Web Application Architecture

N-Tier Systems

Enterprise Class Applications

- What are Enterprise Class Applications?
  - Business critical applications
  - Large numbers of users
  - Often large, complex and physically distributed.

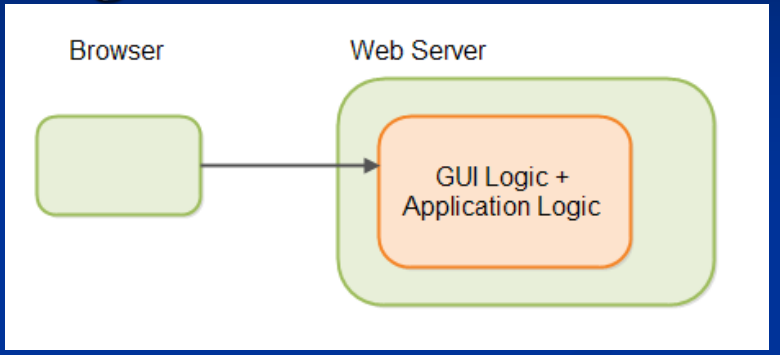
- Enterprise Class Attributes:
  - Availability
  - Reliability
  - Security
  - Scalability
  - Manageability
  - Interoperability
  - Maintainability

How do we achieve Enterprise Attributes?

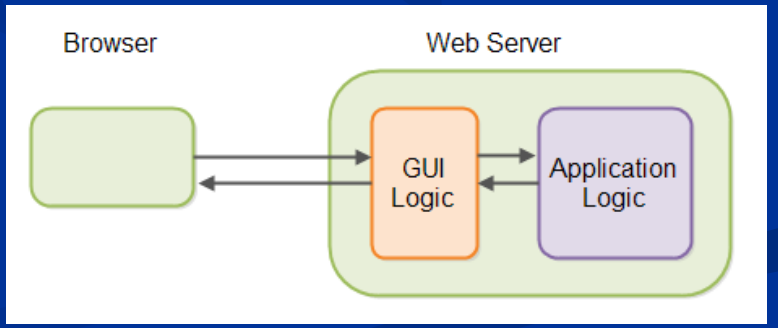
- Architecture (n-tier? SOA? RIA?)
- Object and component oriented design
- Leveraging enterprise frameworks
- Use of open standards for interoperability
- Robust runtime environment and management.

Evolution of Web App Dev Technologies

- First generation of web dev technologies
  - Page oriented
  - GUI logic and application logic inside the same web page
  - Code spread over multiple pages
  - CGI, Java Servlets, JSP (JavaServer Pages), ASP, PHP etc

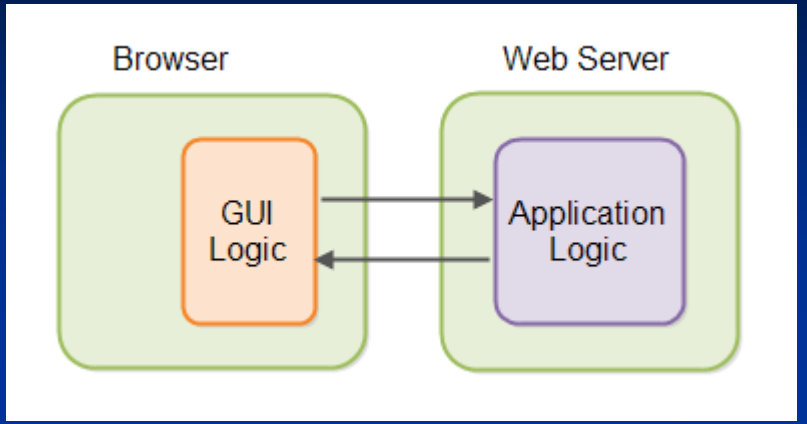


- Second generation of web dev technologies
  - GUI logic separated from the application logic
  - Both GUI and Application logic on the server
  - Frameworks were developed: ASP.NET, Java ServerFaces, Spring MVC (Java)



Evolution of Web App Dev Technologies (Cont.)

- RIA web dev technologies –(Third generation ?)
  - Rich GUI, good for Game, video
  - GUI logic separated from the application logic
  - GUI logic runs in the browser
  - Application logic on the server
  - GUI and back end services only exchange data (no exchange of HTML, CSS and Java Script anymore)
    - Except the firs time the page is loaded

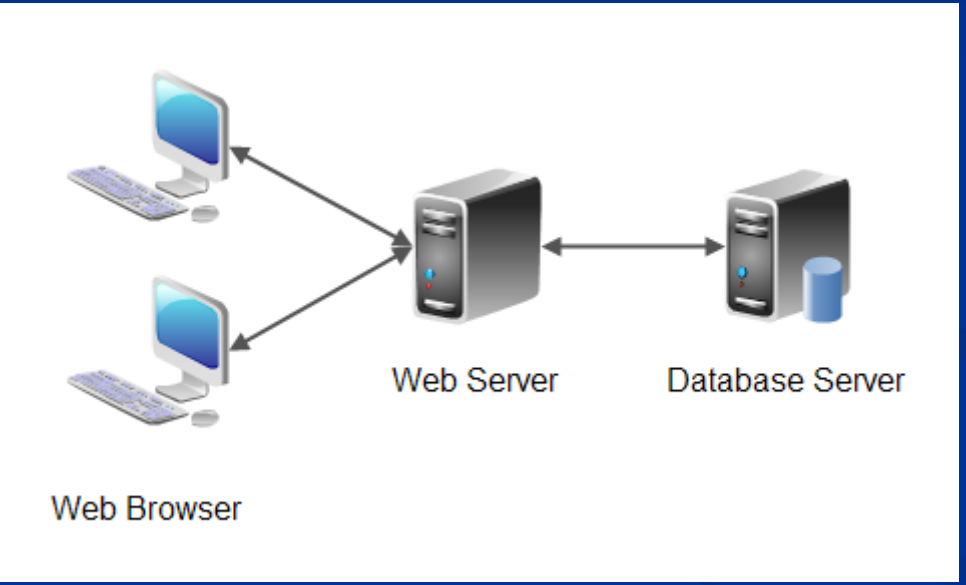


- Well-known RIA technologies
  - HTML5 + CSS3 + JavaScript + JavaScript frameworks
    - jQuery, jQuery Mobile, AngularJS, GWT (Google Web Toolkit), ….
  - JavaFX, Adobe Flex/Flash (fading out), Silverlight (dead)

N-Tier Architecture

- Logic is distributed among *logical* "tiers"
- Processing is distributed among *physical* "tiers"

