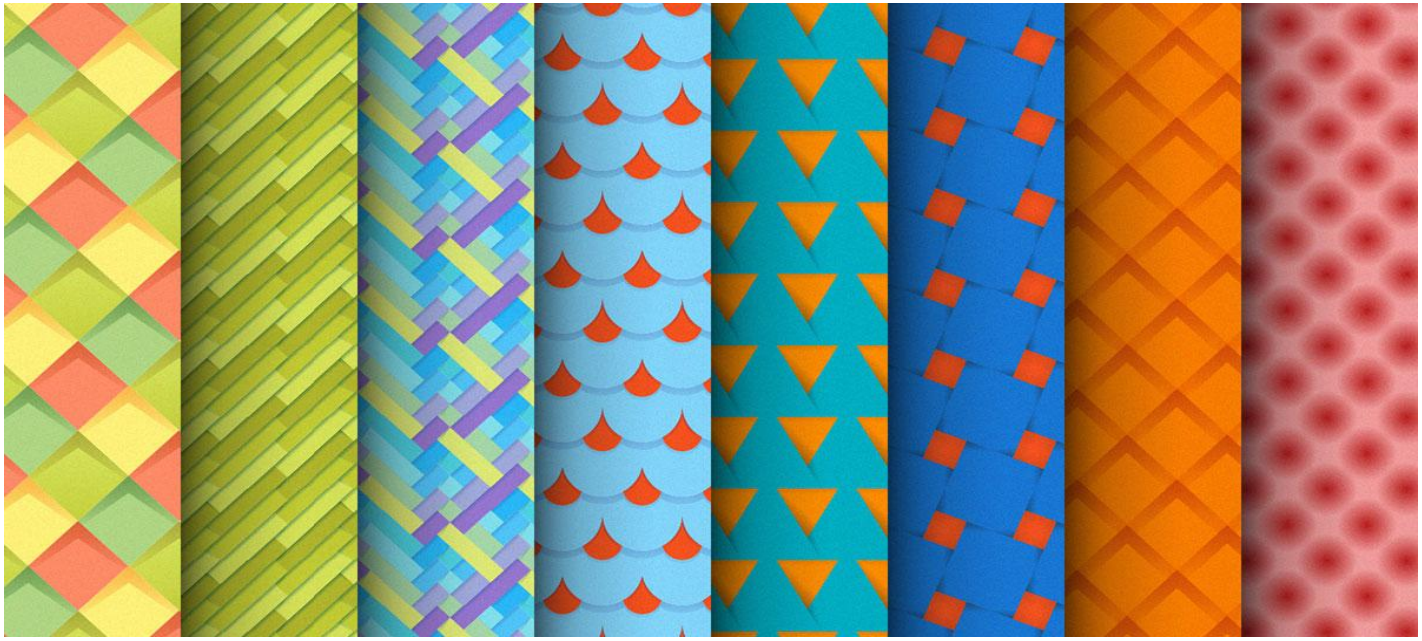
DESIGN PATTERNS

COURSE 9



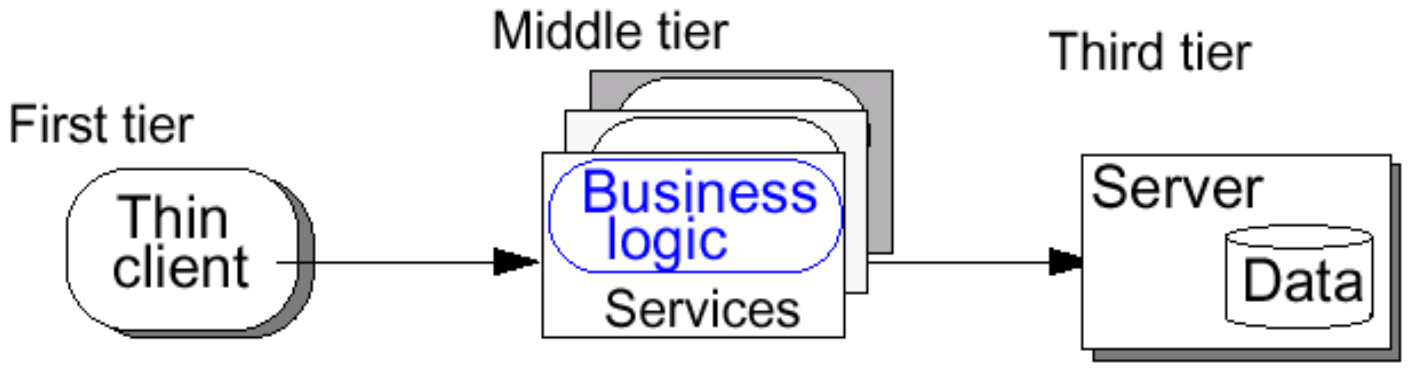
CONTENT

- Applications split on levels
- J2EE Design Patterns

APPLICATION SERVERS

- In the 90's, systems should be *client-server*

- Today, enterprise applications use the *multi-tier* model



APPLICATION SERVERS

- **“Multi-tier applications” have several independent components**
- **An *application server* provides the infrastructure and services to run such applications**
- **Application server products can be separated into 3 categories:**
  - J2EE-based solutions
  - Non-J2EE solutions (PHP, ColdFusion, Perl, etc.)
  - And the Microsoft solution (ASP/COM and now .NET with ASP.NET, VB.NET, C#, etc.)

