

Agenda

1. Context
2. The problem
3. Standard Patterns
4. The Energy Community Modular Infrastructure

Context

- ▶ Sophisticated Web pages present content from numerous data sources
- ▶ A single display page usually includes multiple sub-views
- ▶ A variety of individuals with different skills contribute to the development and maintenance of these Web pages

EnergyCommunity

Crusotto XYZ123

Temperatura

Elettricità

Nuovo cusco Crea

Gestione sensori

Gestione utenti

Crusotto XYZ123

Bacheca

- Sublime deerat caelum
- ligavit: super videre
- securae campos.

Post on update... Post

Temperatura

197 x 100

temp. corrente = 21 °C

minima = 18

massima = 24

e-Lite

The e-Lite Research Group

The e-Lite research group develops intelligent technologies applied to distributed applications. The spirit of the research is integration of complex systems, based on interaction between a human user and the technology, in which the complexity gap is managed by intelligent software components.

The main research areas are:

- [Intelligent Web Applications](#): Web 2.0 and Web 3.0 search engines, classification systems, user interfaces based on intelligent algorithms and semantic elaboration techniques for new generations of web applications
- [Smart Home and Domestic Intelligent Systems](#): integration of domestic plants, residential gateways, intelligent "domestic" agents, environmental control interfaces, formal property verification on the house status
- [Accessibility and Technical Aids](#): open platform for developing computer aids for disabled persons (communication, web browsing, ...), eye-tracking and head-tracking technology for hands-free usage of the computer

TOOLS

- H-DOSE
- DOG
- DogOnt
- DOGSim

LOGIN FORM

Username

Password

Remember Me

LOGIN

- [Forgot your password?](#)
- [Forgot your username?](#)

LATEST NEWS

Il Web Semantico applicato agli edifici intelligenti

Presentazione: Formal Verification of Device State Chart Models

Visual Builder for Domestic Rules

PAST SURVEYS

- Intelligent Home survey results
- Intelligent Home Survey - Control Group

LATEST UPDATES

- [iGCM - Sistemi Informativi Aziendali](#)
- [Thesis / Tesi: Stream Processing for Real-time Intelligent Sensor Data Processing](#)
- [Thesis / Tesi: Support of the NIX protocol in the Open Source DoS2 gateway](#)
- [iGCM - E-Business Web](#)
- [ZigBee4ms](#)
- [Thesis / Tesi: Visual ICE domestic modeler and simulator](#)
- [Thesis / Tesi: ZigBee-based power control system](#)
- [Il Web Semantico applicato agli edifici intelligenti](#)
- [Fokus Como](#)
- [Luca De Russis](#)
- [Current events](#)

POPULAR KEYWORDS

fulvio

come

dose

esercitazione

semantic

2009

datario

platform

aula

esercizi

homino

introduzione

service

thesis

corso

file

esercizi

research

user

soluzione

2010

search

esante

domestic

laborations

applications

Context (2)

- ▶ Multiple pages use similar sub-views,
 - ▶ decorated with different surrounding text,
 - ▶ in a different location within the page
- ▶ Sub-view content might change frequently

[CommonHeader] EnergyCommunity Pinco Pallino ACME corporation Logout

74 x 76

Cruscotto XYZ123 Edit

Temperatura
Elettricità

Nuovo cruscotto Crea

Gestione sensori
Gestione utenti

Cruscotto XYZ123 Edit

Bacheca miei/sede/azienda

- Sublime deerat caelum
- ligavit: super videre
- securae campos.

Post an update... Post

Temperatura

197 x 100

temp. corrente = 21 °C

minima = 18
massima = 24

[CommonHeader] EnergyCommunity Pinco Pallino ACME corporation Logout

74 x 76

Cruscotto XYZ123 Rinomina Elimina

Done

Pubblico/Privato

Elimina cruscotto

Temperatura Rinomina Elimina

Sensore 37 del piano 2 della portineria Medio su 1 ora Grafico

Bacheca (nessuna impostazione)

Consumo illuminazione magazzino Valore istantaneo (kW)

Cerca nuovo sensore Aggiungi

Problems

- ▶ **2 main problems to address:**
 - ▶ Effective sub-view implementation and composition
 - ▶ Effective sub-view management (logic)

Sub-view (1 / 2)

- ▶ Sub-views are “implemented” by embedding formatting code directly within each display page
- ▶ Shortcomings:
 - ▶ Layout changes are difficult to manage
 - ▶ Code is harder to maintain and duplicated in multiple views.
 - ▶ Embedding frequently changing portions of template text potentially affects the availability and administration of the system.
 - ▶ The server may need to be restarted before clients see the modifications or updates to any sub-view.