- Typically 3-Tier
  - Presentation
  - Business Logic
  - Data Access



- N-Tier is usually an expanded model of 3-Tier

N-Tier Architecture

Web Browser

HTTP

Rich (Thick) Client

Web Pages and Presentation Logic

Business Components

Data Access

Relational Databases

XML

Flat Files

Messaging Services

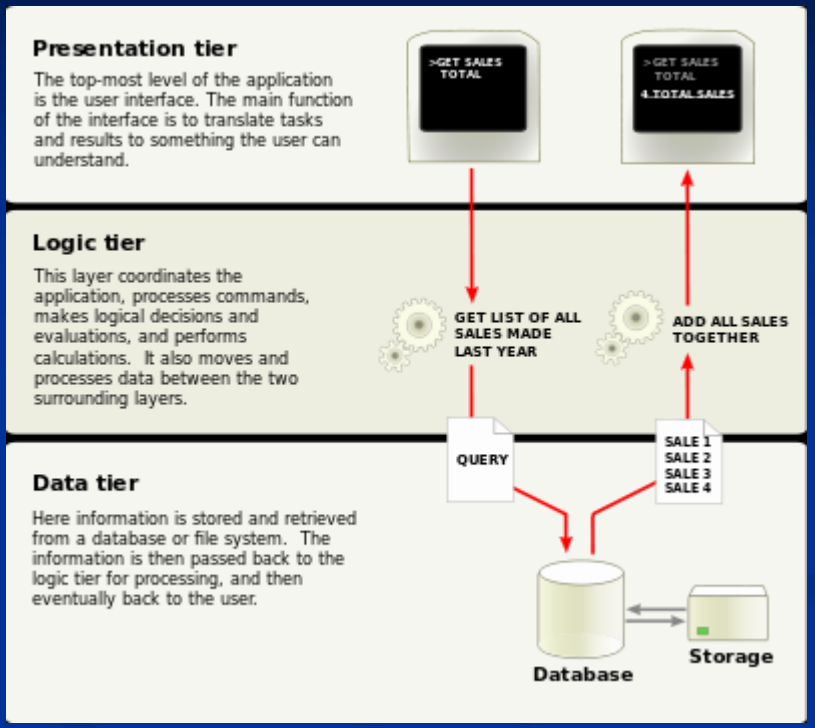
Presentation

Business logic

Data services

Typical Roles of Each Tier

- Presentation
  - Rendering, acquisition and validation.
  - Interpret gestures.
- Business
  - Stateless components (normally???)
  - Implement short-lived business activities
  - Enforce policy
  - Initiate atomic transactions
  - Orchestrate messages and activities
  - Aggregate data from data sources and services
  - Represent data in business-relevant schemas
- Data Access
  - Stateless encapsulation of data access and transformation logic
  - Participate in transactions
  - Provides wrappers for calling stored procedures.