J2EE APPLICATION SERVERS

- **Major J2EE products:**
  - BEA WebLogic
  - IBM WebSphere
  - Sun iPlanet Application Server
  - Oracle 9iAS
  - HP/Bluestone Total-e-Server
  - Borland AppServer
  - Jboss (free open source)

J2EE

- **It is a public specification that embodies several technologies**
- **Current version is 1.3**
- **J2EE defines a model for developing multi-tier, web based, enterprise applications with distributed components**
- **Benefits**
  - High availability
  - Scalability
  - Integration with existing systems
  - Freedom to choose vendors of application servers, tools, components
  - Multi-platform

J2EE

- **Main Components**
  - JavaServer Pages (JSP)
    - Used for web pages with dynamic content
    - Processes HTTP requests (non-blocking call-and-return)
    - Accepts HTML tags, special JSP tags, and scriptlets of Java code
    - Separates static content from presentation logic
    - Can be created by web designer using HTML tools
  - Servlet
  - Enterprise JavaBeans (EJB)

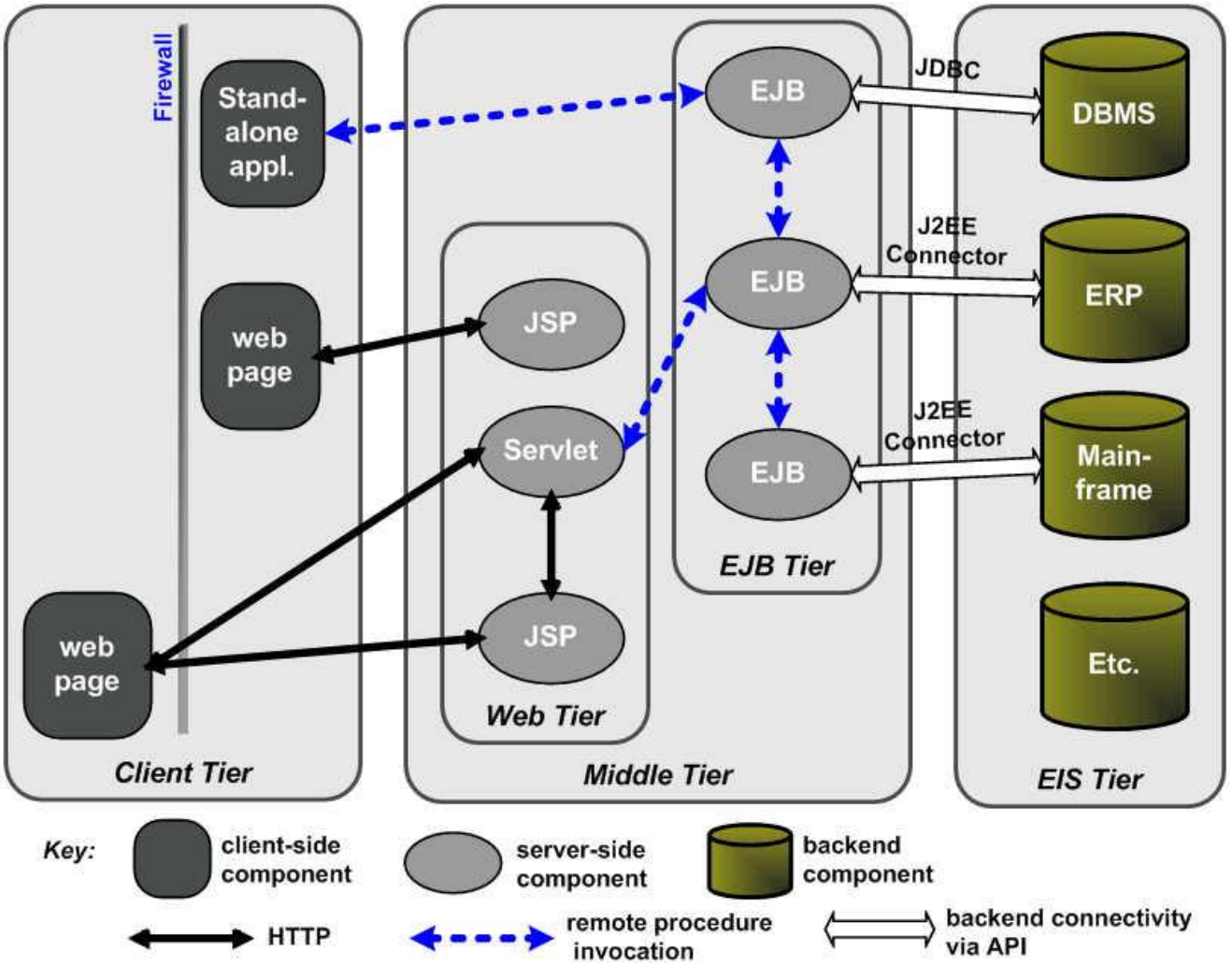
J2EE

- **Main Components**
  - JavaServer Pages (JSP)
  - Servlet
    - Used for web pages with dynamic content
    - Processes HTTP requests (non-blocking call-and-return)
    - Written in Java; uses print statements to render HTML
    - Loaded into memory once and then called many times
    - Provides APIs for session management
  - Enterprise JavaBeans (EJB)