Sub-view (2 / 2)

- ▶ **A more effective solution is needed**
 - ▶ Modular
 - ▶ Sub-views are designed/deployed/maintained separately
 - ▶ Sub-views are composed into “composite” pages
 - ▶ Composite pages change depending on the user/application context
 - ▶ Sub-views are displayed (or not)
 - ▶ Sub-view positions change
 - ▶ ...

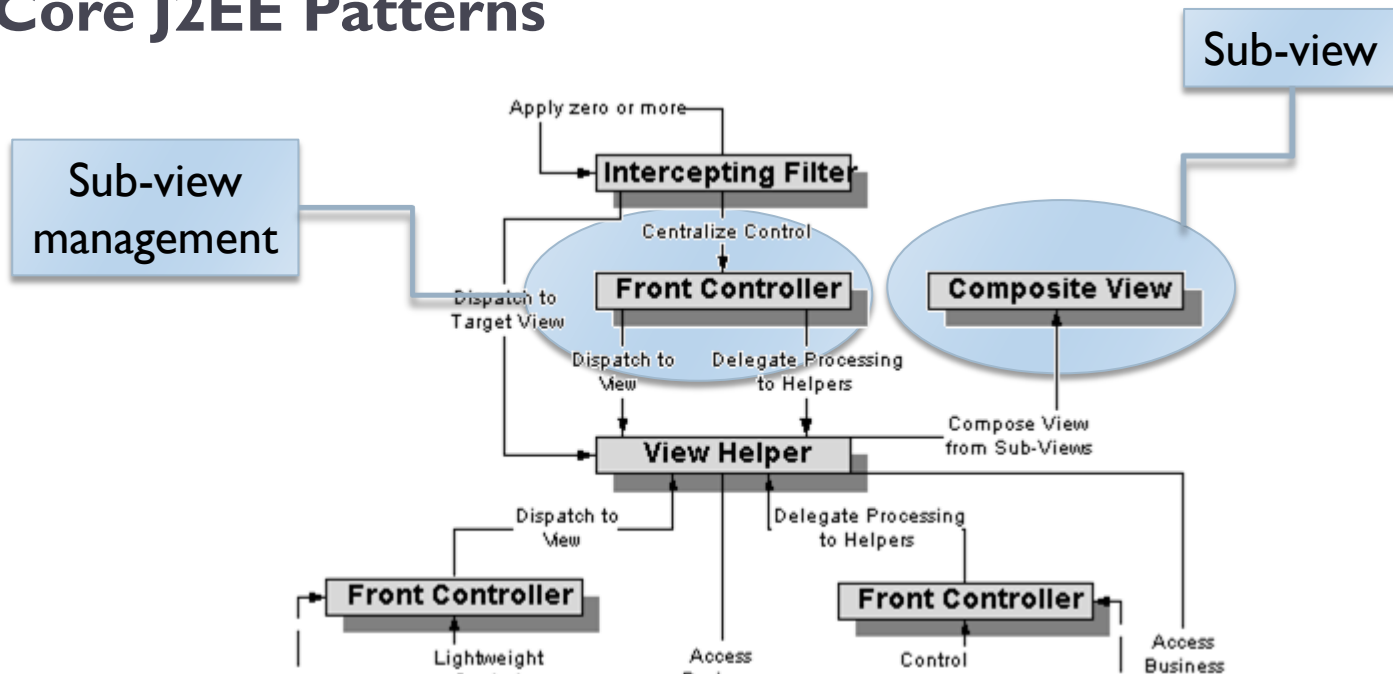
Sub-View Management

▶ Problem

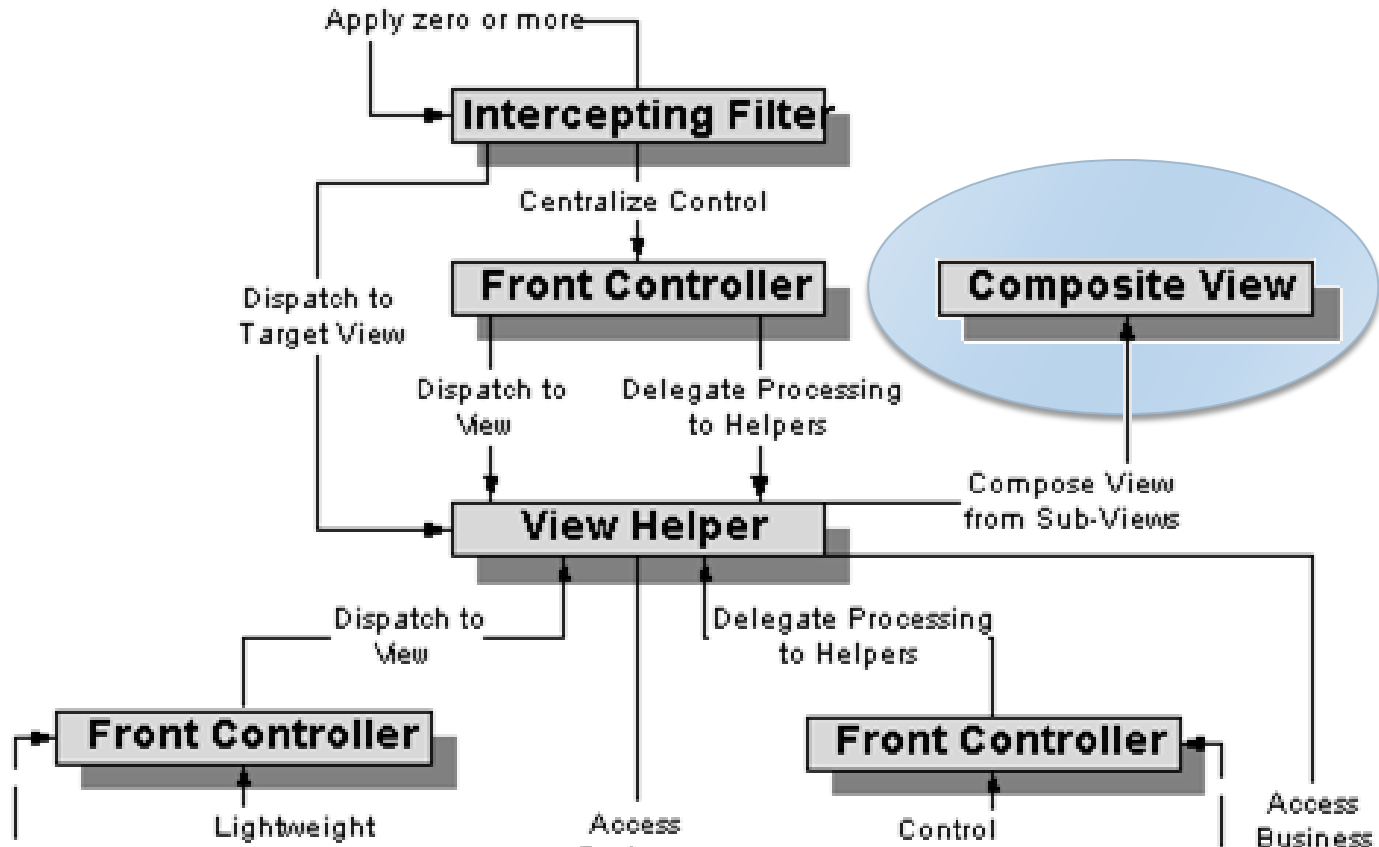
- ▶ The system requires a centralized access point
 - ▶ to support the integration of system services, content retrieval, view management, and navigation.
- ▶ Accessing the view directly without going through a centralized mechanism, leads to 2 problems:
 - ▶ Each view is required to provide its own system services
 - often resulting in duplicate code.
 - ▶ View navigation is left to the views.
 - This may result in commingled view content and view navigation.
- ▶ Distributed control is more difficult to maintain
 - ▶ changes numerous places.

Design patterns?

