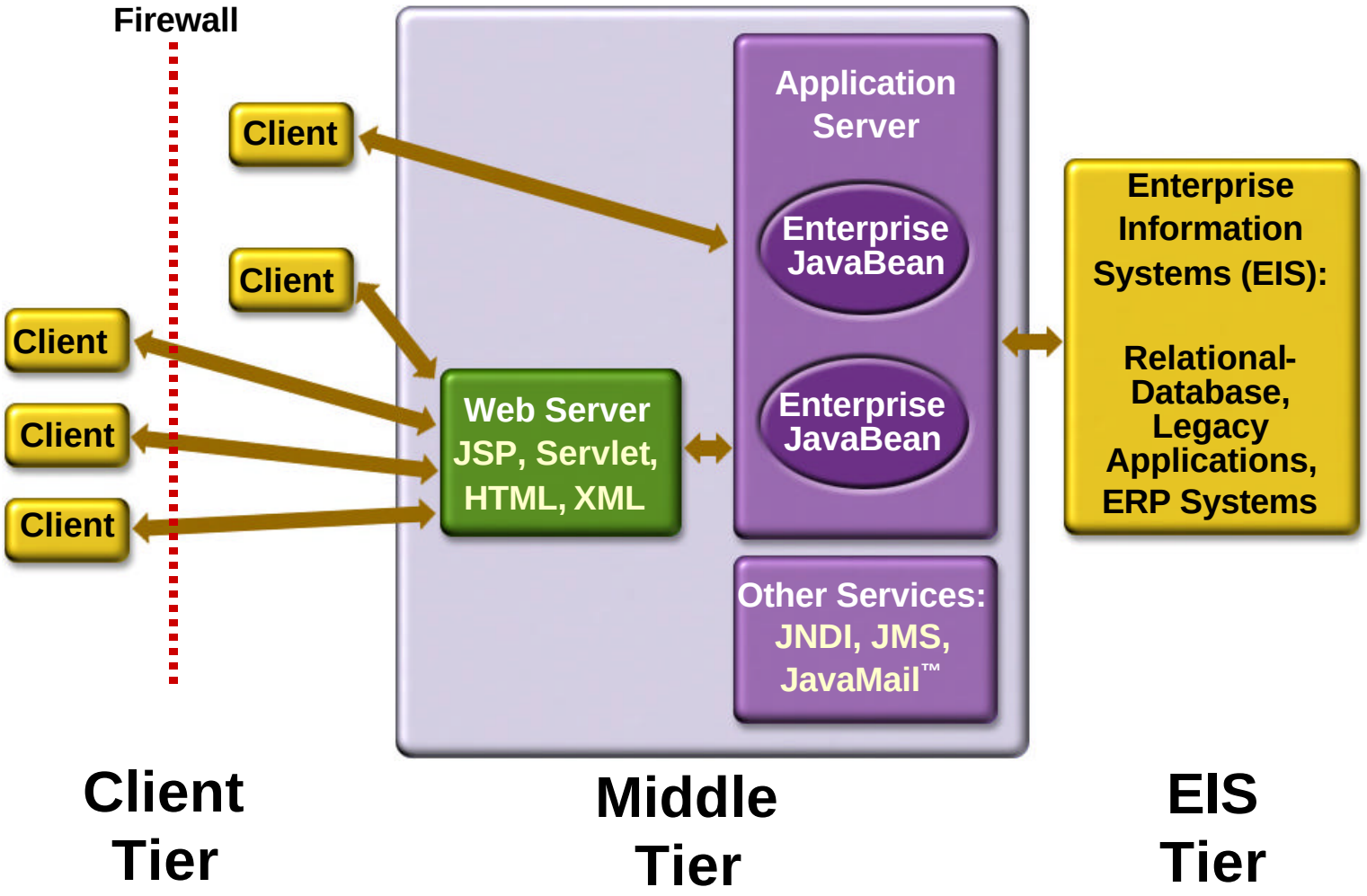
- Gives you the **Advantages** of a **Standardized Layered, Tier** Architecture:
- **Simplifies** the complexity of a **building n-tier applications**.
- **Standardizes** an **API** between **components** and application server **container**.
- **J2EE™ Application Server and Containers provide the framework Services**

Agenda J2EE™ & Blueprints

- J2EE Architecture
- *Sun BluePrints ™ Design Guidelines*
- *Architecture of the sample application*

EMPOWERING
THE DEVELOPER

The J2EE™ Platform Has a **Rich Set of APIs**



Why J2EE™ Blueprints

Rich APIs → **More than one way** of doing things

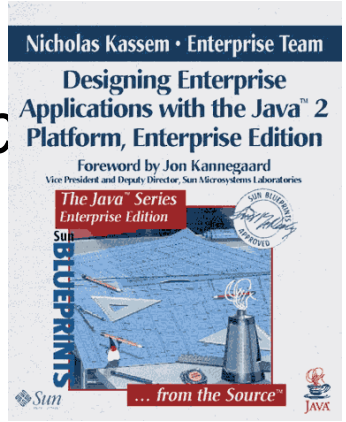
Similar Problems → Need to **leverage time-tested solutions**

Don't reinvent the wheel

Blue Prints → Provides **Answers**, Examples of **Best Practices**

J2EE™ Blueprints

- **Guidelines** and **Best practices** **for** designing *n-tier* enterprise applications with **J2EE™**
- **J2EE™ blueprints include:**
  - Book, *Designing Enterprise Applic*