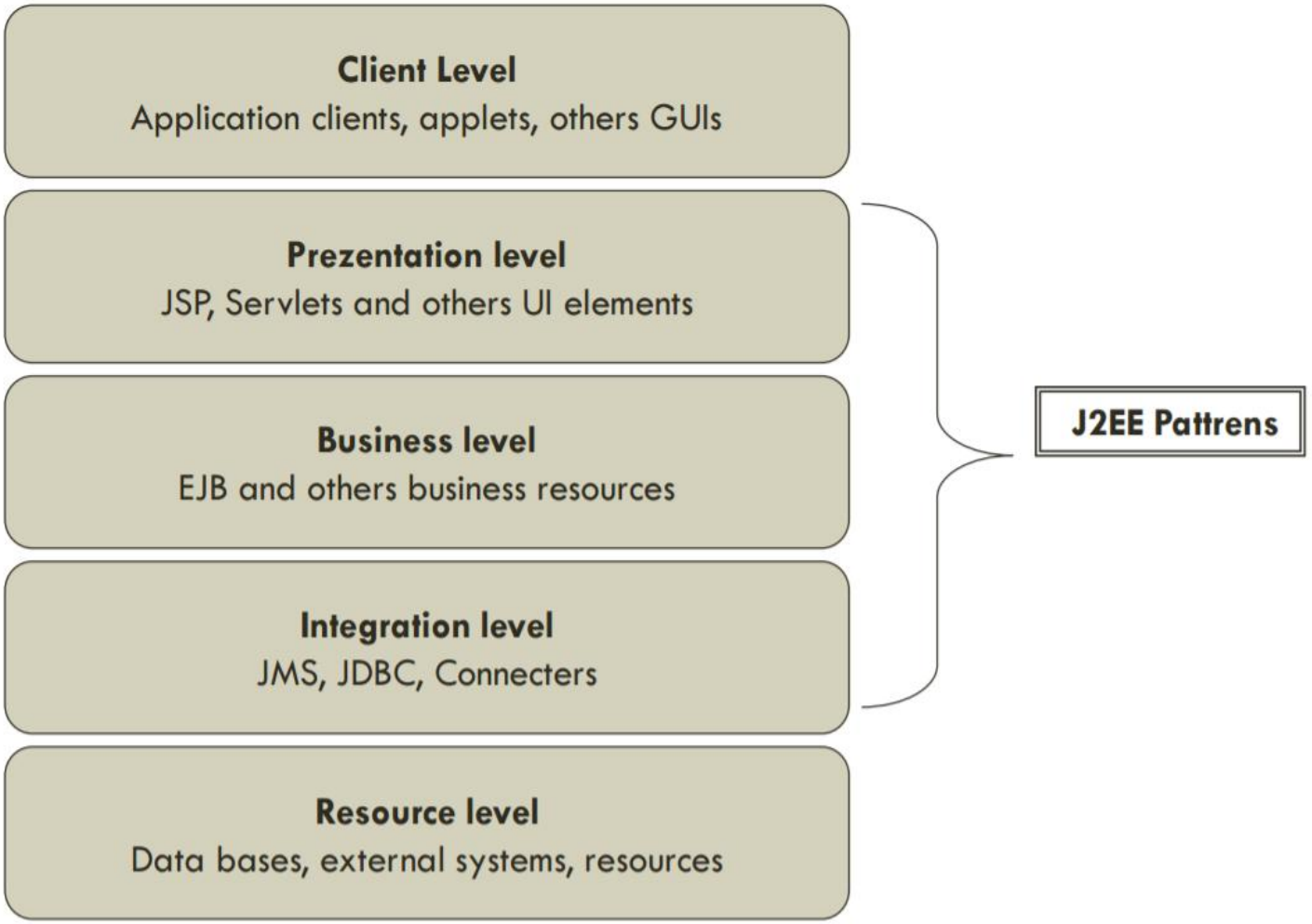
J2EE

- **Main Components**
  - JavaServer Pages (JSP)
  - Servlet
  - Enterprise JavaBeans (EJB)
    - EJBs are *distributed components* used to implement business logic (no UI)
    - Developer concentrates on business logic
    - Availability, scalability, security, interoperability and integrability handled by the J2EE server
    - Client of EJBs can be JSPs, servlets, other EJBs and external aplications
    - Clients see *interfaces*