- ▶ Is this problem new?
- ▶ Are there any **VALIDATED** design patterns addressing such a need?
 - ▶ **Core J2EE Patterns**



Composite View



Composite View

- ▶ Defined in the ORACLE developer resources
 - ▶ <http://java.sun.com/blueprints/corej2eepatterns/Patterns/CompositeView.html>
- ▶ Provides best practices for addressing composite view design and implementation

ORACLE Sun Developer Network (SDN)

SDN Home - Products & Technologies - Java Technology - J2EE - Reference - Blueprints - Welcome to Core J2EE Patterns - Core J2EE Pattern Catalog

Core J2EE Pattern Catalog
Core J2EE Patterns - Composite View

Context
Sprintster: Web pages present content from numerous data sources, using multiple subviews that comprise a single display page. Additionally, a variety of individuals with different skill sets contribute to the development and maintenance of these Web pages.

Problem
Instead of providing a mechanism to combine modular, atomic portions of a view into a composite view, pages are built by embedding formatting code directly within each view.
Modification to the layout of multiple views is difficult and error prone, due to the duplication of code.

Forces
Atomic portions of view content change frequently.
Multiple composite views use similar subviews, such as a customer hierarchy table. These atomic portions are associated with different surrounding templates, or the appear in a different location within the page.
Layout changes are more difficult to manage and code harder to maintain when subviews are directly embedded and duplicated in multiple views.
Embedding frequently changing portions of template code directly into views also potentially affects the usability and administration of the system. The server may need to be restarted before clients see the modifications or updates to these template components.

Solution
Use composite views that are composed of multiple atomic subviews. Each component of the template may be included dynamically into the whole and the layout of the page may be managed independently of the content.
The solution provides for the creation of a composite view based on the inclusion and substitution of modular dynamic and static template fragments. It promotes the reuse of atomic portions of the view by encouraging modular design. It is appropriate to use a composite view to generate pages containing display components that may be combined in a variety of ways. This scenario occurs, for example, with portal sites that include numerous independent subviews, such as news feeds, weather information and stock quotes on a single page. The layout of the page is managed and modified independent of the subview content.
Another benefit of the pattern is that view designers can prototype the layout of a site, plugging static content into each of the template regions. As site development progresses, the actual content is substituted for these placeholders.

Figure 7.17 shows a screen capture of Sun's Java Software homepage, java.sun.com. Four regions are identified: Navigation, Search, Feature Story and headlines. While the content for each of these component subviews may originate from different data sources, they are all out seamlessly to create a single composite page.



Figure 7.17 Screen shot of a modular page, including Search, Navigation, Feature Story and headlines regions

The system is not without its drawbacks. There is a runtime overhead associated with a "placeholder" for the increased flexibility that it provides. Also, the use of a more sophisticated layout mechanism brings with it some manageability and development issues, since there are more artifacts to maintain and a need of implementation instruction to understand.

Structure

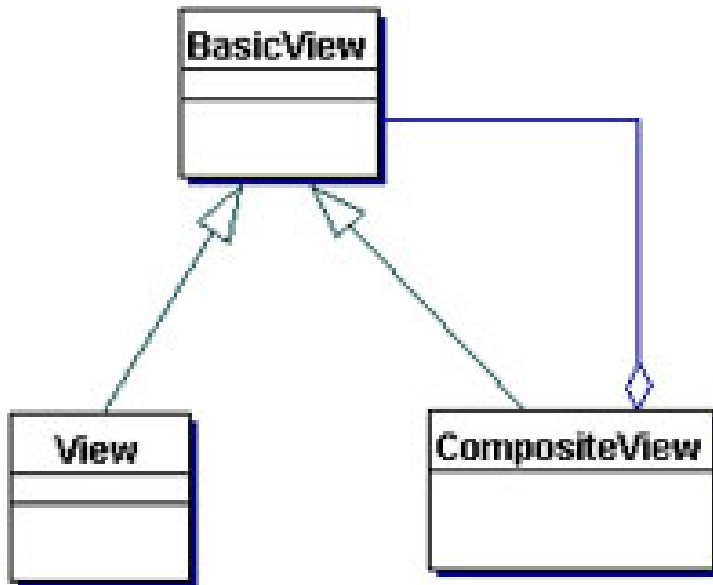
Figure 7.18 shows the class diagram that represents the Composite View pattern.



```
classDiagram
    class CompositeView
    class View
    class CompositeView
    CompositeView --> View
    CompositeView --> CompositeView
```

Composite View - basics

▶ Involved entities



▶ Composite View

- ▶ A composite view is a view that is an aggregate of multiple sub-views.

