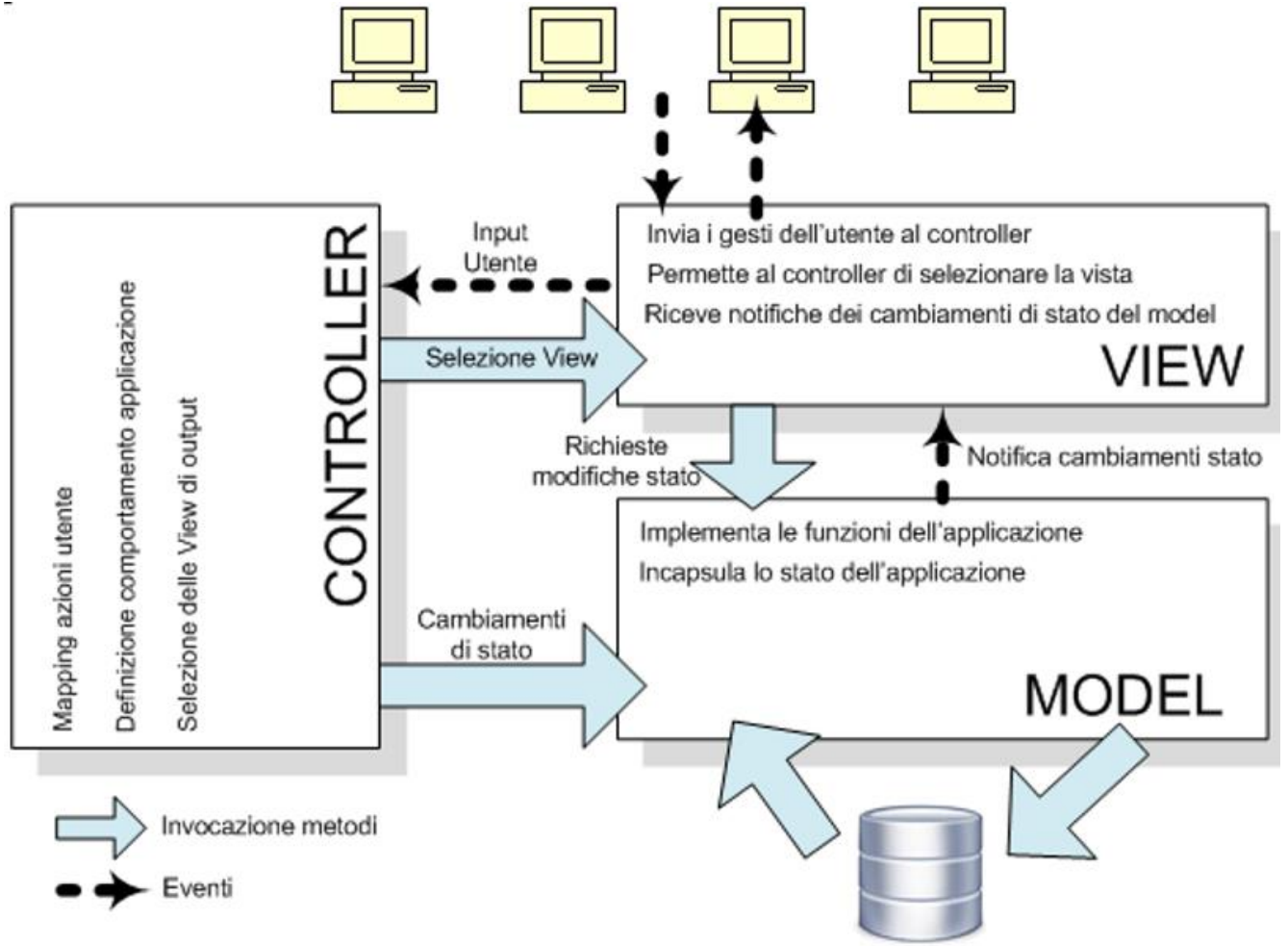
Il pattern MVC applicato al Web

Il pattern MVC applicato al Web

Il modello più utilizzato in ambito delle applicazioni orientate al web (ma non solo) è il pattern Model-View-Controller (MVC).

Questo pattern permette una maggiore strutturazione del codice, con un aumento della manutenibilità software e una suddivisione dell'applicazione in sottosistemi.

Per comprendere la filosofia del modello, nella slide successiva, è riportata la schematizzazione delle tre componenti in cui si divide.