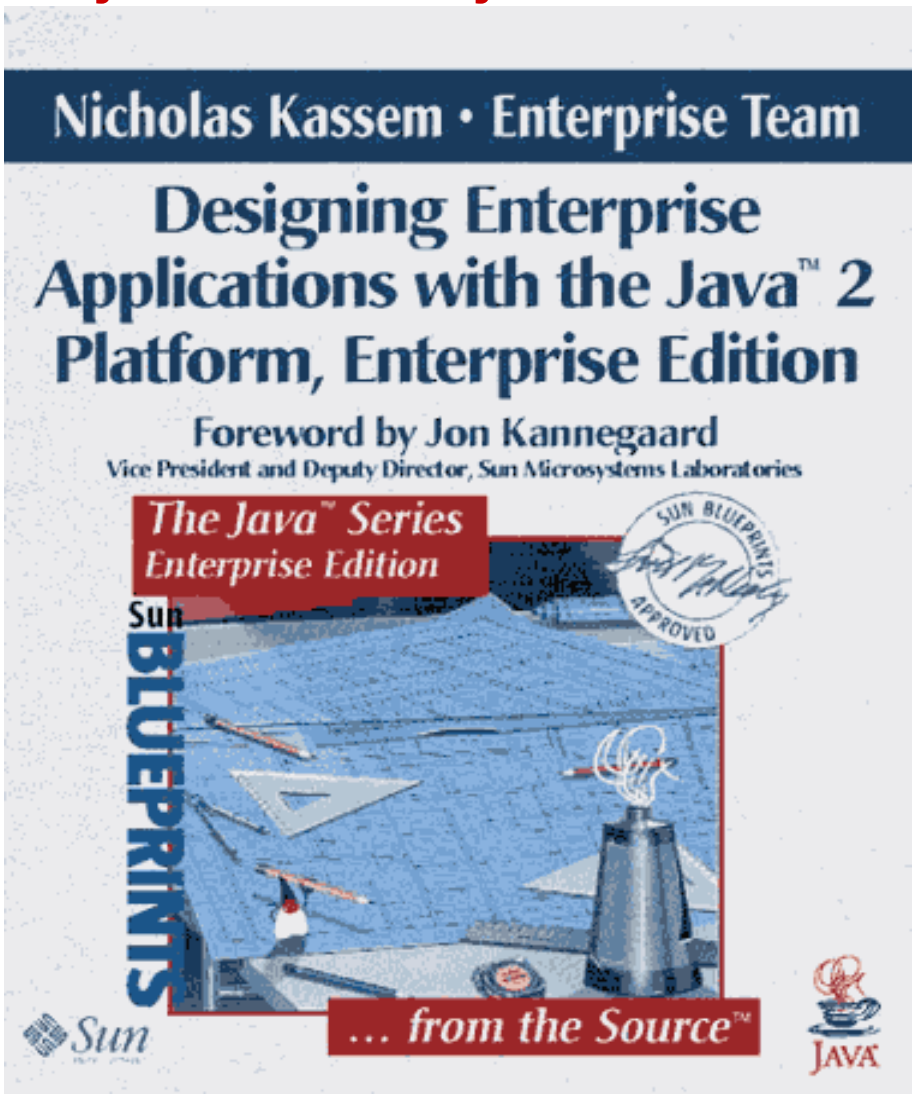
  - Application (**source code**)–
    - "Java Pet Store"