▶ Included View

- ▶ A sub-view that is one atomic piece of a larger whole view.
- ▶ This included view could also potentially be a composite, itself including multiple sub-views.

Composite view - participants

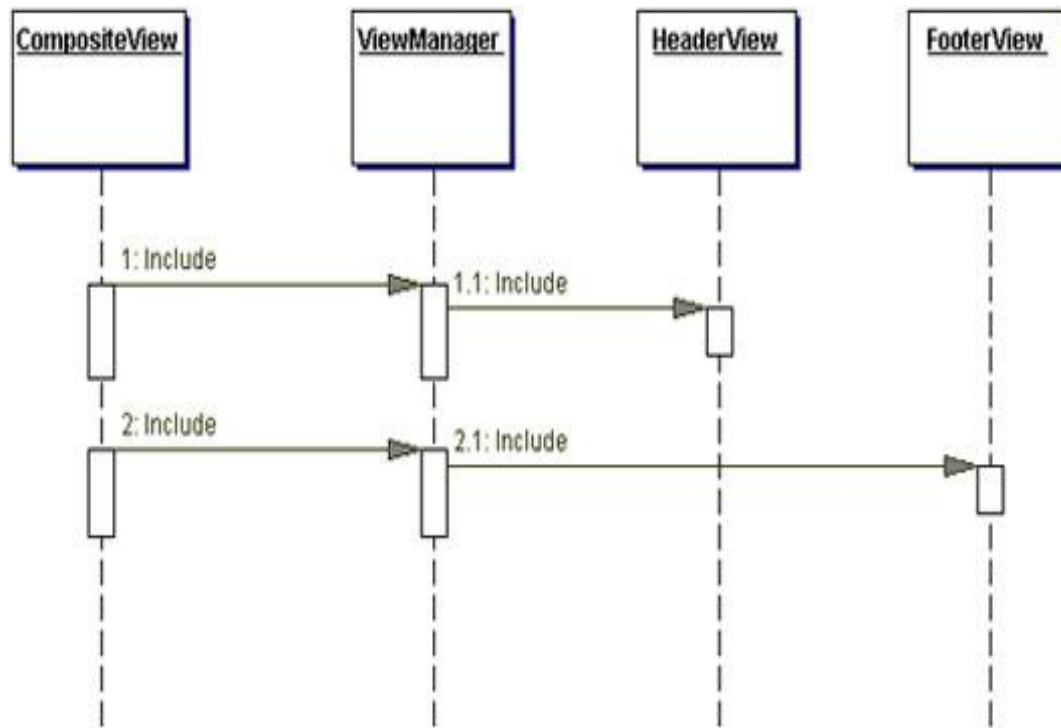
▶ View Manager

▶ Manages the inclusion of portions of template fragments into the composite view.

- ▶ JavaServer Pages (JSP page) pages include tag (<jsp:include>)
- ▶ JavaBean helper (JSP page 1.0+)
- ▶ Custom tag helper (JSP page 1.1+)

Allows for more sophisticated control of the page structure

- ▶ Useful for creating reusable page layouts.



JSP strategy

```
<table border=1 valign="top" cellpadding="2%"
width="100%">
  <tr>
    <td><jsp:include page="news/worldnews.jsp"
      flush="true"/> </td>
  </tr>
  <tr>
    <td><jsp:include page="news/countrynews.jsp"
      flush="true"/> </td>
  </tr>
  <tr>
    <td><jsp:include page="news/customnews.jsp"
      flush="true"/> </td>
  </tr>
  <tr>
    <td><jsp:include page="news/astronomy.jsp"
      flush="true"/> </td>
  </tr>
</table>
```

Front Controller

- ▶ Defined in the ORACLE developer resources
 - ▶ <http://java.sun.com/blueprints/corej2eepatterns/Patterns/FrontController.html>
- ▶ Provides best practices for addressing composite view management

ORACLE Sun Developer Network (SDN)

Sun Java Solaris Communities My SDN Account

APIs Downloads Products Support Training Participate search tips Search

SDN Home > Products & Technologies > Java Technology > J2EE > Reference > BluePrints > Welcome to Core J2EE Patterns > Core J2EE Pattern Catalog >

Core J2EE Pattern Catalog

Core J2EE Patterns - Front Controller

Print-Friendly Version

Context

The presentation-tier request handling mechanism must control and coordinate processing of each user across multiple requests. Such control mechanisms may be managed in either a centralized or decentralized manner.