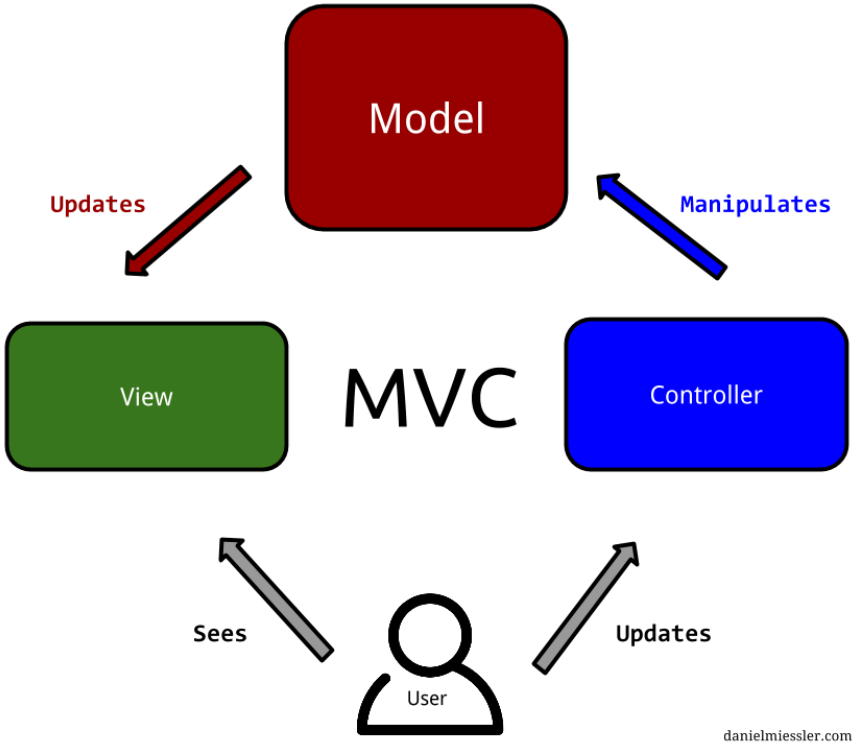


I componenti:

1) La view

2) Il model

3) Il controller



La vista

La vista (view) è il componente dove viene gestita la presentazione dei dati.

E’ rappresentata da tutto quel codice di presentazione che permette all’utente (ad esempio tramite un browser) di operare le richieste.

All’interno di questo livello lavorano sia programmatori che grafici che curano la parte estetica della presentazione

Esempio di vista: una jsp

```
<%@ page language="java" contentType="text/html; charset=US-
ASCII"
    pageEncoding="US-ASCII"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=US-ASCII">
<title>First JSP</title>
</head>
<%@ page import="java.util.Date" %>
<body>
<h3>Hi Pankaj</h3><br>
<strong>Current Time is</strong>: <%=new Date() %>

</body>
</html>
```

Il model

Il modello (model) è il componente che rappresenta e gestisce i dati, tipicamente persistenti su database.

Esempio di model, una classe di dominio:

-Definisce variabili di istanza e metodi setter e getter per l'oggetto

```
package javaskool;
import java.io.*;
...

public class Customer implements Serializable
{
        private String custID;
        private String custName;
        private int qty;

        public String getCustID() {
            return custID;
        }

        public void setCustID(String custID) {
            this.custID = custID;
        }

        public String getCustName() {
            return custName;
        }

        public void setCustName(String custName) {
            this.custName = custName;
        }

        public int getQty() {
            return qty;
        }

        public void setQty(int qty) {
            this.qty = qty;
        }
}
```

Il controller

Non incapsula logica applicativa!

Il controllore (controller) è il componente che veicola i flussi di interazione tra vista e modello, organizzando il comportamento dell'applicazione e tracciando la navigazione con meccanismi che creano uno stato associato all'utente.

Nello specifico, intercetta le richieste HTTP del client e traduce ogni singola richiesta in una specifica operazione per il model; in seguito può eseguire lui stesso l'operazione oppure delegare il compito ad un'altra componente.

In ultima, seleziona la corretta vista da mostrare al client ed inserisce, eventualmente, i risultati ottenuti.

In java, si definiscono classi filtro che svolgono questo compito.

Esempio di filtro:

```
package controller.filter;

import java.io.IOException;
import javax.servlet.Filter;
...

public class UtenteFilter implements Filter {

    public UtenteFilter() {
        ...
    }

    public void destroy() {
        ...
    }

    public void doFilter(ServletRequest request, ServletResponse response, FilterChain chain)
        throws IOException, ServletException {
        ...
    }

    public void init(FilterConfig fConfig) throws ServletException {
        ...
    }

}
```