UNIT : 7

Reference :-Marty Hall, Larry Brown, “Core Servlets and JavaServer Pages Volume – 1”, Pearson Education, 2nd ed.(2004)

Chapter :- 15 : Integrating Servlets and JSP: The Model View Controller (MVC) Architecture

Why integration of servlet & JSP?

- Servlets are good at data processing: reading and checking data, communicating with databases, invoking business logic, and so on.
- JSP pages are good at presentation: building HTML to represent the results of requests. This chapter describes how to combine servlets and JSP pages to best make use of the strengths of each technology.

Need of MVC

- Servlet Capabilities
  - manipulate HTTP status codes and headers,
  - use cookies,
  - track sessions,
  - save information between requests,
  - compress pages,
  - access databases,
  - generate JPEG images on-the-fly,
  - and perform many other tasks flexibly and efficiently
- Generating HTML with servlets can be tedious and can yield a result that is hard to modify.

Need of MVC

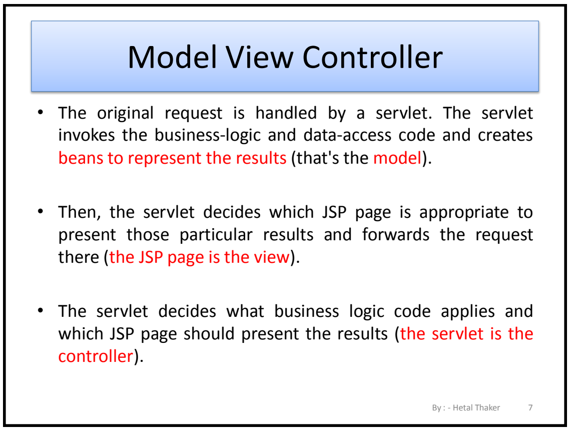
- JSP lets you separate much of the presentation from the dynamic content.
- JSP expressions, scriptlets, and declarations let you insert simple Java code into the servlet that results from the JSP page, and directives let you control the overall layout of the page.

Need of MVC

- What if you want to give totally different results depending on the data that you receive?
- Scripting expressions, beans, and custom tags, although extremely powerful and flexible, don't overcome the limitation that the JSP page defines a relatively fixed, top-level page appearance.
- Similarly, what if you need complex reasoning just to determine the type of data that applies to the current situation? JSP is poor at this type of business logic.

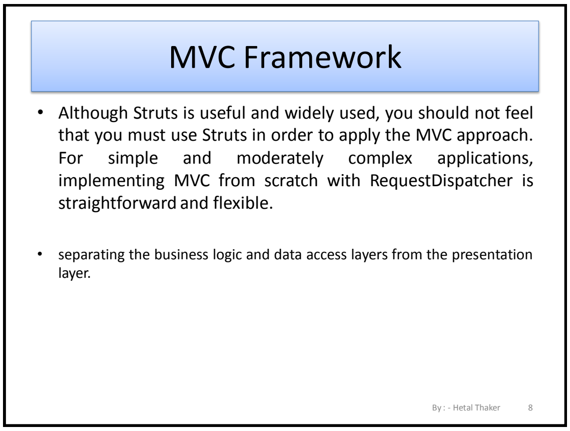
Need of MVC

- The solution is to use both servlets and JavaServer Pages. In this approach, known as the Model View Controller (MVC) or Model 2 architecture, you let each technology concentrate on what it excels at.



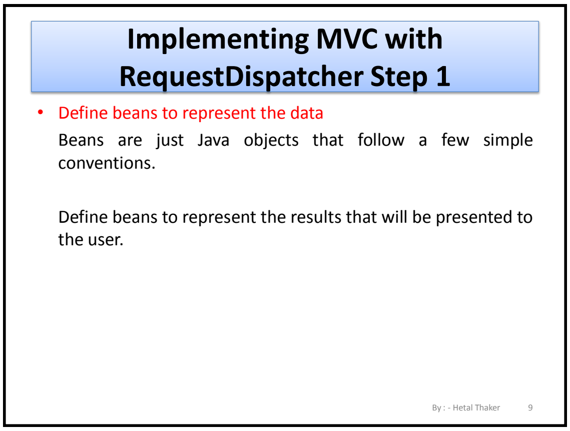
Model View Controller

- The original request is handled by a servlet. The servlet invokes the business-logic and data-access code and creates beans to represent the results (that's the model).
- Then, the servlet decides which JSP page is appropriate to present those particular results and forwards the request there (the JSP page is the view).
- The servlet decides what business logic code applies and which JSP page should present the results (the servlet is the controller).



MVC Framework

- Although Struts is useful and widely used, you should not feel that you must use Struts in order to apply the MVC approach. For simple and moderately complex applications, implementing MVC from scratch with RequestDispatcher is straightforward and flexible.
- separating the business logic and data access layers from the presentation layer.