Problem

The system requires a centralized access point for presentation-tier request handling to support the integration of system services, content retrieval, view management, and navigation. When the user accesses the view directly without going through a centralized mechanism, two problems may occur.

- Each view is required to provide its own system services, often resulting in duplicate code.
- View navigation is left to the views. This may result in commingled view content and view navigation.

Additionally, distributed control is more difficult to maintain, since changes will often need to be made in numerous places.

Forces

- Common system services processing completes per request. For example, the security service completes authentication and authorization checks.
- Logic that is best handled in one central location is instead replicated within numerous views.
- Decision points exist with respect to the retrieval and manipulation of data.
- Multiple views are used to respond to similar business requests.
- A centralized point of contact for handling a request may be useful, for example, to control and log a user's progress through the site.
- System services and view management logic are relatively sophisticated.

Solution

Use a controller as the initial point of contact for handling a request. The controller manages the handling of the request, including invoking security services such as authentication and authorization, delegating business processing, managing the choice of an appropriate view, handling errors, and managing the selection of content creation strategies.

The controller provides a centralized entry point that controls and manages Web request handling. By centralizing decision points and controls, the controller also helps reduce the amount of Java code, called *scripts*, embedded in the JavaServer Pages (JSP) page.

Centralizing control in the controller and reducing business logic in the view promotes code reuse across requests. It is a preferable approach to the alternative-embedding code in multiple views because that approach may lead to a more error-prone, reuse-by-copy- and-paste environment.

Typically, a controller coordinates with a dispatcher component. Dispatchers are responsible for view management and navigation. Thus, a dispatcher chooses the next view for the user and vectors control to the resource. Dispatchers may be encapsulated within the controller directly or can be extracted into a separate component.