    FREE:
    - **http://java.sun.com/j2ee**

EMPOWERING THE DEVELOPER

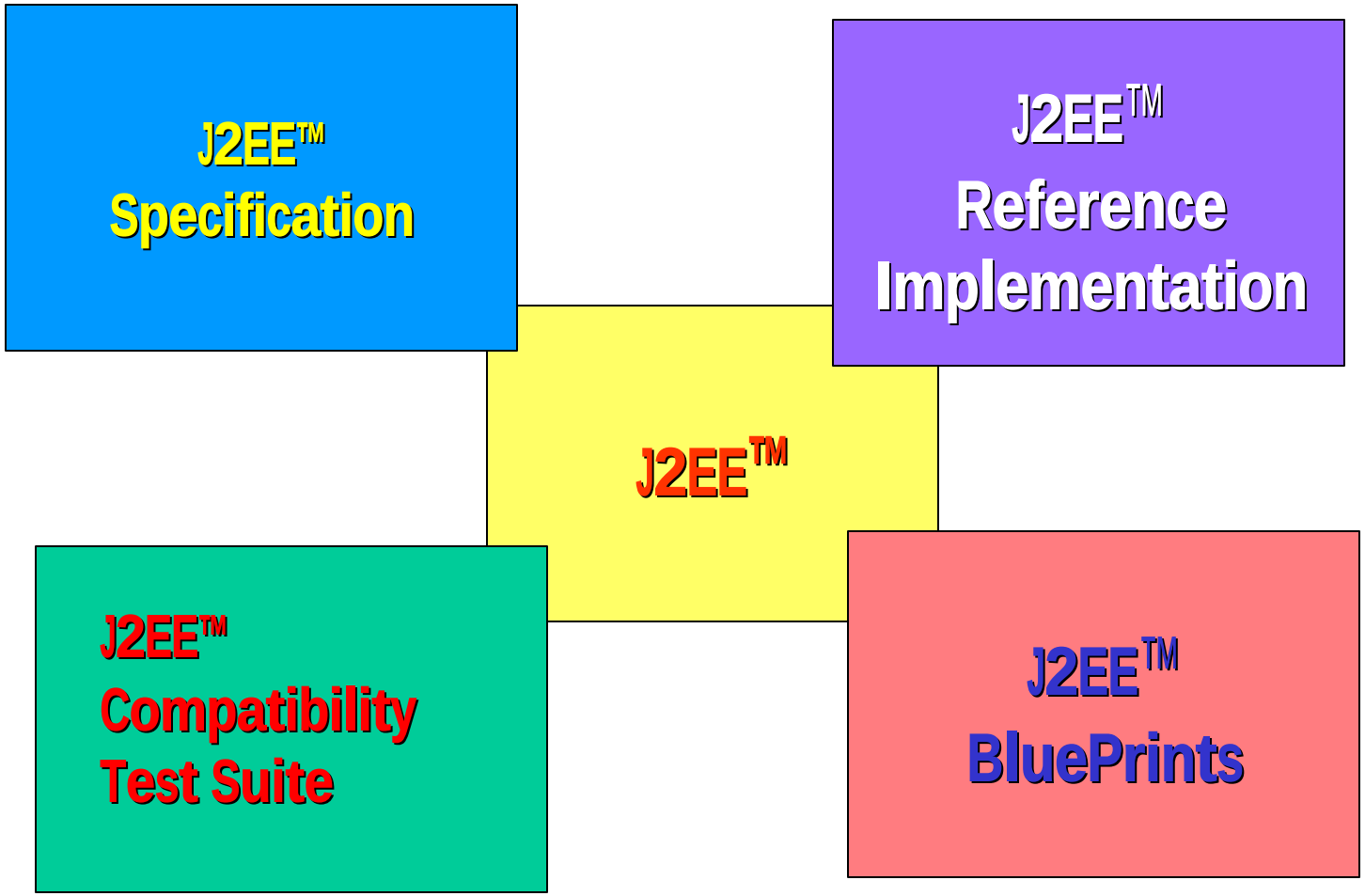
Blueprints

FREE: http://java.sun.com/j2ee



EMPOWERING THE DEVELOPER

J2EE™ Deliverables



“Java™ Pet Store”Demo

Some Key BluePrints™ Questions -Issues Faced by Developers

- **When to use Servlets** verses **JavaServer Pages™**?
- **Where to store client session state**: client-tier, web-tier or EJB™ tier?
- **Access Data Base Resources** via **EJB™** or directly from JSP™ JavaBean ?
- **When to use EJB™** components?
- **When to use Session Beans** and when to use **Entity Beans**?

Design of Sample Application
Blueprints Choices:

Blueprints Choices:

Client tier

- **HTML Clients** for End-User Web Access
- **XML** for **Data externalization (Order procurement)**

Web Server tier

- JavaServer Pages™ (**JSP**)
  for **dynamic content generation**

Application Server tier

- **EJB**™ for **business logic**

J2EE™ Blueprints Guidelines

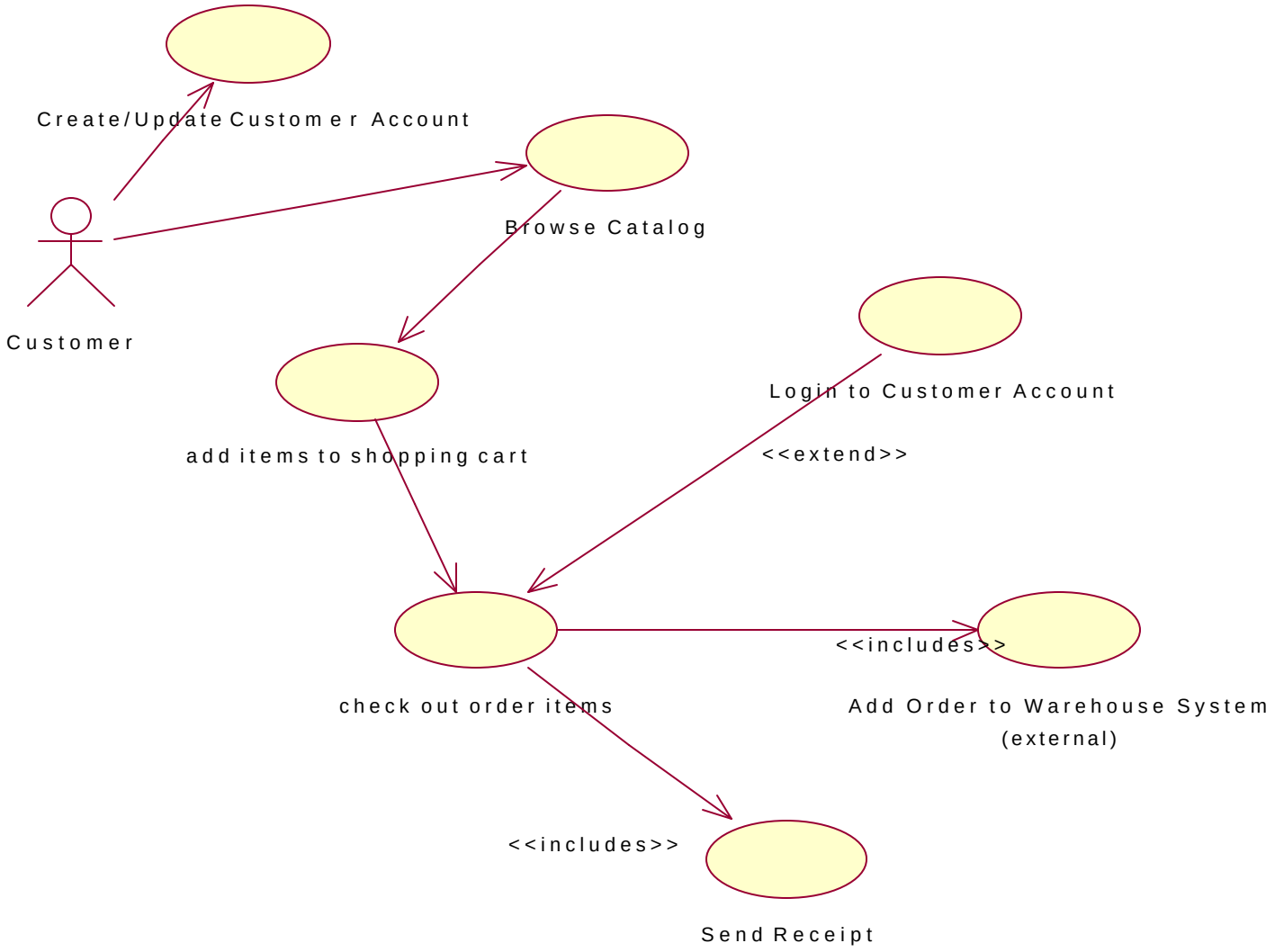
Process for developing an application from **requirements, to design, to implementation**

