Introduction to Modern Application Development

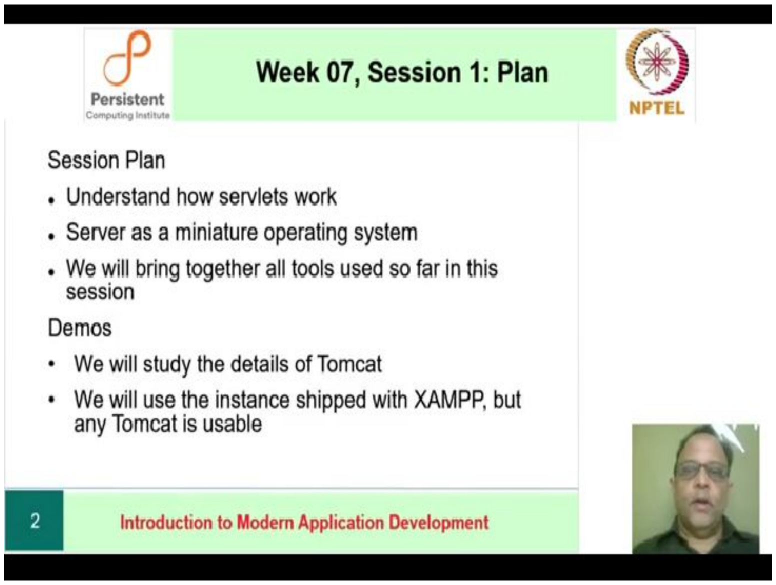
Prof. Aamod Sane
FLAME University and Persistent Computing Institute
Abhijat Vichare
Persistent Computing Institute
Madhavan Mukund
Chennai Mathematical Institute

Lecture-22
Week7 Session1-Part1

Hello, welcome to modern application development week 7 session. In this part of this course, we have reached a point where we are done with basic things such as CGI applications. And now we are going to do servlets, which you can think of as an efficient form of CGI if you want, but it turns out that they end up giving much more than that, and is the preferred way of developing web applications.

Once you are past using the CGI bin to understand exactly how the data and control are flowing in the application.

(Refer Slide Time: 00:59)



Week 07, Session 1: Plan

Session Plan
- Understand how servlets work
- Server as a miniature operating system
- We will bring together all tools used so far in this session

Demos
- We will study the details of Tomcat
- We will use the instance shipped with XAMPP, but any Tomcat is usable

2 Introduction to Modern Application Development

So, our plan is as follows:

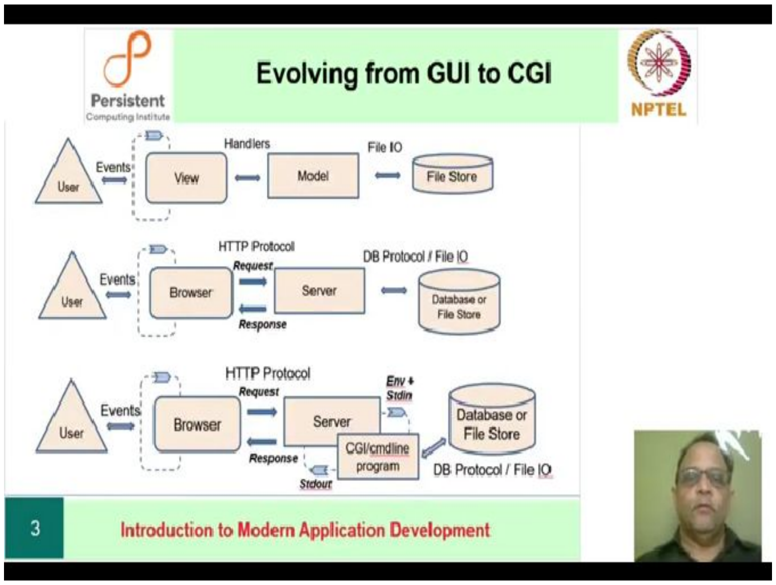
1. We will understand how servlets work.