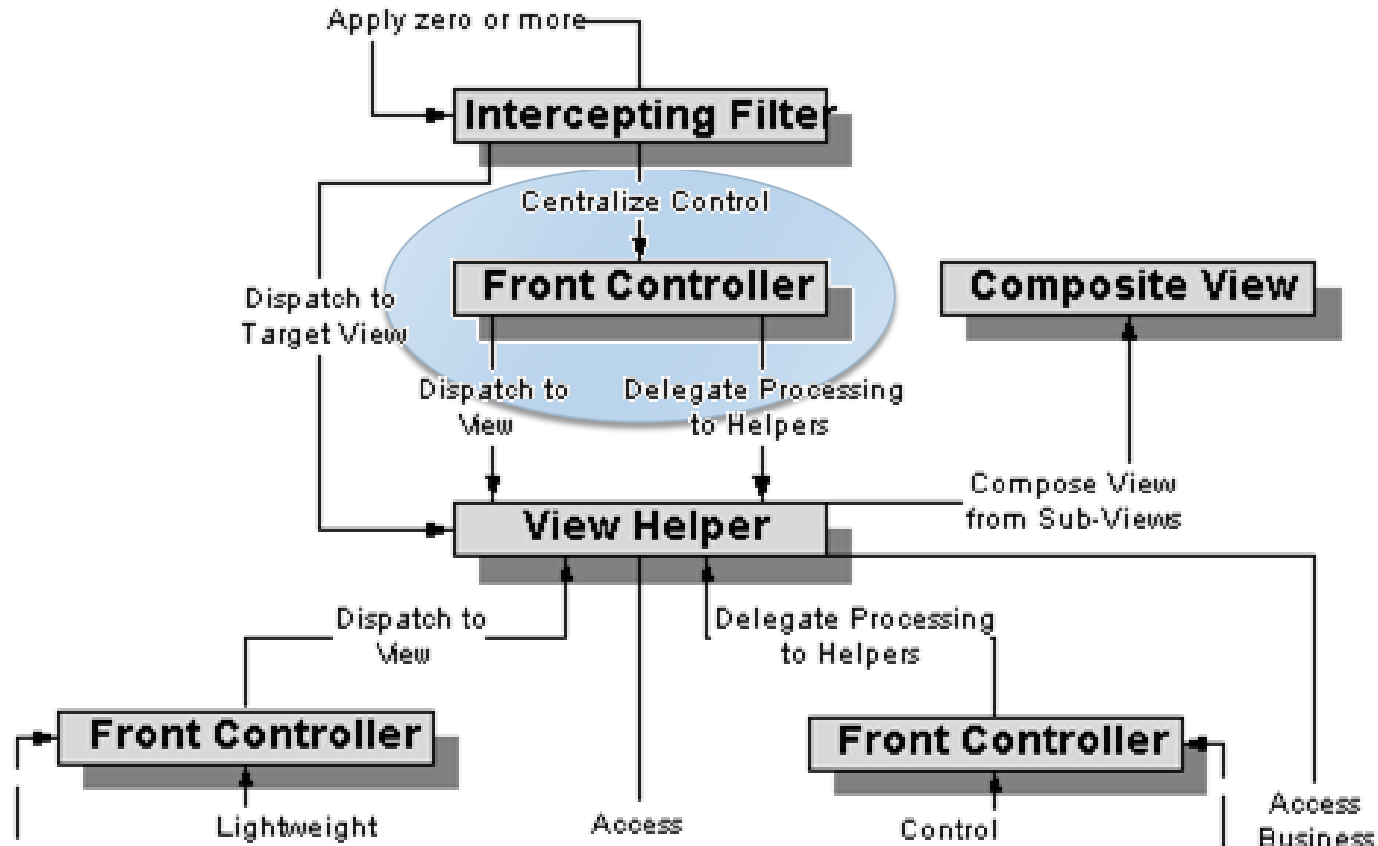
While the Front Controller pattern suggests centralizing the handling of all requests, it does not limit the number of handlers in the system, as does a Singleton. An application may use multiple controllers in a system, each mapping to a set of distinct services.

Structure

Figure 7.7 represents the Front Controller class diagram pattern.

```
classDiagram
    class Client
    class Controller
    Client --> Controller : send request
```

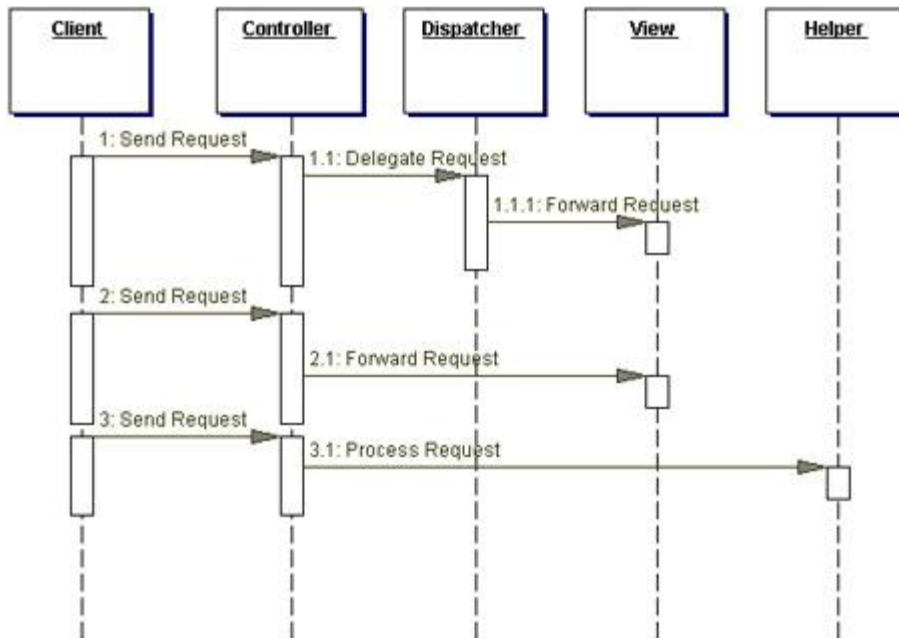
Front Controller



Front Controller Pattern

- ▶ Use a controller as the initial point of contact for handling a request.
- ▶ The controller manages the handling of the request, including
 - ▶ invoking security services such as authentication and authorization,
 - ▶ delegating business processing,
 - ▶ managing the choice of an appropriate view,
 - ▶ handling errors,
 - ▶ managing the selection of content creation strategies.