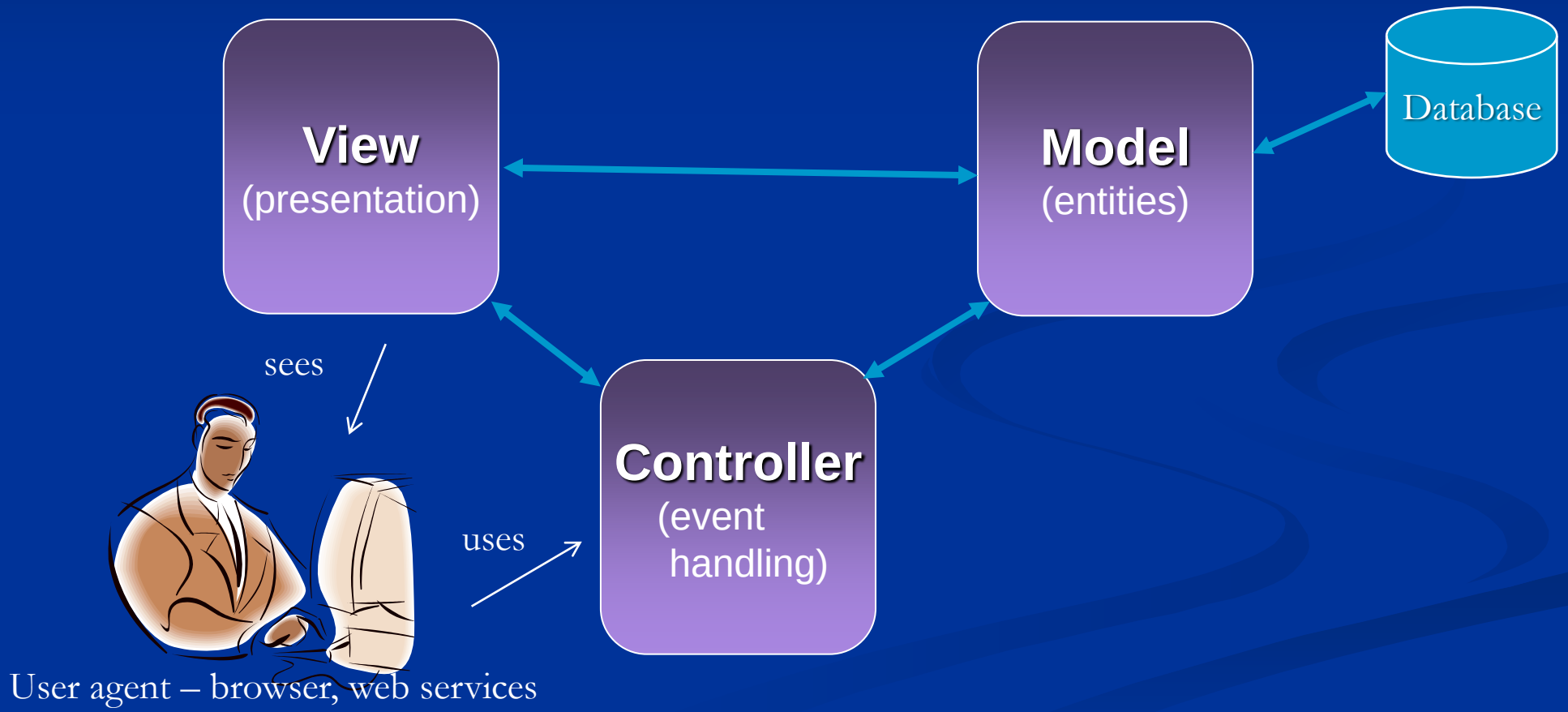


Benefits of N-Tier Architecture

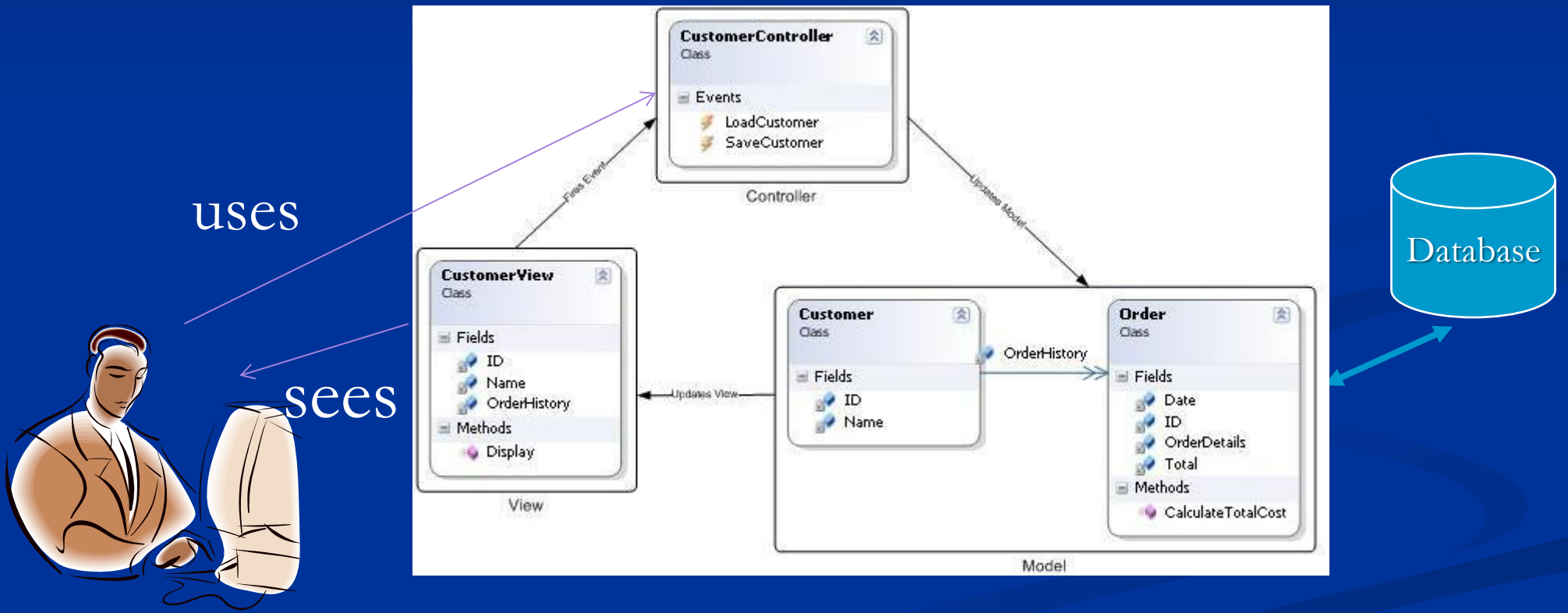
- Scalability and Performance
  - Can use different machines for each tier.
  - Can also use multiple machines per tier.
  - Concentrated resource use supports pooling, caching and recycling.
- Maintainability and Stability
  - Highly specialised components.
  - Separation of concerns.
  - Robustness through physical redundancy.
- Flexibility
  - Supports reuse.
    - Different types of clients can access the same business services.
  - Interchange components with same interface
  - Loosely coupled → flexible deployment

Alternatives to N-Tier

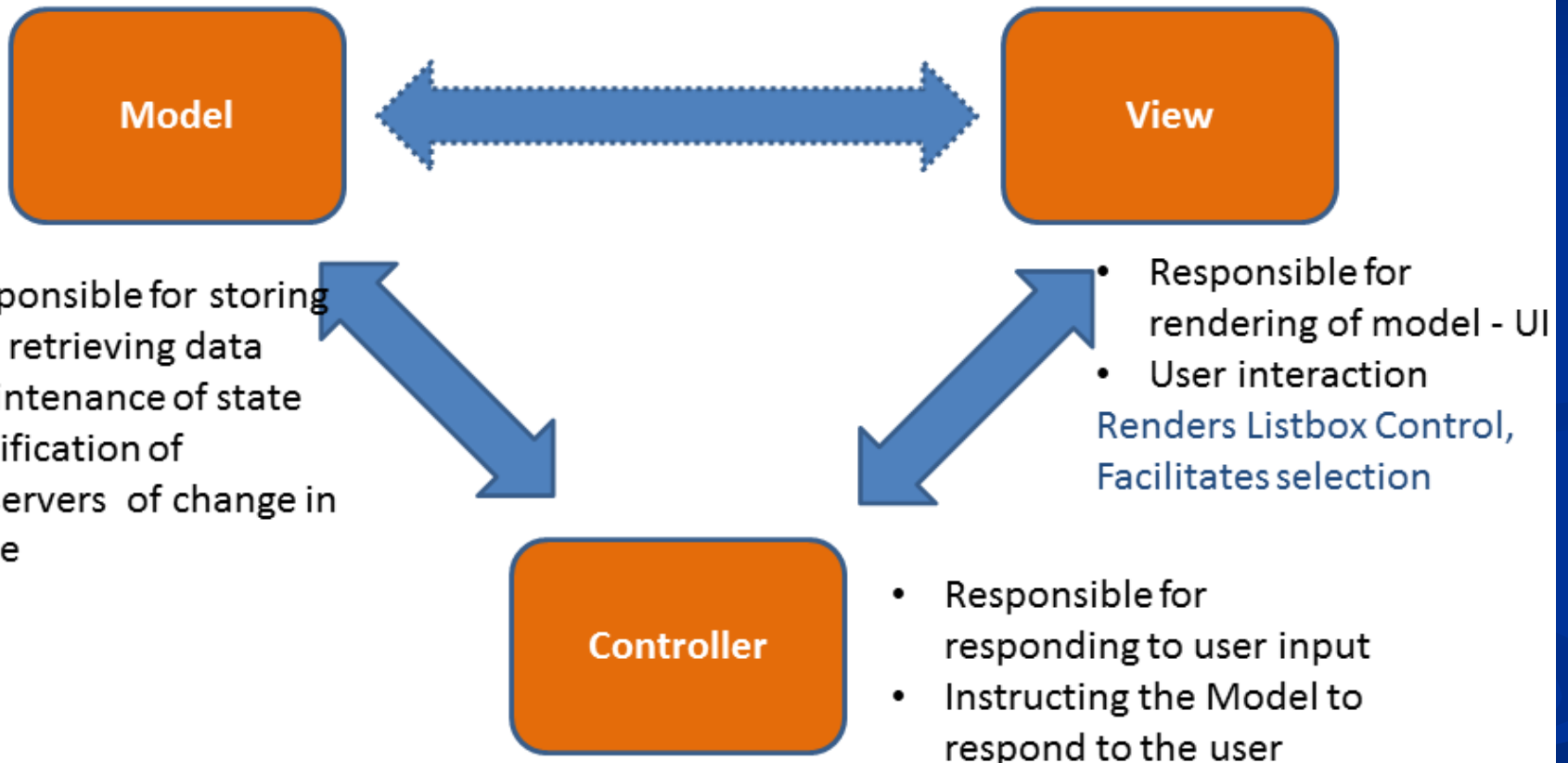
- Model View Controller (MVC)