J2EE

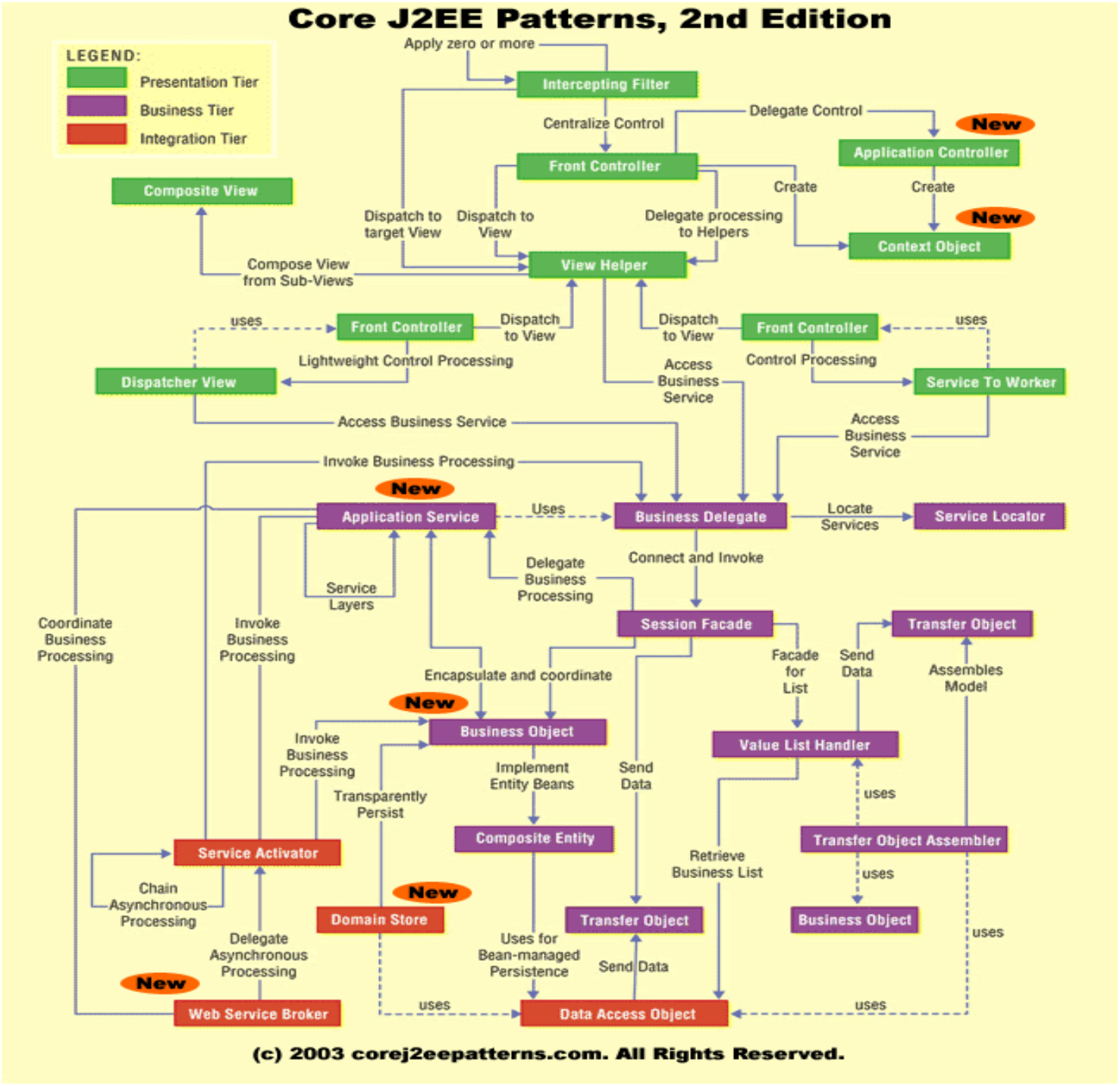

Firewall
Stand-alone appl.
web page
web page
JSP
Servlet
JSP
Web Tier
EJB
EJB
EJB
EJB Tier
JDBC
J2EE Connector
J2EE Connector
DBMS
ERP
Main-frame
Etc.
Client Tier
Middle Tier
EIS Tier
Key:
client-side component
server-side component
backend component
HTTP
remote procedure invocation
backend connectivity via API

APPLICATIONS SPLIT ON LEVELS



PATTERNS CLASSIFICATION

- Patterns applicable on presentation level
- Patterns applicable on business level
- Patterns applicable on integration level



PRESENTATION PATTERNS

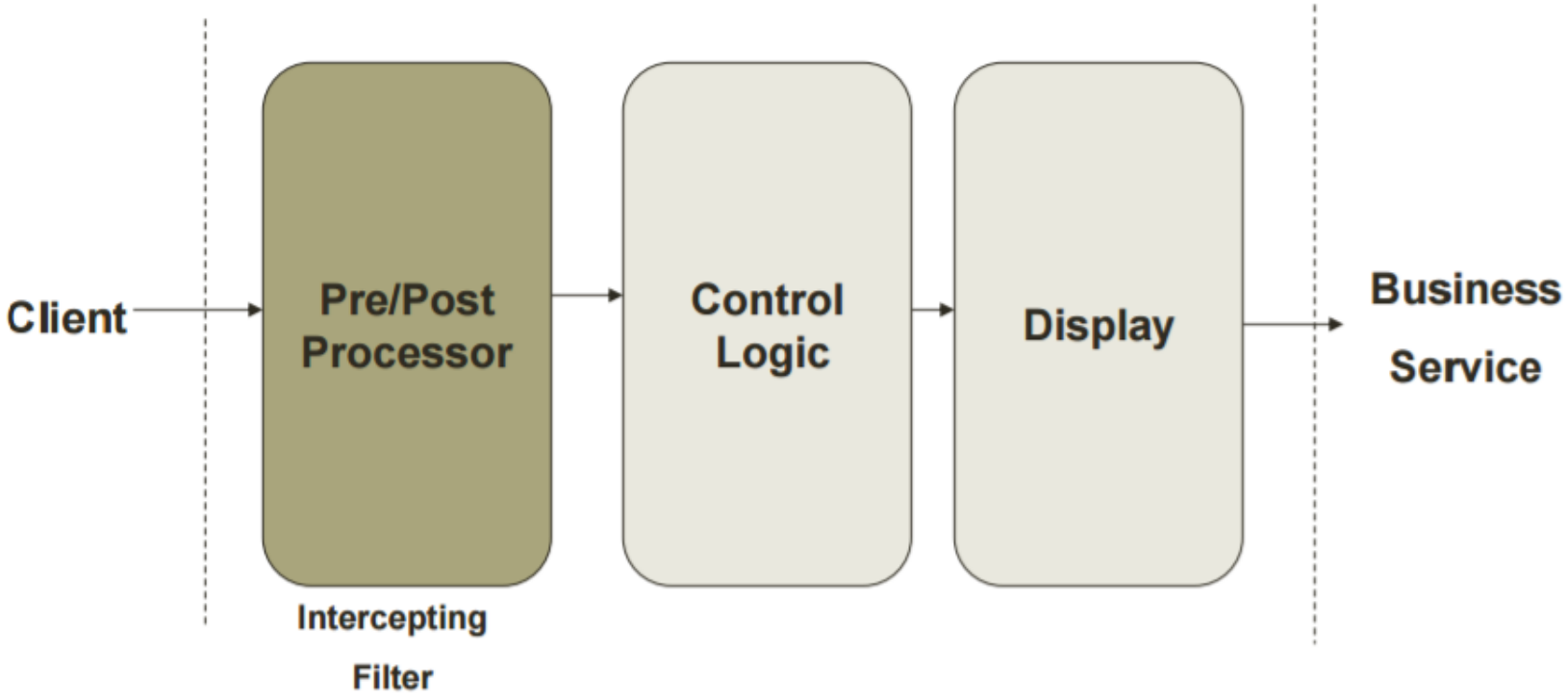
- Intercepting Filter
- Front Controller
- Context Object
- Application Controller
- View Helper
- Composite View
- Service to Worker
- Dispatcher View