1. **Scenarios or Use Cases**
2. partitioning of functionality into modules
3. Application architecture using Model-View-Controller:
    1. assignment of functionality to tiers:
        1. Design of the Model
        2. Design of the View
        3. Design of the Controller
    2. object decomposition within tiers

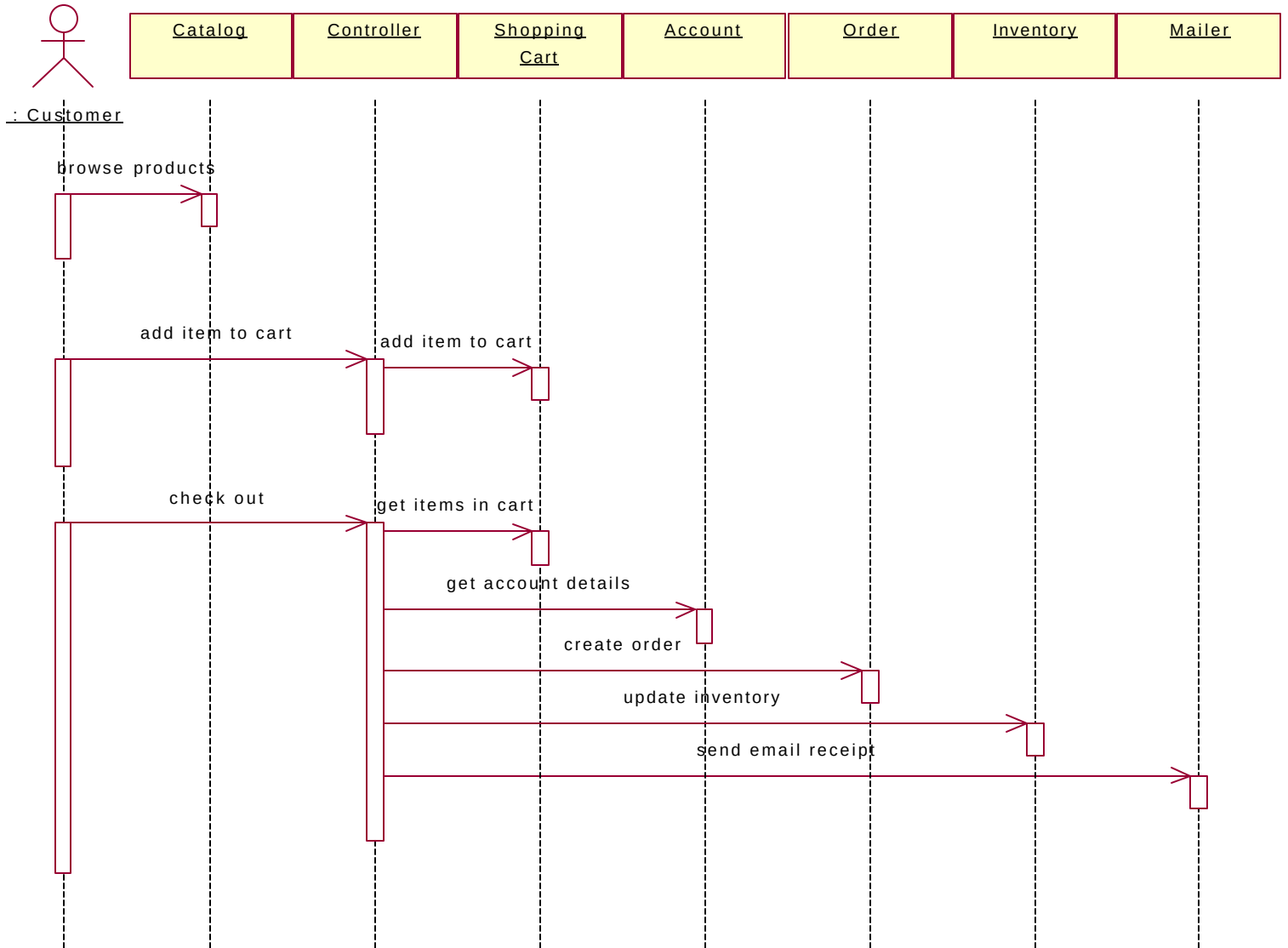
1) Sample Application – Shopping Scenario

1. Web-based **browsing** of a product **catalog**
2. Creation and maintenance of a **shopping cart**
3. User **account** creation/ Login
4. **Placing orders**
5. Secure order processing
    1. B2B transactions
    2. **Externalisation of order data** (expressed in XML)
    3. **Order confirmation** using e-mail

1) Use Case Shopping Scenario



Shopping Scenario Sequence Diagram

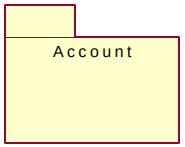


J2EE™ Blueprints

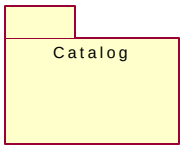
Process for developing an application from **requirements, to design, to implementation**