Model View Controller example



Model View Controller (MVC) Arch Pattern



Popular MVC Web Frameworks

- Java
  - Spring (http://www.springsource.org/)
  - Apache Struts (http://struts.apache.org/)
- .NET
  - .NET MVC (http://www.asp.net/mvc/)
- Ruby
  - Ruby on Rails (http://rubyonrails.org/)
- …

Leveraging Enterprise Frameworks

Enterprise Application Frameworks

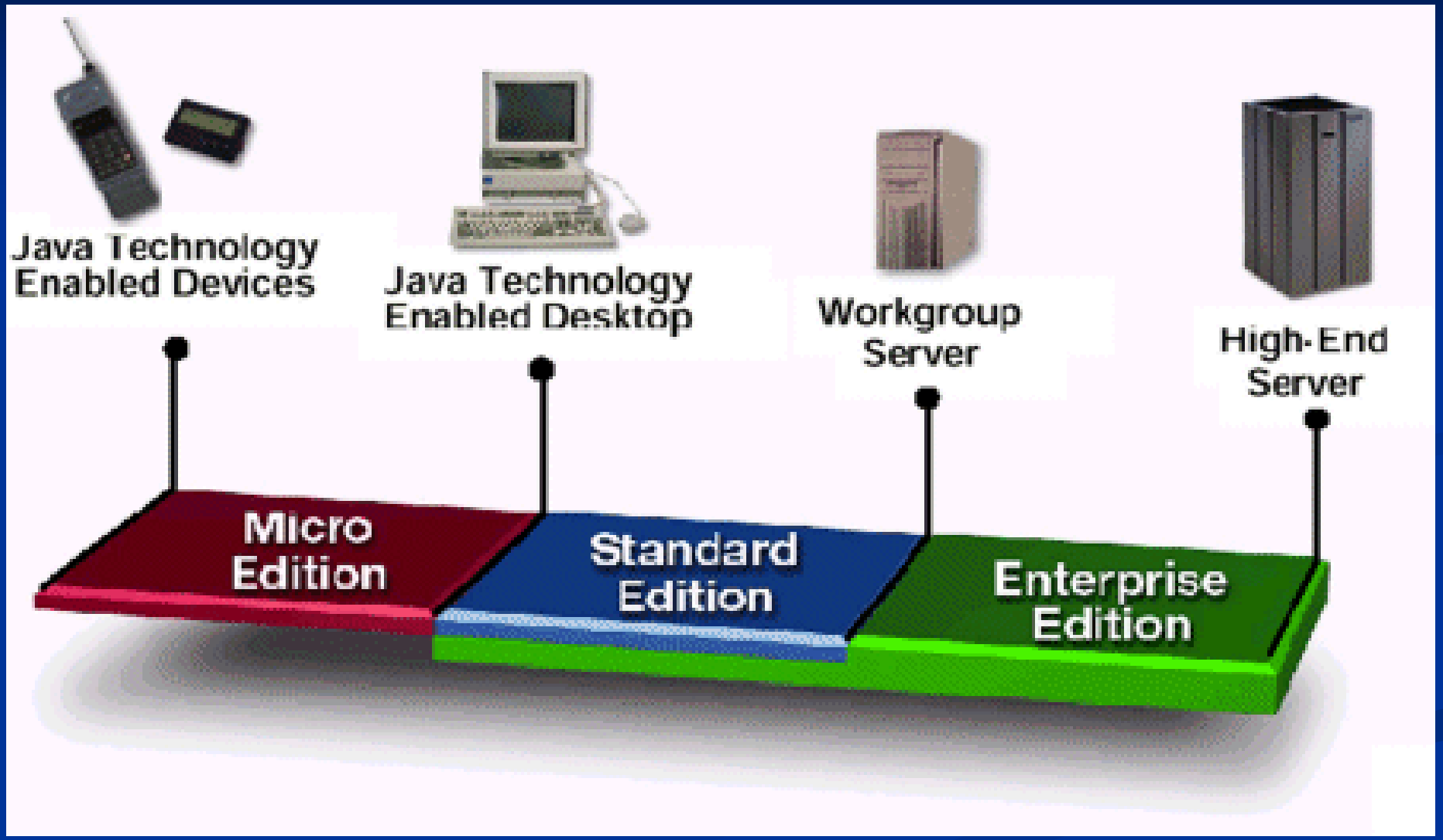
- End-to-End software frameworks for developing component-based distributed enterprise applications.
  - Windows Distributed interNet Applications (DNA)
  - Java Platform, Enterprise Edition (Java EE)
  - .NET (Enterprise Services)
  - Service Component Architecture (SCA)
  - Windows Communication Foundation (WCF)

Enterprise Application Frameworks

Platform	Presentation	Business Logic	Communication/ Integration	Data Access
Windows DNA	ASP	COM COM+ (MTS)	DCOM MSMQ	ADO
Java EE	JSP Servlets JSF	EJB	remote EJB Java RMI JAXP Web Services	JDBC Connectors
.NET	ASP.NET	.NET COM+	ASP.NET Web Services .NET remoting MSMQ	ADO.NET

Java Platform, Enterprise Edition (Java EE)

Java Platform, Enterprise Edition (Java EE)



What is Java EE?