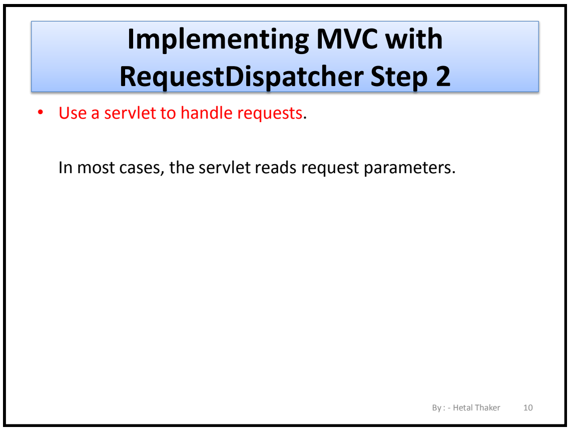


Implementing MVC with RequestDispatcher Step 1

- Define beans to represent the data

Beans are just Java objects that follow a few simple conventions.

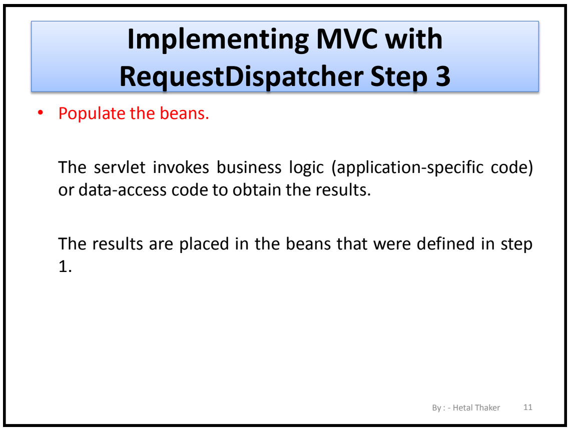
Define beans to represent the results that will be presented to the user.



Implementing MVC with RequestDispatcher Step 2

- Use a servlet to handle requests.

In most cases, the servlet reads request parameters.

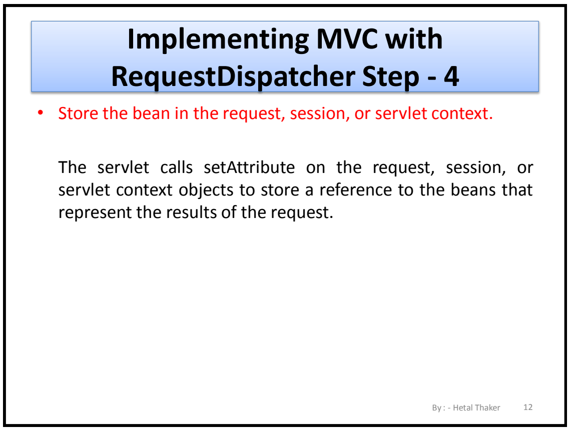


Implementing MVC with RequestDispatcher Step 3

- Populate the beans.

The servlet invokes business logic (application-specific code) or data-access code to obtain the results.

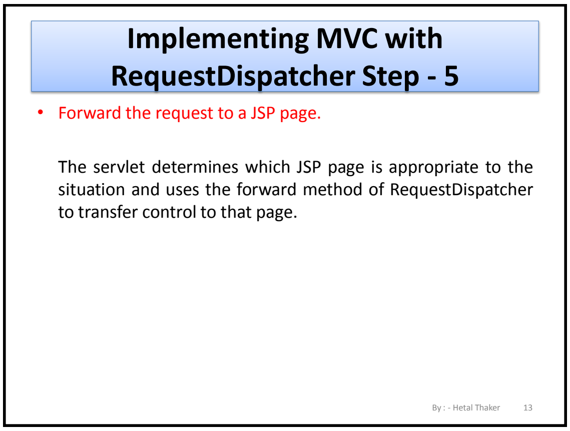
The results are placed in the beans that were defined in step 1.



Implementing MVC with RequestDispatcher Step - 4

- Store the bean in the request, session, or servlet context.

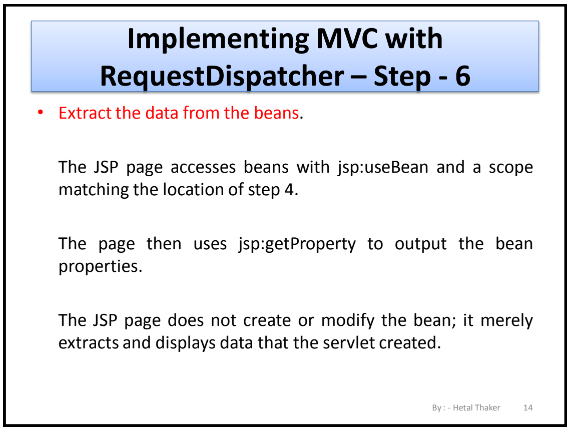
The servlet calls setAttribute on the request, session, or servlet context objects to store a reference to the beans that represent the results of the request.



Implementing MVC with RequestDispatcher Step - 5

- Forward the request to a JSP page.

The servlet determines which JSP page is appropriate to the situation and uses the forward method of RequestDispatcher to transfer control to that page.



Implementing MVC with RequestDispatcher – Step - 6

- Extract the data from the beans.

The JSP page accesses beans with jsp:useBean and a scope matching the location of step 4.

The page then uses jsp:getProperty to output the bean properties.

The JSP page does not create or modify the bean; it merely extracts and displays data that the servlet created.