1. Scenarios or Use Cases
2. **partitioning of functionality into modules**
3. Application architecture using Model-View-Controller:
    1. assignment of functionality to tiers:
        1. Design of the Model
        2. Design of the View
        3. Design of the Controller
    2. object decomposition within tiers

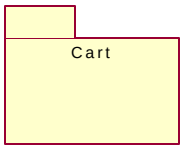
2) Divide the Application Into Modules Based on Services (Functionality)



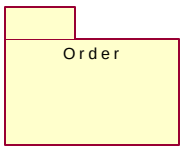
User Account Module: tracks user account information



Product Catalog Module: provides search for products, and product details

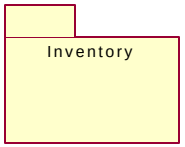


Shopping Cart Module: allows user to save selected items for the session.

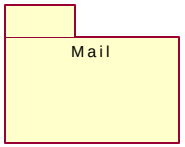


Order Processing Module: performs order Processing when user buys the items in the cart.

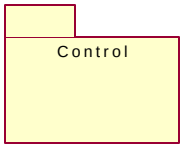
2) Divide the Application Into Modules Based on Services



Inventory Module: maintains information on the number of each type of product in stock.

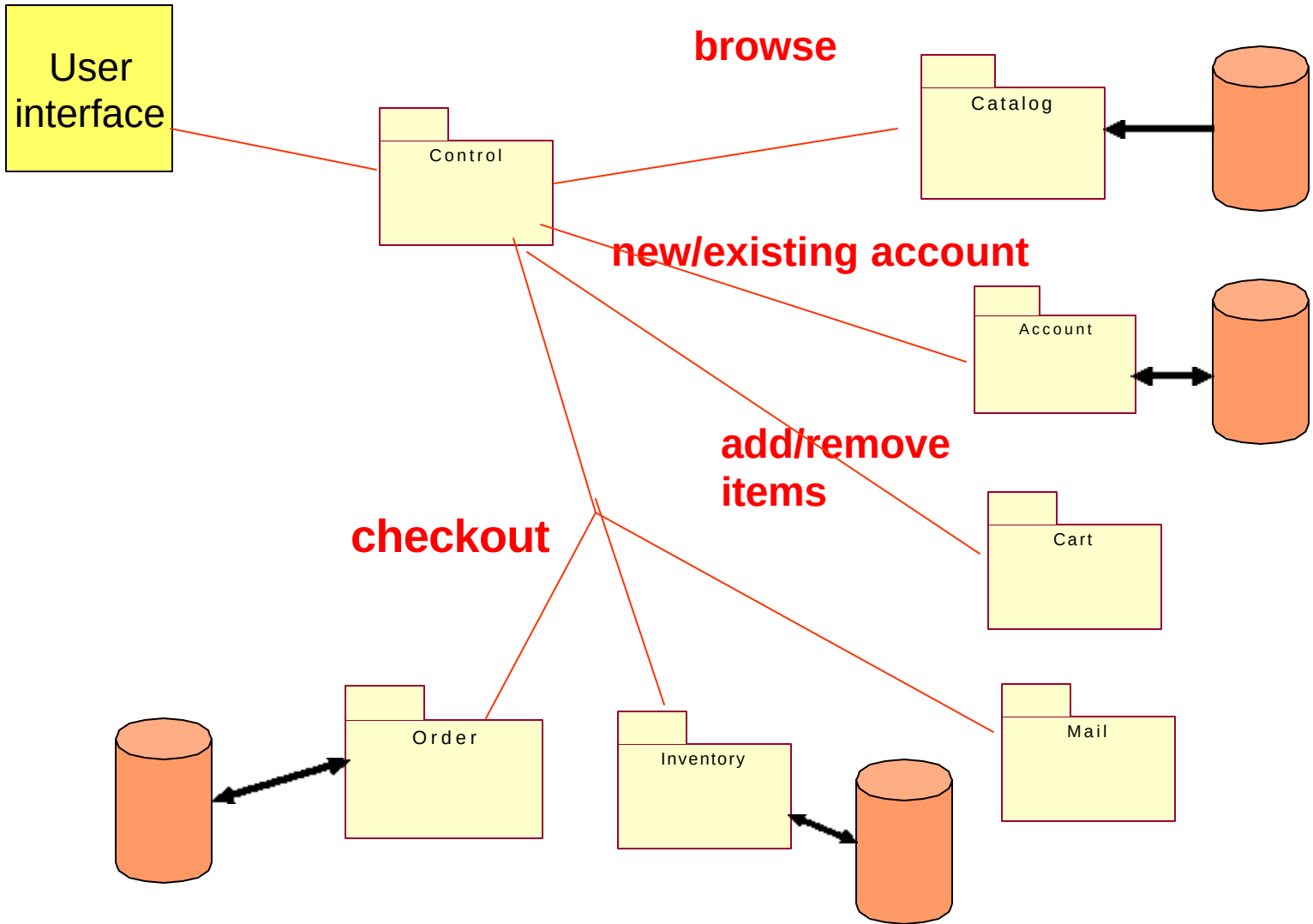


Messaging Module: sends confirmation receipt messages.



Control Module: control interactions between user (browse, add items, check out) and business objects.