Front Controller Participants



▶ Controller

- ▶ initial contact point for handling all requests in the system

- ▶ may delegate specific tasks to a helper

▶ Dispatcher

- ▶ responsible for view management and navigation,

- ▶ next view to present to the user

- ▶ mechanism for vectoring control to this resource

- ▶ can be encapsulated within a controller (or can be a separate)

- ▶ static dispatching or dynamic dispatching

▶ Helper

- ▶ A helper is responsible for helping a view or controller complete its processing.

- ▶ typically implemented as JavaBeans components (JSP 1.0+) and custom tags (JSP 1.1+).

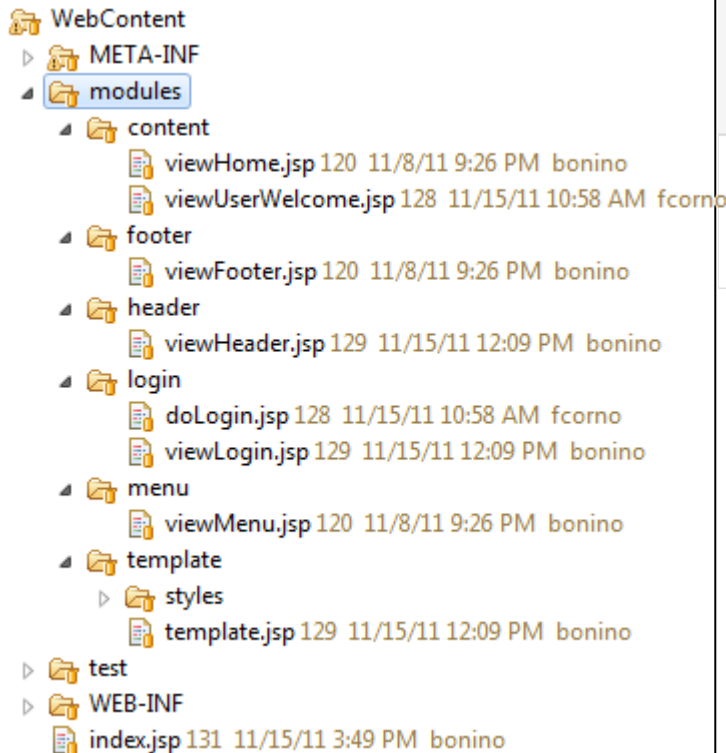
▶ View

- ▶ A view represents and displays information to the client.

The Energy Community Modular Infrastructure

- ▶ **Simplified combination of**
 - ▶ front-controller
 - ▶ composite view
- ▶ **Uses JSP strategy for both**
- ▶ **Sub-Views**
 - ▶ Organized in homogeneous modules
 - ▶ Managed by a front controller named `index.jsp`
 - ▶ Basically forwards requests to the `template.jsp` composite view

The Energy Community Modular Infrastructure



Licenza d'uso



- ▶ Queste diapositive sono distribuite con licenza Creative Commons “Attribuzione - Non commerciale - Condividi allo stesso modo 2.5 Italia (CC BY-NC-SA 2.5)”

- ▶ Sei libero:

- ▶ di riprodurre, distribuire, comunicare al pubblico, esporre in pubblico, rappresentare, eseguire e recitare quest'opera
- ▶ di modificare quest'opera



- ▶ Alle seguenti condizioni:

- ▶ **Attribuzione** — Devi attribuire la paternità dell'opera agli autori originali e in modo tale da non suggerire che essi avallino te o il modo in cui tu usi l'opera.
- ▶ **Non commerciale** — Non puoi usare quest'opera per fini commerciali.
- ▶ **Condividi allo stesso modo** — Se alteri o trasformi quest'opera, o se la usi per crearne un'altra, puoi distribuire l'opera risultante solo con una licenza identica o equivalente a questa.



- ▶ <http://creativecommons.org/licenses/by-nc-sa/2.5/it/>