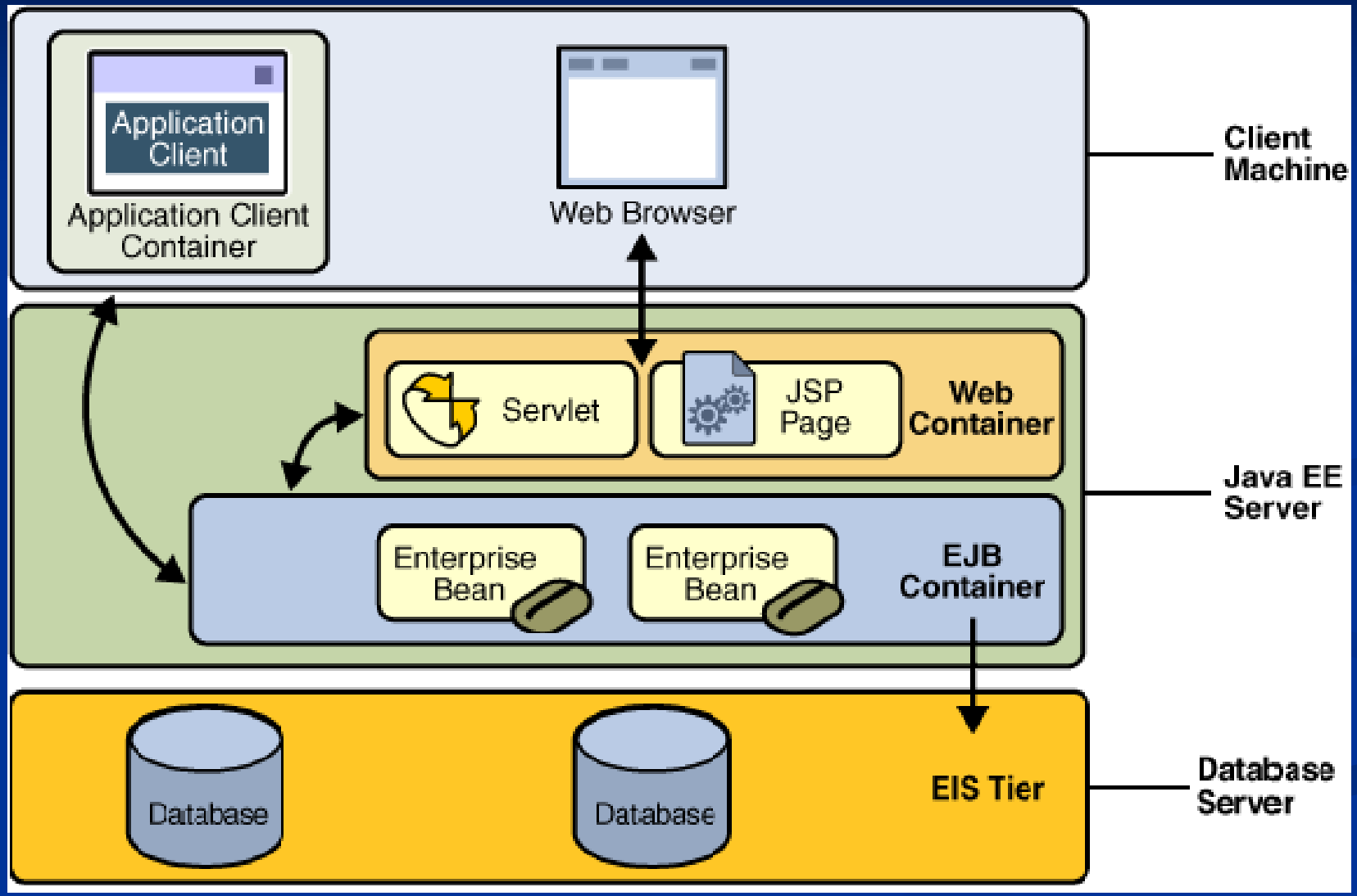
- The Java™ Platform, Enterprise Edition (Java™ EE) defines a standard architecture for developing multitier, enterprise services.

It consists of:

- **Java EE Platform**
  - A standard platform for hosting Java EE applications.
- **Java EE Compatibility Test Suite**
  - A suite of compatibility tests for verifying that a Java EE platform product complies with the Java EE platform standard.
- **Java EE Reference Implementation**
  - A reference implementation for prototyping Java EE applications and for providing an operational definition of the Java EE platform.
- **Java EE BluePrints**
  - A set of best practices for developing multitier, thin-client services.

Based around *containers* …

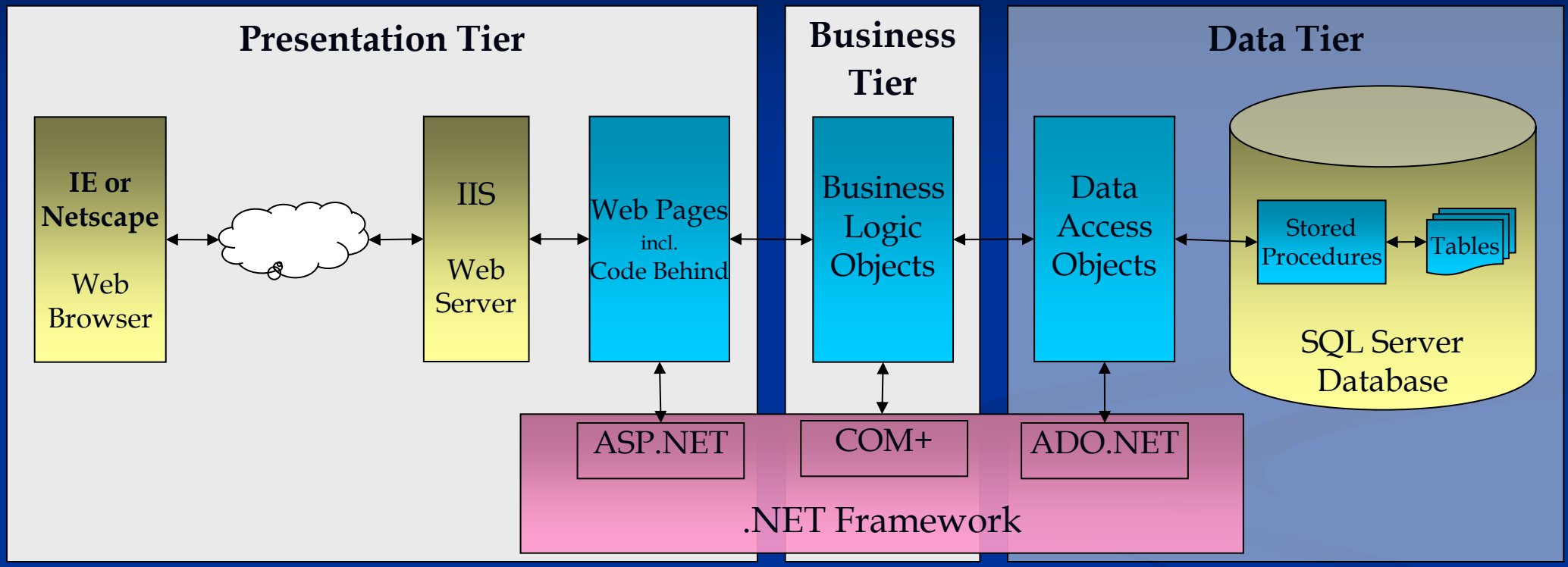
Java EE Architecture



Java EE References

- Java EE 6 Specification:
  http://jcp.org/en/jsr/detail?id=316
- Introduction:
  http://www.oracle.com/technetwork/java/javaee/overview/
- Tutorials:
  http://download.oracle.com/javaee/6/tutorial/doc/

.NET N-tier Architecture



N-Tier Architecture Reference

http://msdn.microsoft.com/architecture

- http://msdn.microsoft.com/library/bb384587.aspx
- http://msdn.microsoft.com/en-au/library/bb384398.aspx
- http://msdn.microsoft.com/en-au/library/bb882661.aspx

http://java.sun.com/blueprints/guidelines/designing_enterprise_applications

How do we Implement the Middle Tier?