Interrelationship of the modules



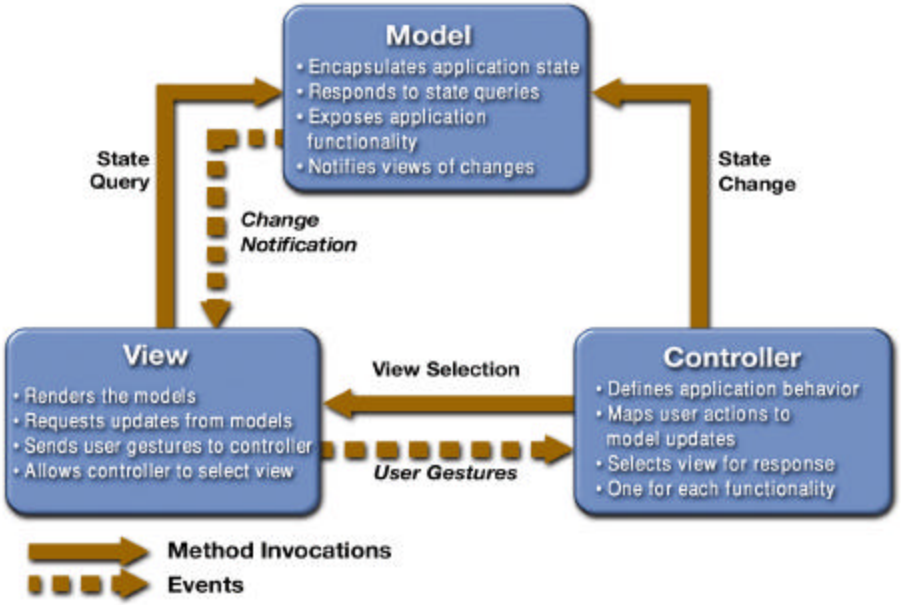
J2EE™ Blueprints

Process for developing an application from **requirements, to design, to implementation**

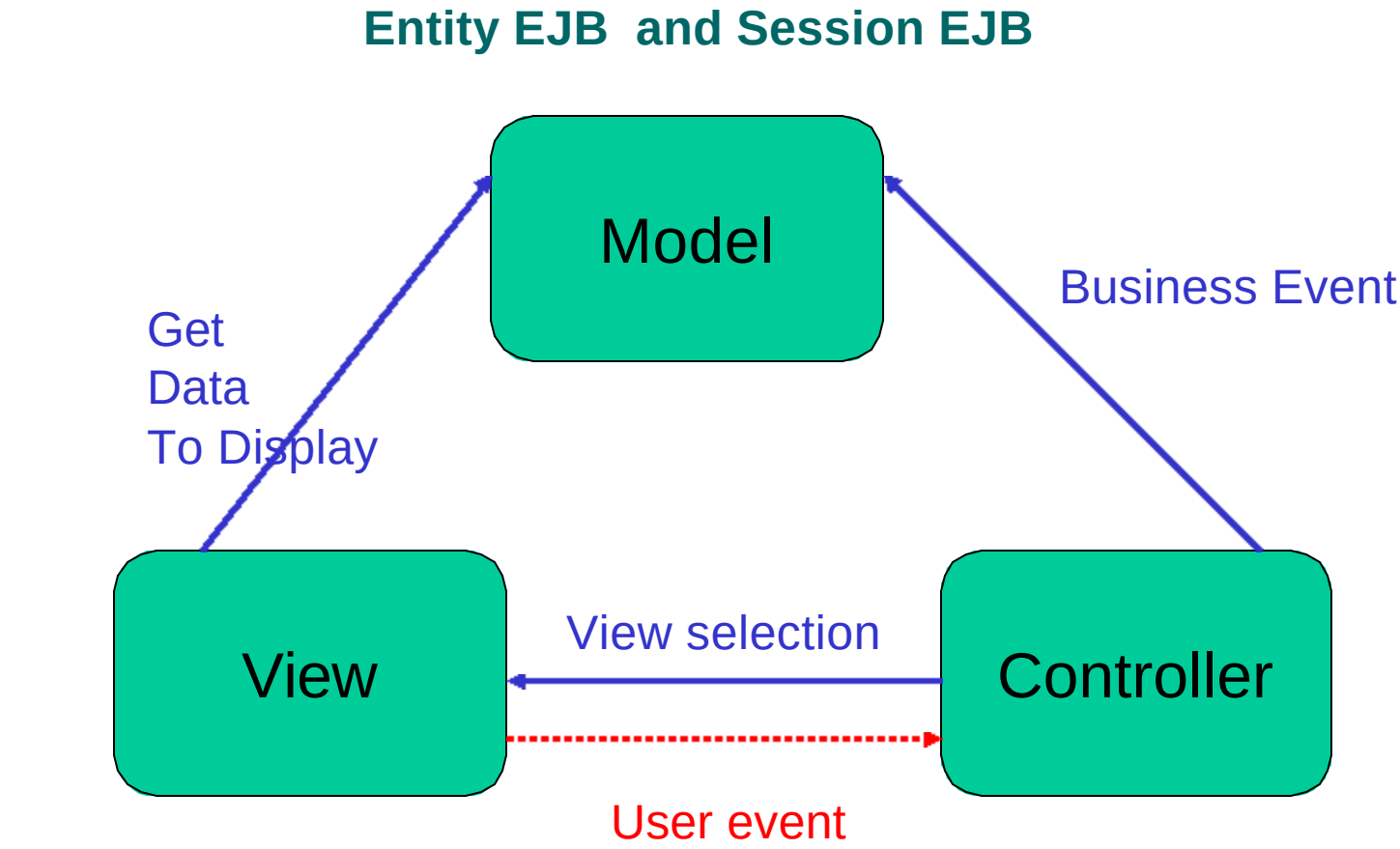
1. Scenarios or Use Cases
2. partitioning of functionality into modules
3. **Application architecture using Model-View-Controller**:
    1. **assignment of functionality to tiers**:
        1. Design of the **Model**
        2. Design of the **View**
        3. Design of the **Controller**
    2. object decomposition within tiers

