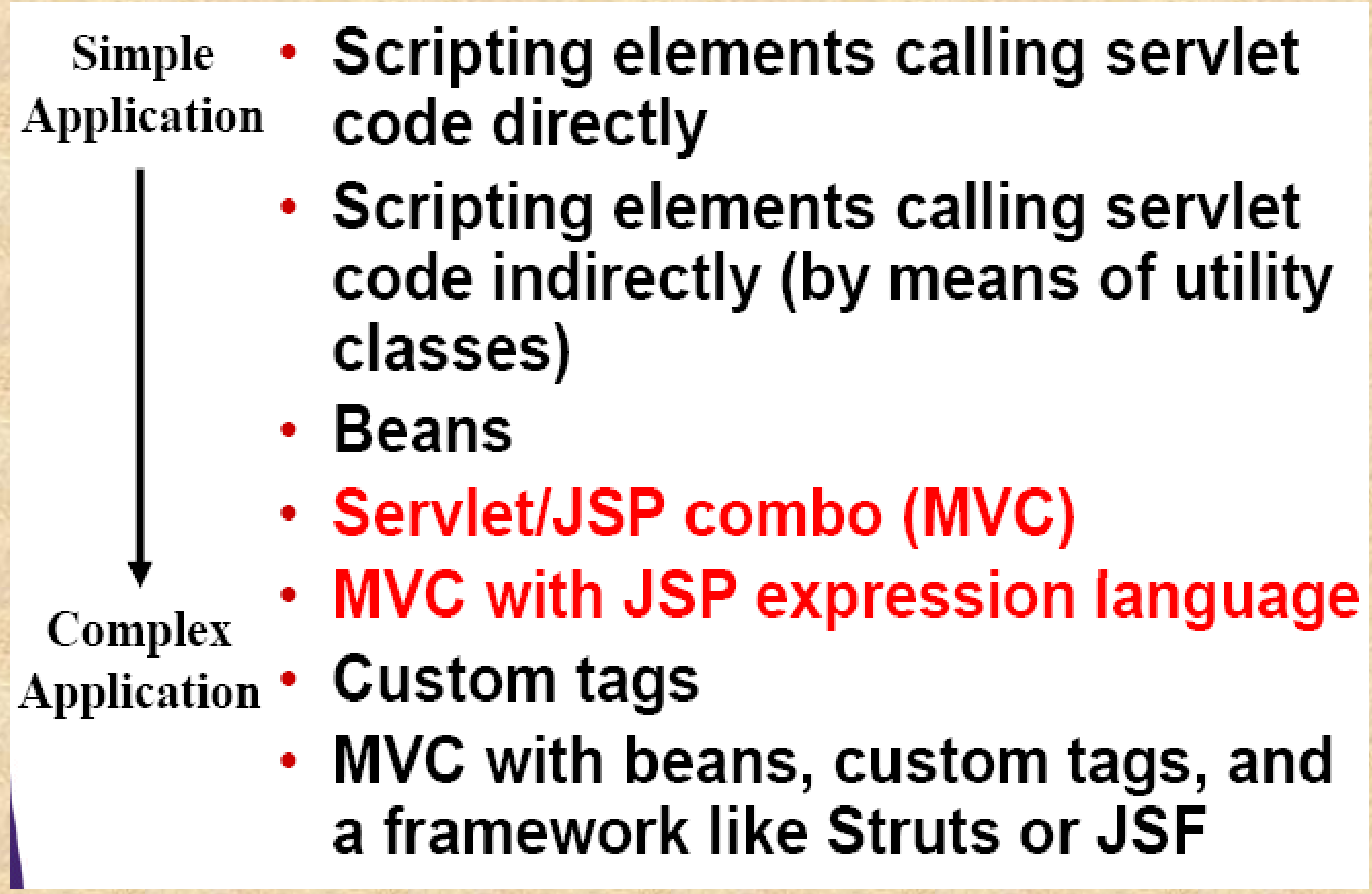
Understanding the need for MVC

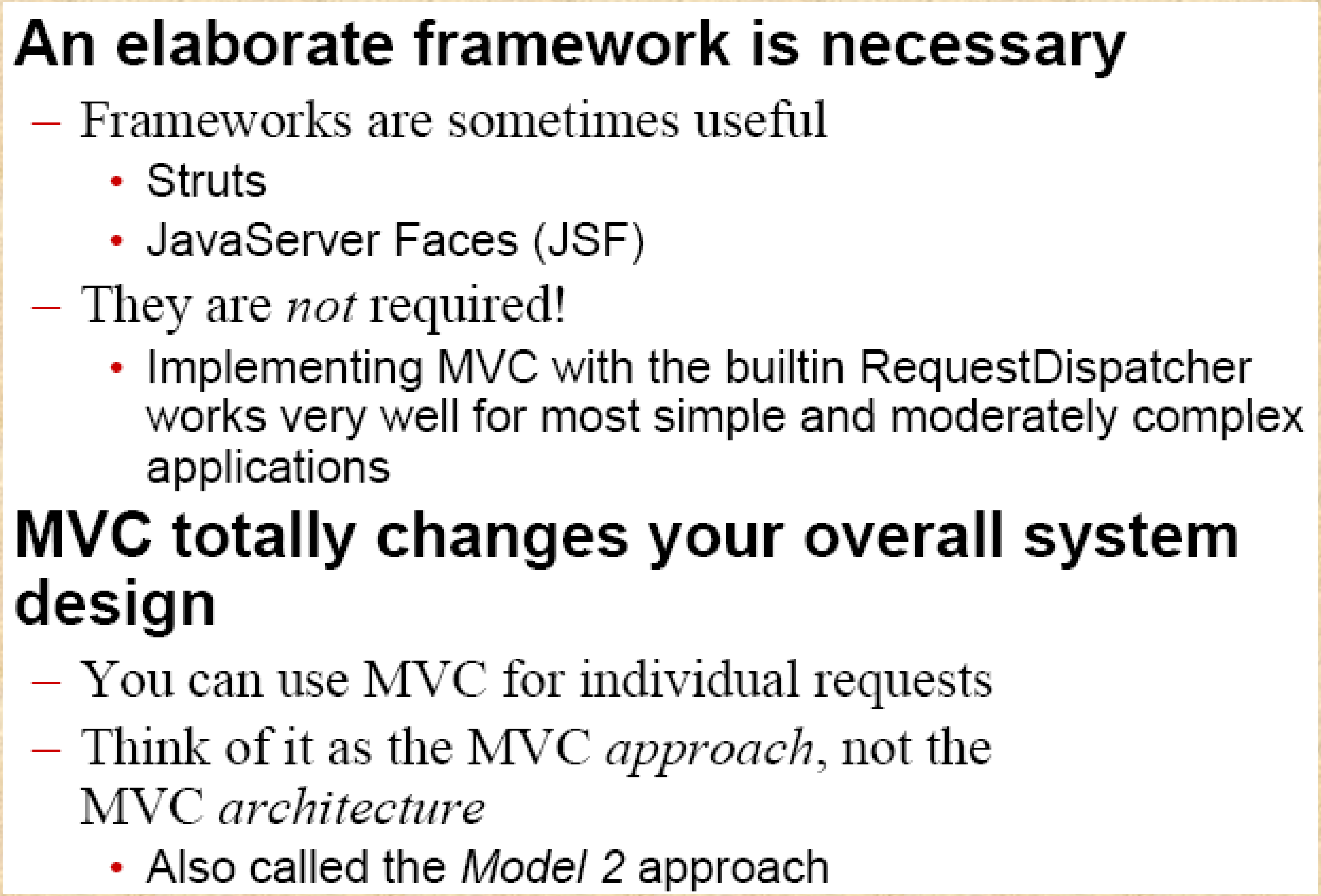


Simple Application

Complex Application

- Scripting elements calling servlet code directly
- Scripting elements calling servlet code indirectly (by means of utility classes)
- Beans
- Servlet/JSP combo (MVC)
- MVC with JSP expression language
- Custom tags
- MVC with beans, custom tags, and a framework like Struts or JSF