PRESENTATIONS PATTERNS

- **Intercepting Filter**
- Front Controller
- Context Object
- Application Controller
- View Helper
- Composite View
- Service to Worker
- Dispatcher View

- **Facilitates pre/post request/respons processing**
- **Useful for security check, caching, packaging**
- **Independent chain filtering**

PRESENTATION PATTERNS. INTERCEPTING FILTER



PRESENTATION PATTERNS. INTERCEPTING FILTER

- **Problem**
  - You want to intercept and manipulate a request and a response before and after the request is processed.
  - Example

```
<%
 if (session.getAttribute("user") == null) {
 //redirect to a login page
 }
%>
```

- **Examples**
  - Has the client a valid session?
  - The request satisfies all the constraints?
  - What encoding is used to pass date?
  - Is the request stream encoded or compressed?
  - From which browser came the request?

PRESENTATION PATTERNS. INTERCEPTING FILTER

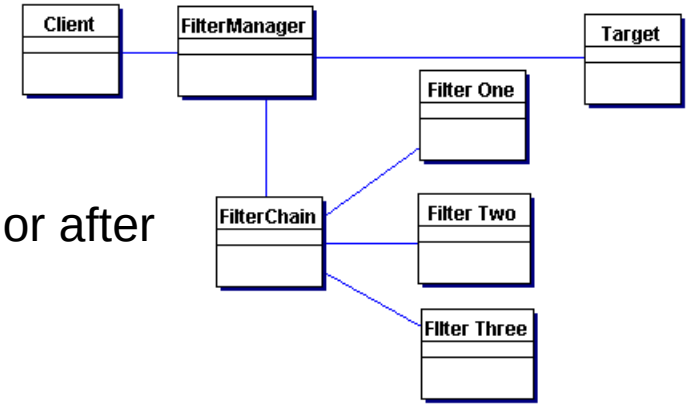
- **Forces**
  - You want centralized, common processing across requests, such as checking the data-encoding scheme of each request, logging information about each request, or compressing an outgoing response.
  - You want pre and postprocessing components loosely coupled with core requesthandling services to facilitate unobtrusive addition and removal.
  - You want pre and postprocessing components independent of each other and self contained to facilitate reuse.

PRESENTATION PATTERNS. INTERCEPTING FILTER

- **Solution**
  - Use an Intercepting Filter as a pluggable filter to pre and postprocess requests and responses. A filter manager combines loosely coupled filters in a chain, delegating control to the appropriate filter. In this way, you can add, remove, and combine these filters in various ways without changing existing code.

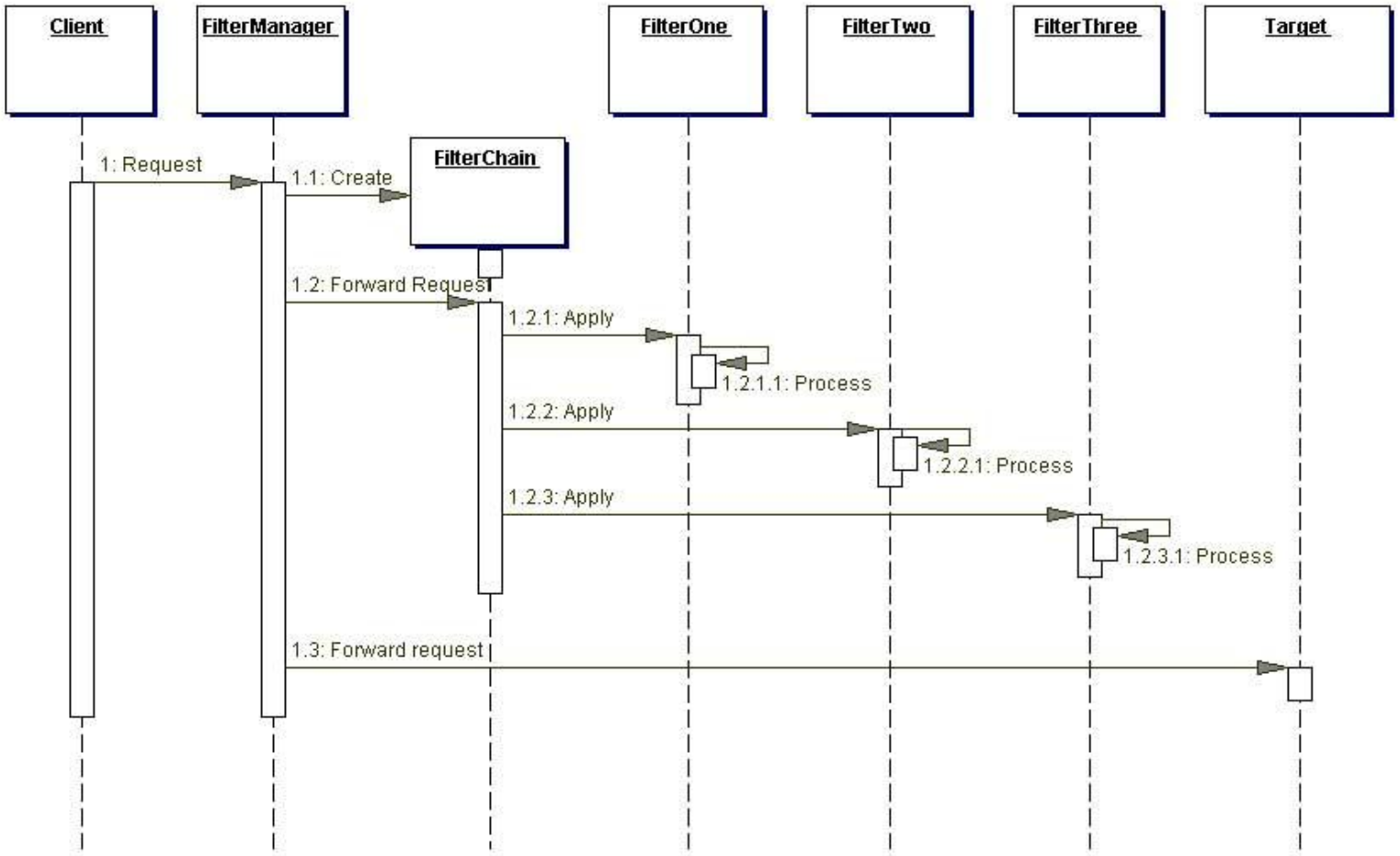
- **Examples**
  - Servlets Filter for HTTP request/responce
  - Message Handles for SOAP request/responce