Divide Application Objects into 3 categories:

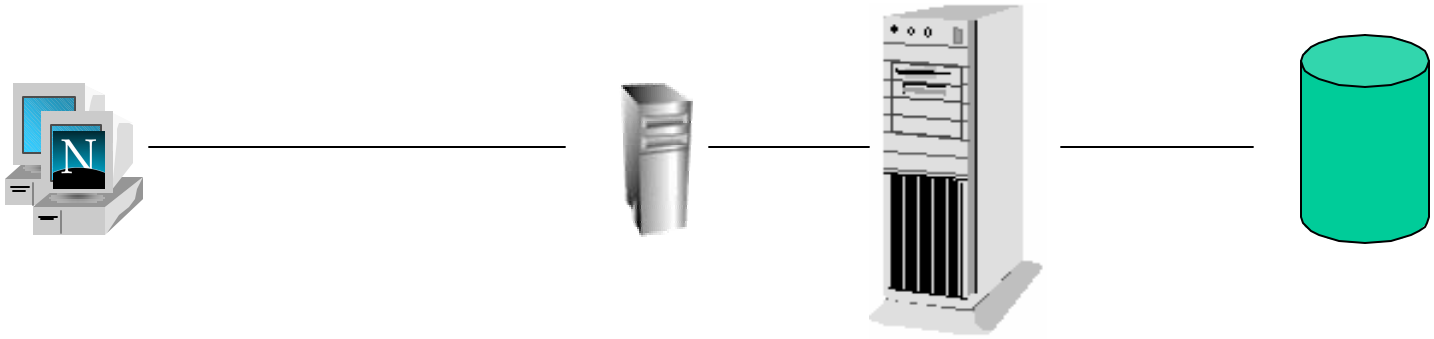
- Model View Controller (MVC) UI Pattern
  - **Model** represents the application **business data and rules**
  - **View** is a **screen representation of model**, **presents the user interface**
  - **Controller mediates** their **interactions**



Model View Controller Design Pattern



Assignment of Functionality to Tiers

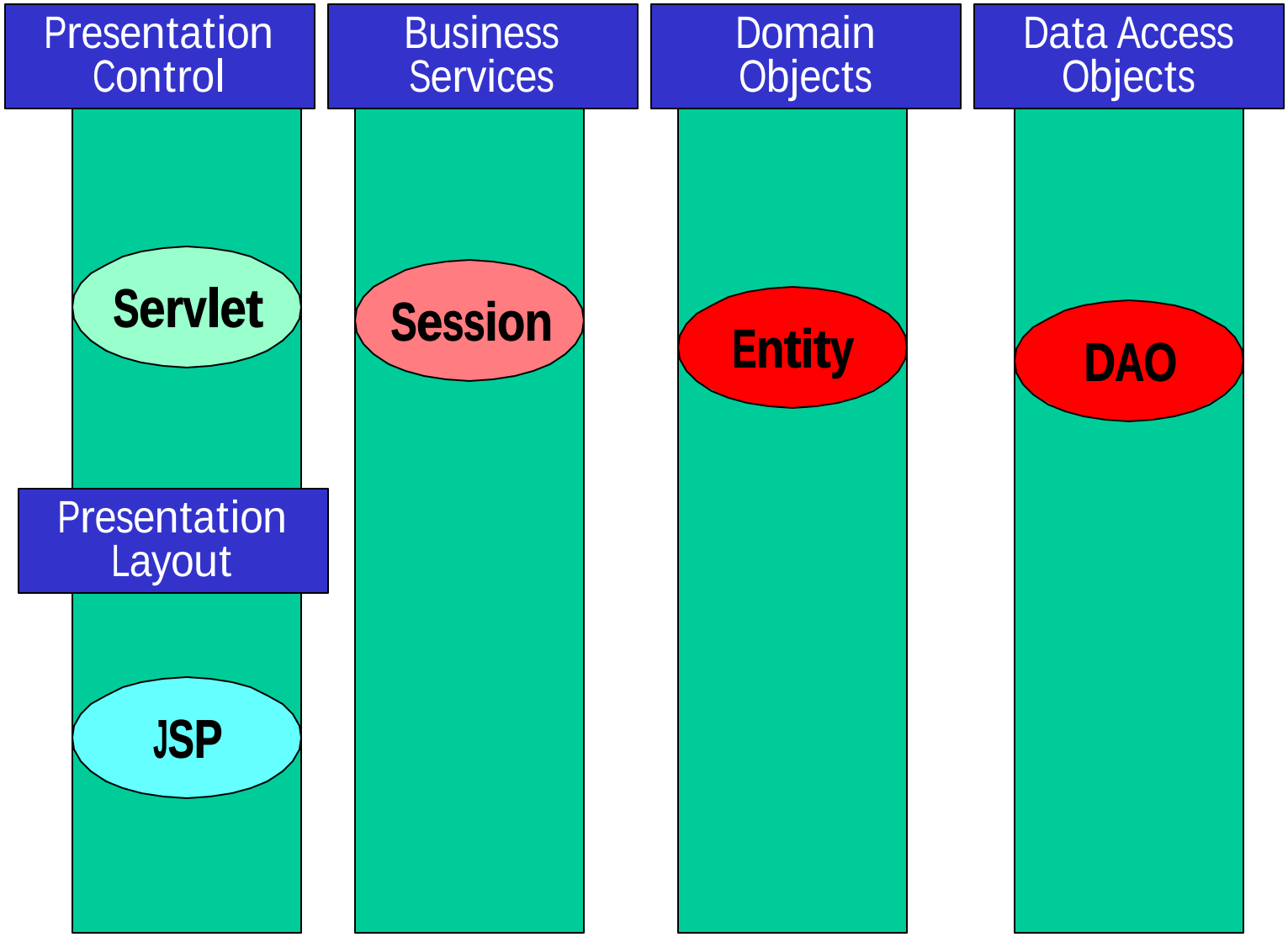


Presentation	Business Logic	Data
Presentation Logic	Application Requests	Business Services	Domain Objects	Persistence
View	Controller	Model