2. We will understand the server as a kind of miniature operating system.
3. We will start bringing together all the tools used so far in this session.

As part of the demo this time:

1. We will study the details of Tomcat and how it works as a servlet server or servlet container and such other many different names get used.
2. We will use the Tomcat instance that is shipped with xampp. (But you can use any Tomcat instance that you want. You can learn about it on the installation part on the web at the Tomcat site). And in this session once your Tomcat is installed, we will look at how to get it configured and how it works.

(Refer Slide Time: 02:02)



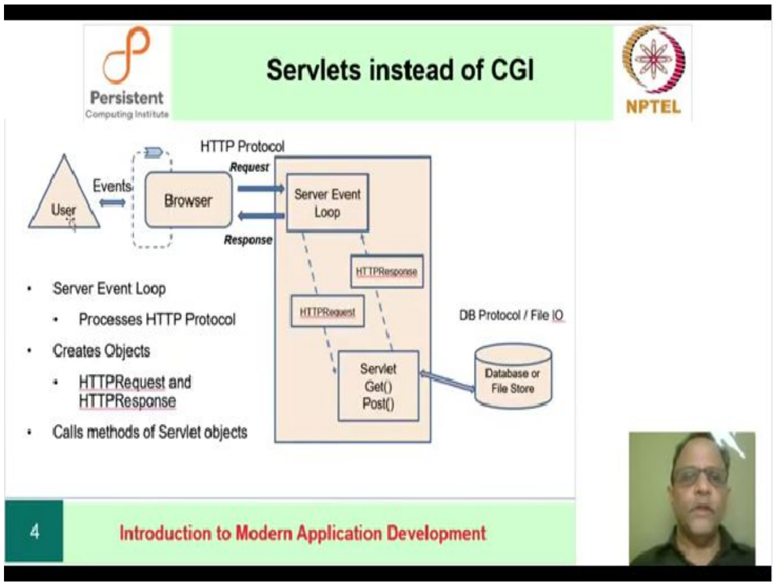
So far, we have seen the following stages of our application:

1. We began by looking at a basic application i.e. CGI bin style.
2. We then saw that you can have a GUI with a user, view, model and a file store.
3. Then we saw how the place of the view is taken by the browser, the event loop between the user and the browser is connected with the server through the HTTP protocol and the server works with a database or a file store to store the information that you have sent.

4. Next, we saw that the user connects to the browser. The server can be made to do different things using CGI/cmdline programs. We saw that the CGI/cmdline program is a distinct process and that it communicates using STDIN and STDOUT. CGI/cmdline programs can use the database or file store.

Servlets is a further evolution of the server - CGI relationship. This, the result looks like the following:

(Refer Slide Time: 03:18)



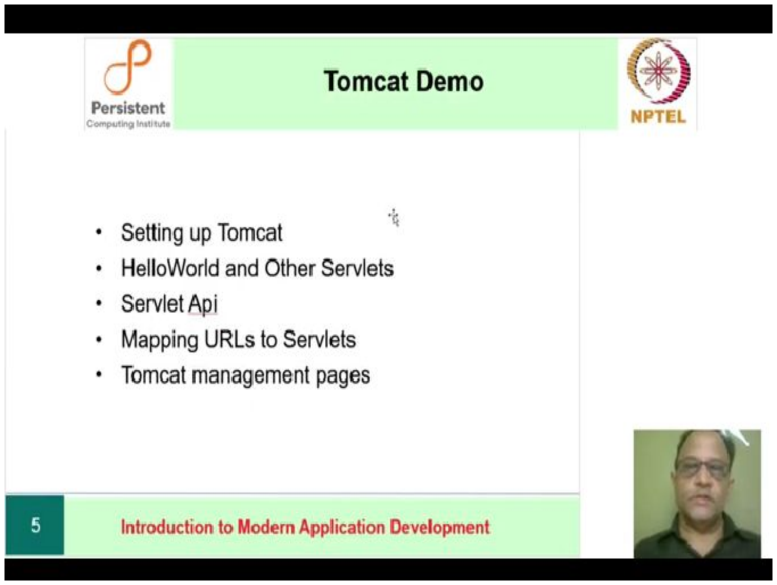
Servlets instead of CGI

- Server Event Loop
  - Processes HTTP Protocol
- Creates Objects
  - HTTPRequest and HTTPResponse
- Calls methods of Servlet objects