How do we achieve enterprise class attributes?

Object-Oriented Development

- Best practice for designing and implementing software
- Base software architecture around "real world" objects which will "stand the test of time".
  - Minimizes architectural change in the face of evolving needs.
- Supports reuse at the source code level.
  - supports reuse over the lifetime of an application but not so well between applications.
  - often implicit dependences on other code limits reuse.
- Also need more flexible deployment options
  - upgrade parts of a deployed system ...

Component-Oriented Development

- A *software component* is:
  - a *binary unit of composition* with
  - *contractually specified interfaces* and
  - *explicit context dependences* only.
- A software component can be:
  - *deployed independently* and is
  - subject to *composition by third parties*.
- Component Frameworks:
  - COM
  - CORBA
  - Java Beans
  - .NET (Assemblies)

Enterprise Services

- Extended component models that provide additional services which contribute to enterprise class attributes.
  - Mostly used in the business tier.
  - Typically provided by a container or wrapper mechanism.
  - Allow the programmer to concentrate on the business problem rather than on complex low level details (plumbing).
    - decouples business logic from system and resource logic
  - Provide mechanisms to configure components at deployment time.
    - can differ between applications and hosts.

- Examples:
  - COM+
  - Enterprise Java Beans
  - .NET Enterprise Services

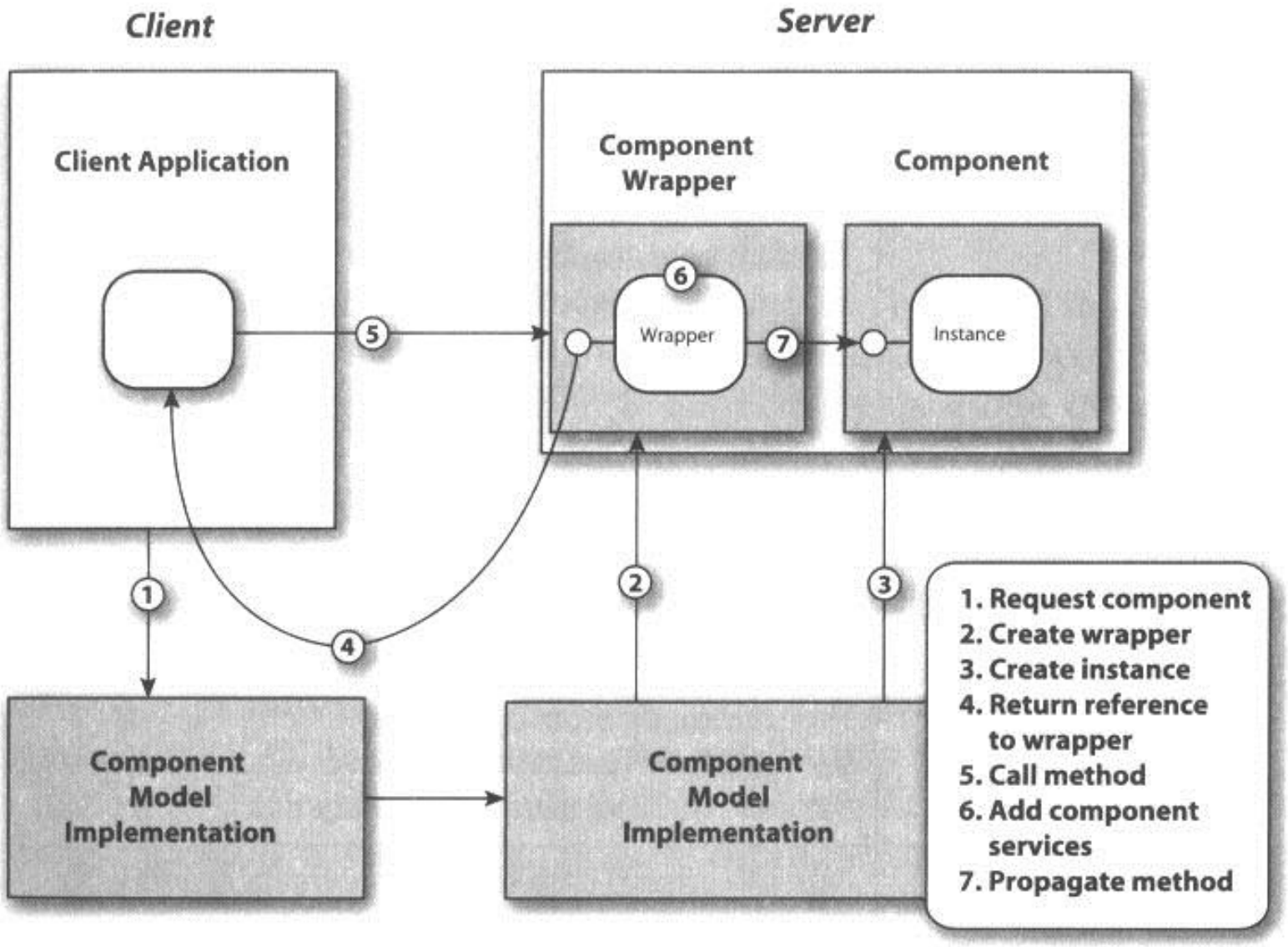


Figure 35-2: *Generic middle-tier component platform.*

Example of Enterprise Services

- .Net Enterprise Services:
  - Transactions
  - Role-base security
  - Queued components
  - Synchronization
  - Application and object pooling
  - Load balancing
  - Resource management
  - Dynamic parameterized construction
  - Loosely-coupled events
  - Compensating resource management
- Enterprise Java Bean Services:
  - Transactions
  - Security
  - Messaging (reliable, asynchronous)
  - Concurrency control
  - Component pooling
  - Load balancing
  - Resource management
  - State management
  - Persistence