PRESENTATION PATTERNS. INTERCEPTING FILTER. STRUCTURE



- Filter
  - Filter which will performs certain task prior or after execution of request by request handler.
- Filter Chain
  - Filter Chain carries multiple filters and help to execute them in defined order on target.
- Target
  - Target object is the request handler
- Filter Manager
  - Filter Manager manages the filters and Filter Chain.
- Client
  - Client is the object who sends request to the Target object.

PRESENTATION PATTERNS. INTERCEPTING FILTER



PRESENTATION PATTERNS. INTERCEPTING FILTER

- **Strategies**
  - Standard filters (Servlet 2.3)
    - Components used in deployment descriptor
  - Personalized filters
  - Base filters
  - Filters based on templates

PRESENTATION PATTERNS. INTERCEPTING FILTER

- **Exemple**
  - Standard Filter

```
public class HelloWorldFilter implements Filter {
public FilterConfig filterCfg;
 @Override
 public void destroy() { }

 @Override
 public void init(FilterConfig filterCfg) throws ServletException { this.filterCfg = filterCfg;}

 public void doFilter(final ServletRequest request, final ServletResponse response,
            FilterChain chain) throws java.io.IOException, javax.servlet.ServletException {
              System.out.println("In Filter");
              request.setAttribute("hello", "Hello World!");
              chain.doFilter(request, response);
              System.out.println("Out HelloWorldFilter");
} }
```

PRESENTATION PATTERNS. INTERCEPTING FILTER

- **Exemple**
  - Standard Filte
  - Deployment descriptor

```
<filter>
 <filter-name>helloWorld</filter-name>
 <filter-class>web.filter.baseStrategy.HelloWorldFilter </filter-class>
</filter>
<filter-mapping>
 <filter-name>helloWorld</filter-name>
 <url-pattern>/jsp/filter.jsp</url-pattern>
</filter-mapping>
```

PRESENTATION PATTERNS. INTERCEPTING FILTER

- **Exemple**
  - Template filters focus on pre/post processing

```
public abstract class TemplateFilter implements Filter {
    private FilterConfig filterCfg;
    public void init(FilterConfig filterCfg) throws ServletException {
        this.filterCfg = filterCfg;
    }
    public void doFilter(ServletRequest request, ServletResponse response,
                         FilterChain chain) throws IOException,ServletException {
        doPreProcessing(request, response);
        chain.doFilter(request, response);
        doPostProcessing(request, response);
    }
    public abstract void doPostProcessing(ServletRequest request,
                                          ServletResponse response);
    public abstract void doPreProcessing(ServletRequest request,
                                         ServletResponse response);
    public void destroy() { }
}
```

PRESENTATION PATTERNS. INTERCEPTING FILTER

- **Consequences**
  - Centralizes control with loosely coupled handlers
  - Improves reusability
  - Declarative and flexible configuration
  - Information sharing is inefficient

PRESENTATIONS PATTERNS

- Intercepting Filter
- **Front Controller**
- Context Object
- Application Controller
- View Helper
- Composite View
- Service to Worker
- Dispatcher View

- **Offers a centerlasid controller for request management**
- **Enter point in system. It does not have to be too large (it delegates attributes to Application Controller)**
- **Request processing:**
  1. Protocol handeling and context transformations
  2. Navigation and routing
  3. Request pocessing
  4. Control transfer