MVC With J2EE™ Technology

- Build **View** in the **Web**-tier
  - Use **JSP**™ components for **presentation layout**
  - Custom tags and **JavaBeans**™ components for **presentation logic**
- Build **Controller** in the **Web / EJB**™ tier
  - **Servlet** for **dispatching**
  - **Session** bean for **control within business objects**
- Build **Model** in the EJB™ tier
  - **Session** beans for **Business Services**: Process and Rules
  - **Entity** beans to maintain **model data**

Assignment of Functionality to Tiers or Vertical Layering



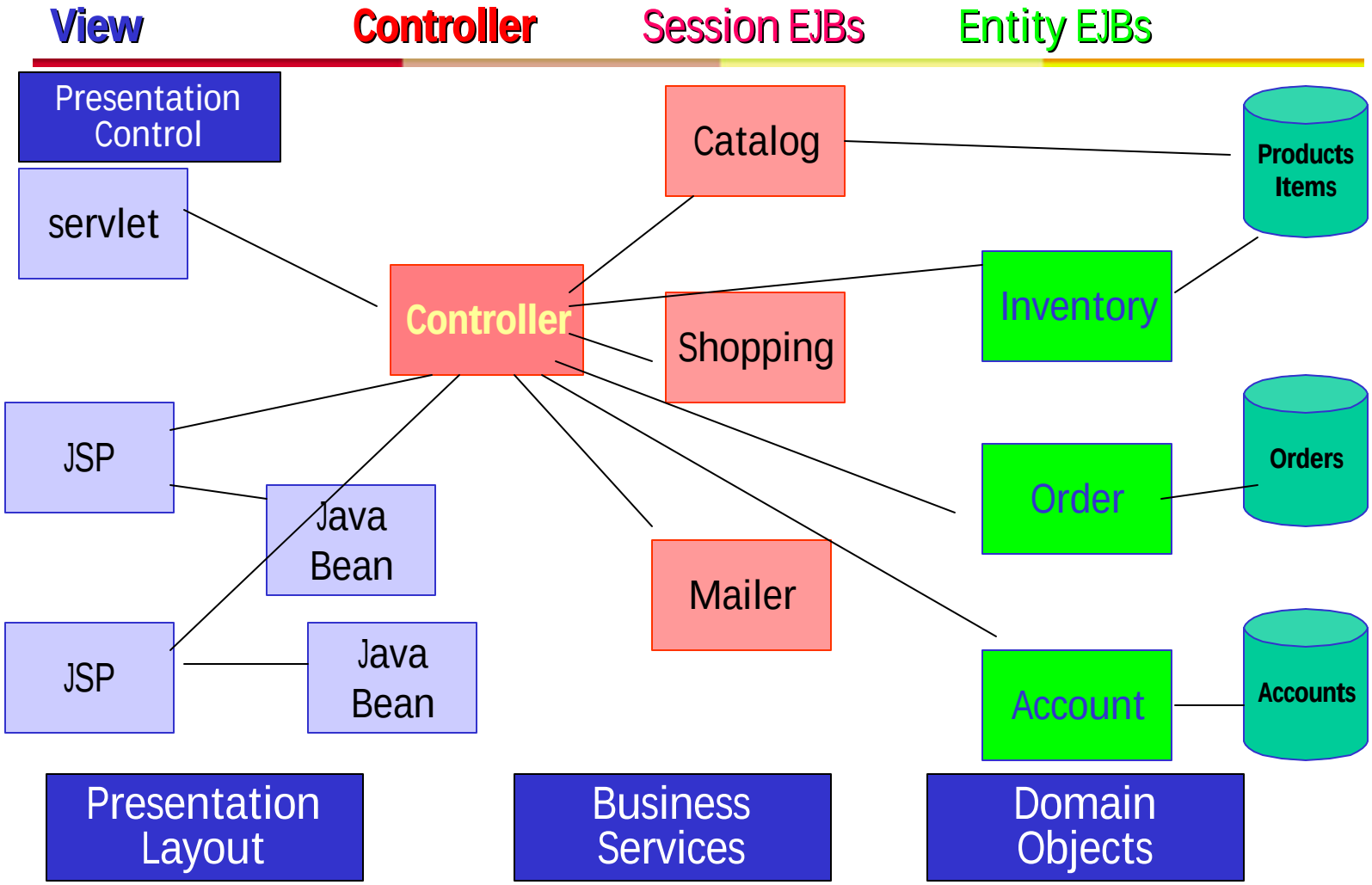
What are Domain Objects?

- Domain Objects **model the real world entities, nouns** in business problem
- Also known as **Data Model, business object model**
- **Example**
  - User Account,
  - Order,
  - Product,
  - Inventory

What are Business Services?

- **Business Services** **Model the** **verbs** in application
- **Business processes** are the **services in the use cases**
- **Example**
  - **Browse** Catalog
  - **Add** **item** to Cart
  - **Purchase** items

Sample Application Architecture **Vertical Layering**



Summary

- **We went over:**
  - Advantages of **horizontal layers**
  - Advantages of **Vertical Tiers**
  - **Affect of Layers and Tiers on Capabilities.**
  - the **process of architecting** the sample application
  - the **Model View Controller** Design Pattern