Enterprise JavaBeans (EJB)

- A collection of Java classes that implement certain standard EJB interfaces, plus a deployment descriptor file that is used for deployment time configuration.
- Must be hosted within an EJB container which conforms to the Java EE specification.
- Types of EJBs:
  - Session Bean - used for dealing with a single client
  - Message-Driven Bean – asynchronous querying based
  - Entity Bean – replaced by the Java Persistence Framework

EJB References

- Specification:

  http://www.oracle.com/technetwork/java/javaee/ejb

- Tutorial:

  http://download.oracle.com/javaee/6/tutorial/doc/

.NET Enterprise Services

- Provides .NET components access to COM+ services.
- Inherit from System.EnterpriseServices.ServicedComponent
- Use attributes to declaratively specify use and configuration of enterprise services.
- "Serviced" components must be registered in the COM+ catalog.
  - regsvcs.exe Account.dll

.NET Enterprise Services References

http://support.microsoft.com/kb/308672

http://msdn2.microsoft.com/en-us/library/ms952392.aspx

Inter-Tier Communication

Web Browser

HTTP

Web Pages and Presentation Logic

Rich (Thick) Client

Business Components

Data Access

Relational Databases

XML

Flat Files

Messaging Services

Presentation

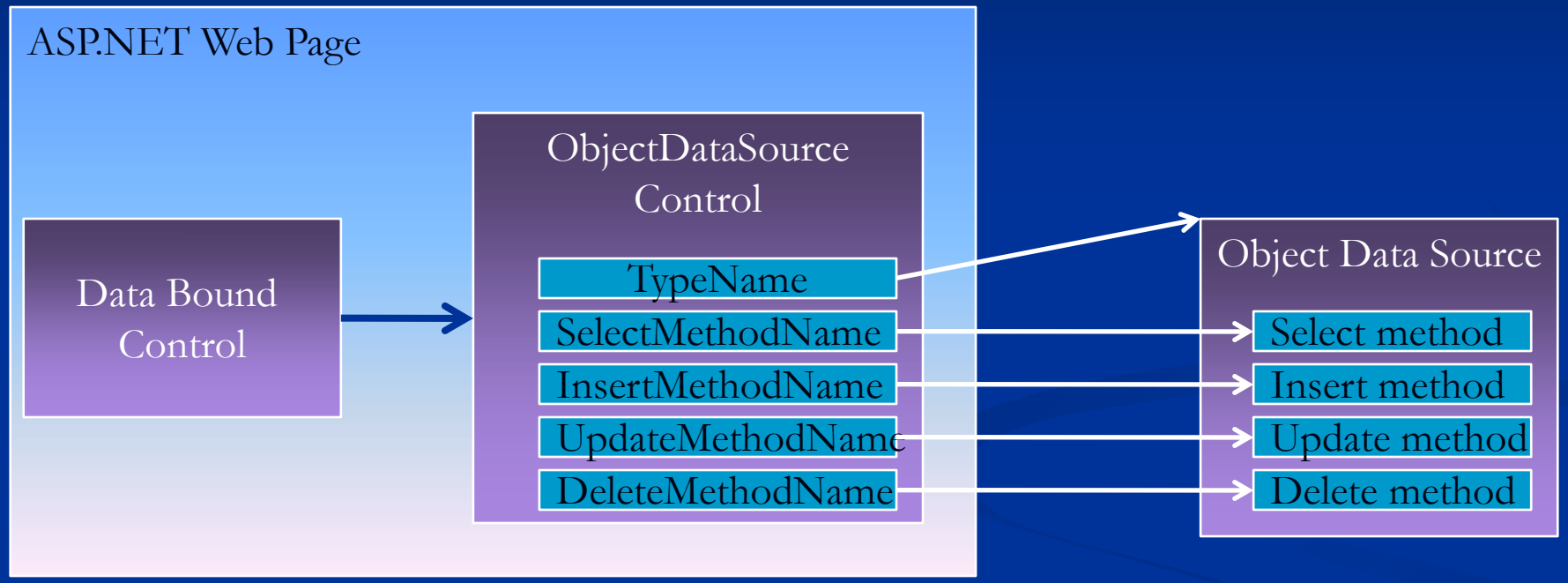
Business logic

Data services

Communication Options

	Same Machine	Different Server
Same Technology	easy in process direct linking	can use proprietary remote communication technologies
Different Technology	Need to use open communication standards e.g.: Web Services	