MVC Frameworks



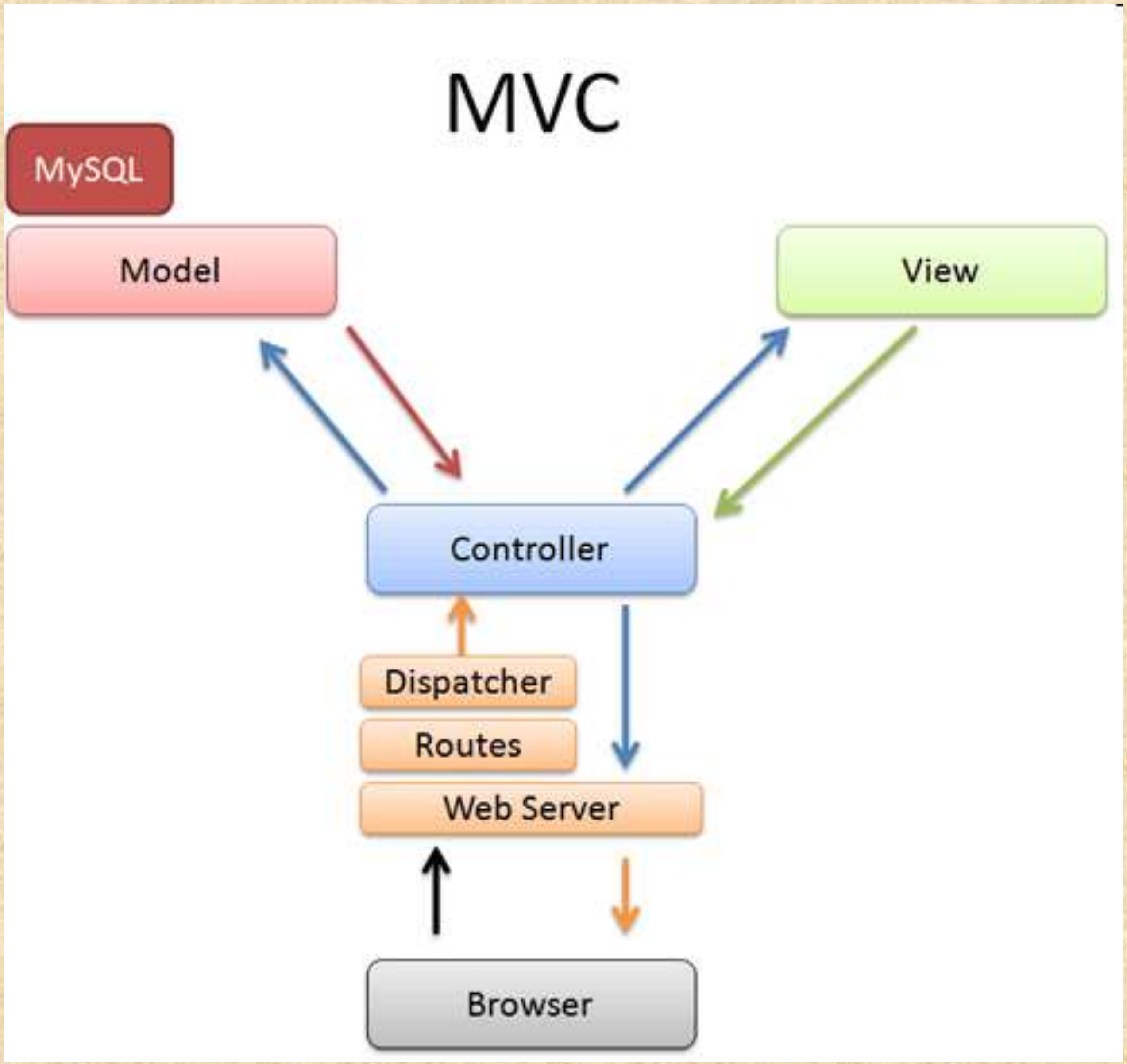
An elaborate framework is necessary

- Frameworks are sometimes useful
  - Struts
  - JavaServer Faces (JSF)
- They are *not* required!
  - Implementing MVC with the builtin RequestDispatcher works very well for most simple and moderately complex applications

MVC totally changes your overall system design

- You can use MVC for individual requests
- Think of it as the MVC *approach*, not the MVC *architecture*
  - Also called the *Model 2* approach

MVC Architecture



MVC Architecture

The term “Architecture” suggests “overall system design”. It is quite common for applications to handle some requests with servlets.

The other requests with JSP and servlets acting in conjunction. Due to this, we have to use the MVC approach .

Implementing MVC with RequestDispatcher

The most important point about MVC is the idea of separating the business logic and data access layers from the presentation layer.

1) Define bean to represent data
2) Use a servlet to handle requests
3) Populate the Beans
4) Store the bean in the request, session, or servlet
5) Forward request to JSP
6) Extract data from the Bean

Summarizing MVC Code

This section Summarizes the code that would be used for request-based, session-based and application-based MVC approaches

1) Request-Based Data Sharing

In request-based sharing the servlets stores the beans in the HttpServlet-Request, where they are accessible only to the destination JSP pages.

Servlet:

Request.setAttribute(“Key”, Value);

RequestDispatcher d=new request.getRequestDispatcher(“page.jsp”);

Dispatcher.forward(request, response)

JSP Page:

<jsp:useBean id=“key” type=“class” scope=“request” />

<jsp:getProperty name=“key” Property=“xyz” />

Summarizing MVC Code

2) Session Based Data Sharing

In this the servlet stores the beans in the HttpSession, where they are accessible to the same client in the destination JSP Page.

Servlet:

Session.setAttribute(“Key”, Value);

RequestDispatcher d=new request.getRequestDispatcher(“page.jsp”);

Dispatcher.forward(request, response)

JSP Page:

<jsp:useBean id=“key” type=“class” scope=“Session” />

<jsp:getProperty name=“key” Property=“xyz” />

Summarizing MVC Code

3) Application Based Data Sharing

In this the servlet stores the beans in the Servlet-Context, where they are accessible to any client in the destination JSP Page.

Servlet:

```
getServletContext().setAttribute(“Key”, Value);
RequestDispatcher d=new request.getRequestDispatcher(“page.jsp”);
Dispatcher.forward(request, response)
```

JSP Page:

```
<jsp:useBean id=“key” type=“class” scope=“Application” />
<jsp:getProperty name=“key” Property=“xyz” />
```