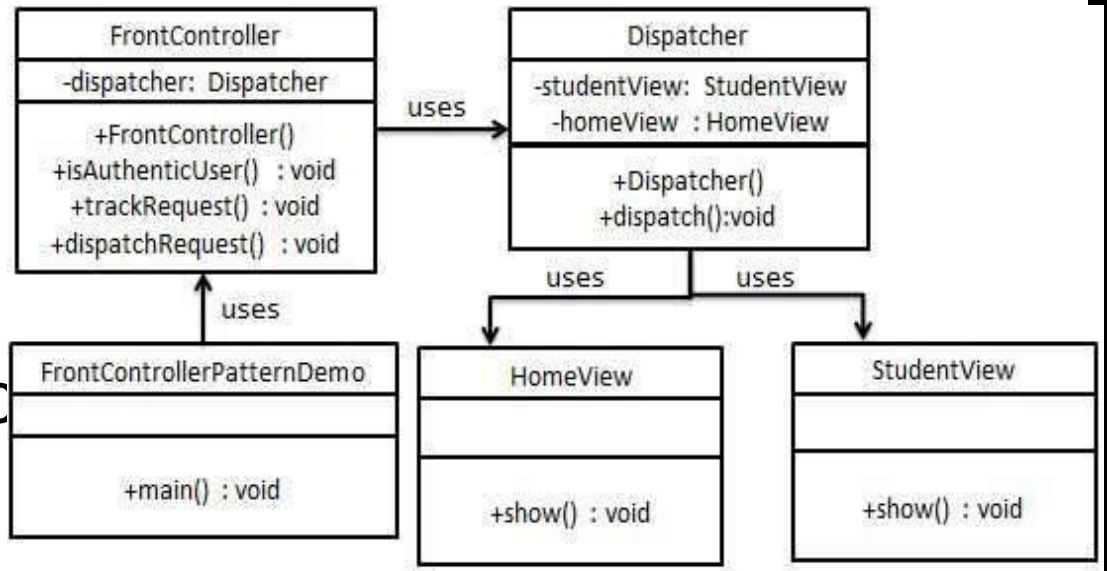
PRESENTATIONS PATTERNS

- Intercepting Filter
- **Front Controller**
- Context Object
- Application Controller
- View Helper
- Composite View
- Service to Worker
- Dispatcher View

- **Offers a centerlasid controller for request management**
- **Enter point in system. It does not have to be too large (it delegates attributes to Application Controller)**
- **Request processing:**
  1. Protocol handeling and context transformations
  2. Navigation and routing
  3. Request pocessing
  4. Control transfer

PRESENTATIONS PATTERNS

- Intercepting Filter
- **Front Controller**
- Context Object
- Application Contro
- View Helper
- Composite View
- Service to Worker
- Dispatcher View



PRESENTATIONS PATTERNS

- Intercepting Filter
- Front Controller
- **Context Object**
- Application Controller
- View Helper
- Composite View
- Service to Worker
- Dispatcher View

- **Incapsulates the state in order to avoid using protocol-specific system information outside of its relevant context**
- **The application componesnts does not have to know HTTP protocol. They must call methods getXX on an object of the context.**
- **Struts - ActionForm**

PRESENTATIONS PATTERNS

- Intercepting Filter
- Front Controller
- Context Object
- **Application Controller**
- View Helper
- Composite View
- Service to Worker
- Dispatcher View

- **You want to centralize and modularize action and view management.**
- **You want to reuse action and view-management code.**
- **You want to improve request-handling extensibility, such as adding use case functionality to an application incrementally.**
- **You want to improve code modularity and maintainability, making it easier to extend the application and easier to test discrete parts of your request-handling code independent of a web container.**
- **Struts**

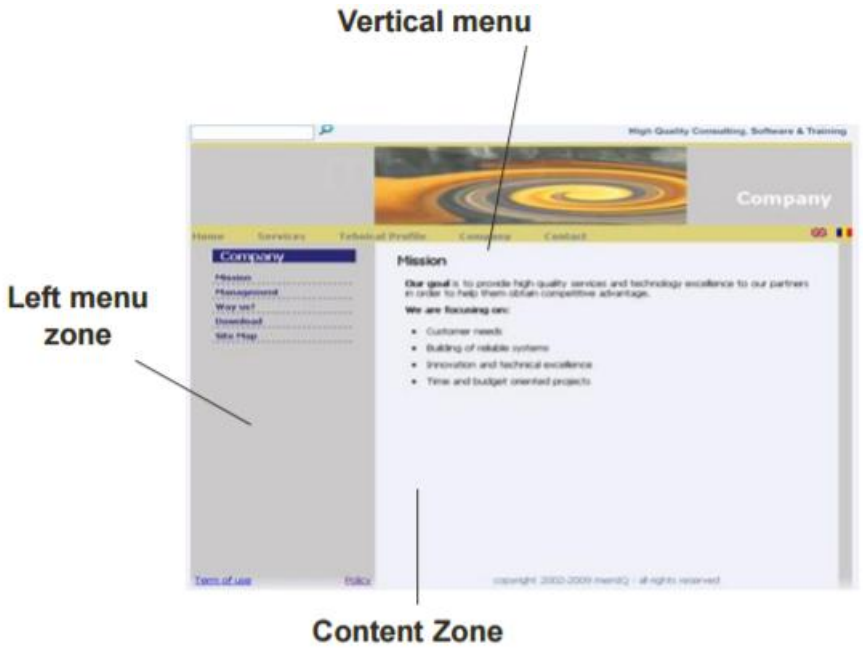
PRESENTATIONS PATTERNS

- Intercepting Filter
- Front Controller
- Context Object
- Application Controller
- **View Helper**
- Composite View
- Service to Worker
- Dispatcher View

- **You want to separate a view from its processing logic.**
- **You want to use template-based views, such as JSP.**
- **You want to avoid embedding program logic in the view.**
- **You want to separate programming logic from the view to facilitate division of labor between software developers and web page designers.**
- **Expresion Language, JSLT**