.NET Object Data Sources



Presentation Tier

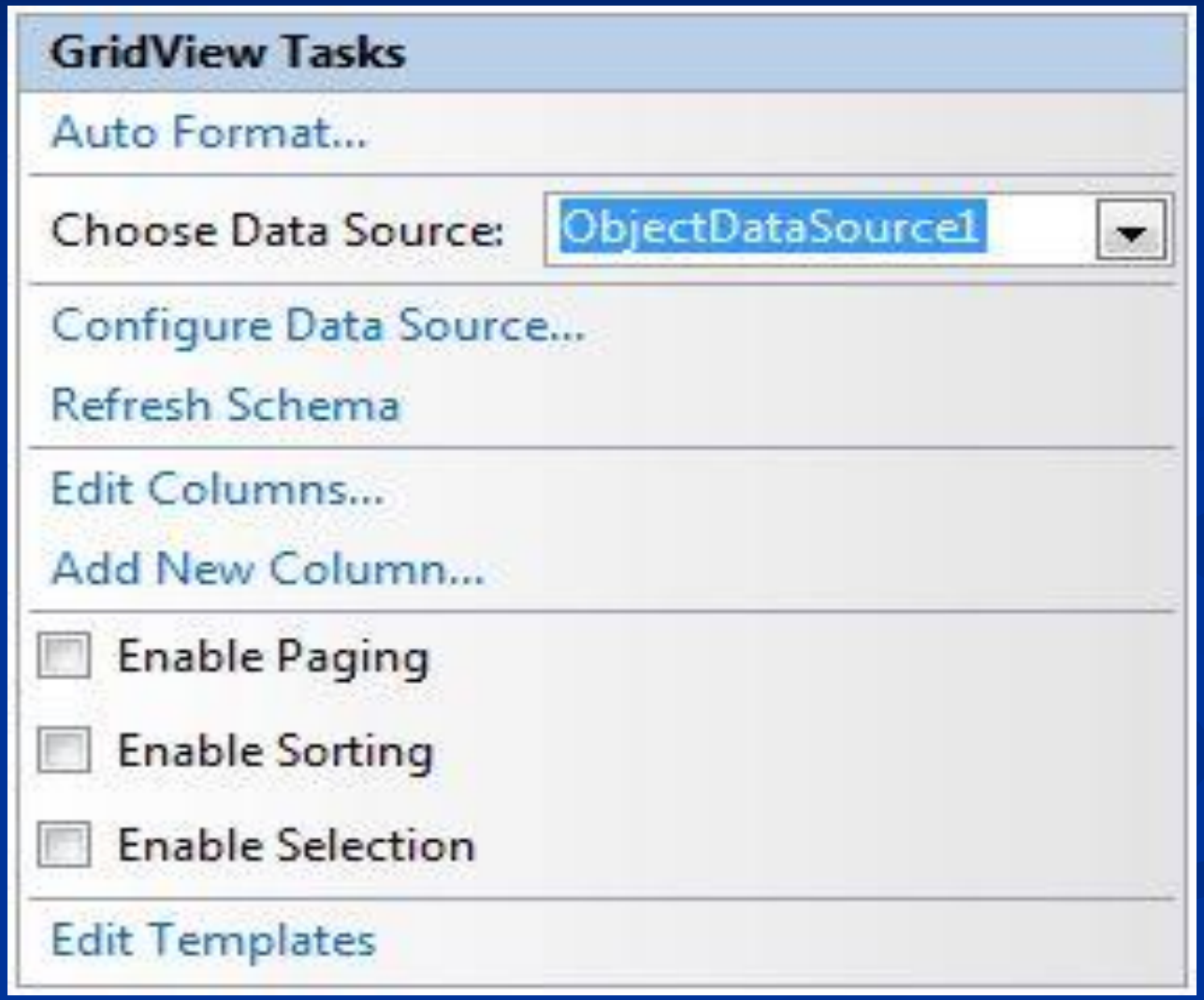
Business Tier

Object Data Source

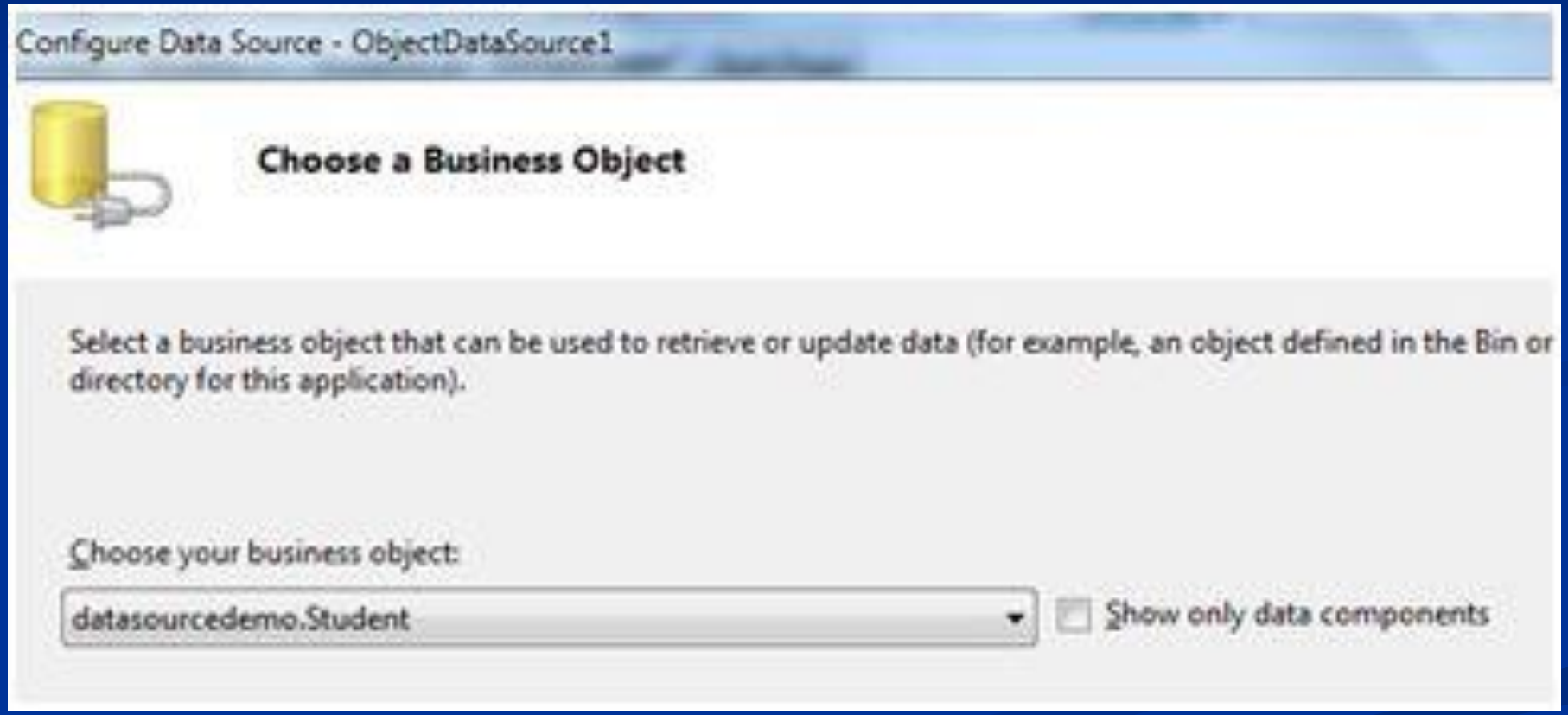
```
[DataObject(true)]
public class Business
{
    [DataObjectMethod(DataObjectMethodType.Select, true)]
    public List<Product> getProducts()
    {

    }

    [DataObjectMethod(DataObjectMethodType.Delete, true)]
    public void DeleteProduct(Product p)
    {

    }

    [DataObjectMethod(DataObjectMethodType.Update, true)]
    public void updateProduct(Product p)
    {

    }

    [DataObjectMethod(DataObjectMethodType.Insert, true)]
    public void insertProduct(Product p)
    {

    }
}
```

```
public class Product
{
    private int id;
    private string name;
    private decimal cost;

    [DataObjectField(true, true, false)]
    public int ID
    {
        get { return id; }
        set { id = value; }
    }

    [DataObjectField(false, false, false)]
    public string Name
    {
        get { return name;  }
        set { name = value; }
    }

    [DataObjectField(false, false, false)]
    public decimal Cost
    {
        get { return cost;  }
        set { cost = value; }
    }
}
```

Bind data bound control to object data source



Configure the data source – select object



Data Method Patterns