PRESENTATIONS PATTERNS

- Intercepting Filter
- Front Controller
- Context Object
- Application Controller
- View Helper
- **Composite View**
- Service to Worker
- Dispatcher View



- **Creates and agregates views from atomic components**
- **Tiles**

PRESENTATIONS PATTERNS

- **Intercepting Filter**
- Front Controller
- Context Object
- Application Controller
- View Helper
- Composite View
- **Service to Worker**
- Dispatcher View

- **You want to perform core request handling and invoke business logic before control is passed to the view**

- **Use Service to Worker to centralize control and request handling to retrieve a presentation model before turning control over to the view. The view generates a dynamic response based on the presentation model.**

PRESENTATIONS PATTERNS

- Intercepting Filter
- Front Controller
- Context Object
- Application Controller
- View Helper
- Composite View
- Service to Worker
- **Dispatcher View**

- **You want a view to handle a request and generate a response, while managing limited amounts of business processing.**

- **Use Dispatcher View with views as the initial access point for a request. Business processing, if necessary in limited form, is managed by the views.**

BUSINESS PATTERNS

- **Business Delegate**
- Service Locator
- Session Facade
- Application Service
- Business Object
- Composite Entity
- Transfer Object
- Transfer Object Assembler
- Value List Handler

- **You want to hide clients from the complexity of remote communication with business service components.**
- **You want to access the business-tier components from your presentation-tier components and clients, such as devices, web services, and rich clients.**
- **You want to minimize coupling between clients and the business services, thus hiding the underlying implementation details of the service, such as lookup and access.**
- **You want to avoid unnecessary invocation of remote services.**
- **You want to translate network exceptions into application or user exceptions.**
- **You want to hide the details of service creation, reconfiguration, and invocation retries from the clients.**

BUSINESS PATTERNS

- Business Delegate
- **Service Locator**
- Session Facade
- Application Service
- Business Object
- Composite Entity
- Transfer Object
- Transfer Object Assembler
- Value List Handler

- **You want to transparently locate busines components and services in a uniform manner**
- **Use a Service Locator to implement and encapsulate service and component lookup.**
- **A Service Locator hides the implementation details of the lookup mechanism and encapsulates related dependencies.**

BUSINESS PATTERNS

- Business Delegate
- Service Locator
- **Session Facade**
- Application Service
- Business Object
- Composite Entity
- Transfer Object
- Transfer Object Assembler
- Value List Handler

- **You want to expose business components and services to remote clients.**
- **Use a Session Façade to encapsulate business-tier components and expose a coarse-grained service to remote clients.**
- **Clients access a Session Façade instead of accessing business components directly**

BUSINESS PATTERNS

- Business Delegate
- Service Locator
- Session Facade
- **Application Service**
- Business Object
- Composite Entity
- Transfer Object
- Transfer Object Assembler
- Value List Handler

➢ **You want to centralize business logic across several business-tier components and services**

➢ **Use an Application Service to centralize and aggregate behavior to provide a uniform service**

BUSINESS PATTERNS

- Business Delegate
- Service Locator
- Session Facade
- Application Service
- **Business Object**
- Composite Entity
- Transfer Object
- Transfer Object Assembler
- Value List Handler

- **You have a conceptual domain model with business logic and relationship**
- **Use Business Objects to separate business data and logic using an object model.**

BUSINESS PATTERNS

- Business Delegate
- Service Locator
- Session Facade
- Application Service
- Business Object
- **Composite Entity**
- Transfer Object
- Transfer Object Assembler
- Value List Handler

- **You want to use entity beans to implement your conceptual domain model.**
- **Use a Composite Entity to implement persistent Business Objects using local entity beans and POJOs.**
- **Composite Entity aggregates a set of related Business Objects into coarse-grained entity bean implementations.**

BUSINESS PATTERNS

- Business Delegate
- Service Locator
- Session Facade
- Application Service
- Business Object
- Composite Entity
- **Transfer Object**
- Transfer Object Assembler
- Value List Handler

- **You want to transfer multiple data elements over a tier**

- **Use a Transfer Object to carry multiple data elements across a tier.**

BUSINESS PATTERNS

- Business Delegate
- Service Locator
- Session Facade
- Application Service
- Business Object
- Composite Entity
- Transfer Object
- **Transfer Object Assembler**
- Value List Handler

➢ **You want to obtain an application model that aggregates transfer objects from several business components.**

BUSINESS PATTERNS

- Business Delegate
- Service Locator
- Session Facade
- Application Service
- Business Object
- Composite Entity
- Transfer Object
- Transfer Object Assembler
- **Value List Handler**

- **You have a remote client that wants to iterate over a large results list.**
- **Use a Value List Handler to search, cache the results, and allow the client to traverse and select items from the results**

BUSINESS PATTERNS

- Business Delegate
- Service Locator
- Session Facade
- Application Service
- Business Object
- Composite Entity
- Transfer Object
- Transfer Object Assembler
- **Value List Handler**

➢ **Use a Value List Handler to search, cache the results, and allow the client to traverse and select items from the results**

BUSINESS PATTERNS

- **Business Delegate** ➢ **Will be discussed in detail at the next course**
- **Service Locator**
- **Session Facade**
- Application Service
- Business Object
- Composite Entity
- Transfer Object
- Transfer Object Assembler
- Value List Handler

INTEGRATION PATTERNS

- Data Access Object
- Service Activator
- Domain Store
- Web Service Broker