As usual we have the user events and the browser and the HTTP Protocol. However, this time, there is an event loop inside of the server, which listens to the HTTP events. And when it receives some kind of a request, it creates an object called *HttpRequest*. This is a Java object and a Servlet is another object which offers interfaces for the functions *Get()* and *Post()*. There is a certain style of writing these interfaces that we will see when we look at the Java code.

When the servlet is done executing the request, it creates a *HttpResponse* object and sends it to the server event loop. Now the *HTTRequest* and *HttpResponse* objects contain exactly the same information that we have seen in HTTP requests and HTTP responses so far. So as we go further, we will see the details of how the *HttpRequest* and *HttpResponse* objects contain exactly the HTTP headers that we have seen in the previous sessions.

(Refer Slide Time: 04:30)



Tomcat Demo

- Setting up Tomcat
- HelloWorld and Other Servlets
- Servlet Api
- Mapping URLs to Servlets
- Tomcat management pages

5 Introduction to Modern Application Development

Now lets go ahead to our demo. In this demo, we will see how Tomcat is set up once the basics are installed. As usual, we will proceed by looking at simple servlets like *HelloWorld* and a few others. These are shipped with Tomcat and hence makes it very easy to understand the basics of the working of the system.

As we go ahead with the demo, we will study:

1. The servlet API.
2. Discuss things such as mapping URLs to servlets.
3. What the Tomcat management pages contain.**(Video Starts: 05:15)**

We do the following:

1. Go to the **xampp directory**.
2. Start the **xampp control**.
3. Start **Tomcat** from the console.

The console displays Tomcat's process ID (896, here) and the ports it is running in, of which 8080 is the one that we are interested in.

Tomcat has documentation, configuration, examples, etc. built in into the install.

1. Open Browser, enter the URL: https://localhost:8080 (i.e. the port where Tomcat is running as shown in console).
2. Select Documentation.
3. Go to the hyperlink First web application (i.e. http://localhost:8080/docs/appdev/index.html). It has links named: *Introduction*, *installation*, *Deployment Organization*, etc.

Similar to CGI files that are being stored in CGI bin directory and the HTML file that actually shows us the application is kept in htdocs under Apache, there is a setup in Tomcat as well.

In the xampp/Tomcat directory, there are *conf*, *lib* and *logs* directories. The existence of *conf* and *log* is characteristic of most web applications.

Tomcat manager app has authentication built in, but usually the default password does not work. To add a password, go to conf/Tomcat_users.xml. Add the following lines of code before `<tomcat_users/>`

```
<user username="manager" password="manager" roles="manager-gui"/>
<user username="admin" password="admin" roles="admin-gui"/>
```

This adds authentication to roles like manager GUI, which allows us to go to the manager part of the application. As time has passed, people notice that for various security reasons, you need fairly fine grained access to the various tools that are available to the web server.

NOTE: In real life, password needs to be more unpredictable for making the access secure.

Upon restarting Tomcat, enter your new credentials in the manager app to login as manager. Just like different fairshare applications stored in apache, there are different manager applications. such a docs app, examples app, and the manager app itself. There are a number of different commands available for working with these applications like START, STOP, UNDEPLOY, etc.

To deploy more applications, upload them as web archive files(.WAR files). So already, Tomcat has given us a more methodical way of organizing applications unlike the CGI bin applications where we have to come up with an organization ourselves. Servers like Tomcat were developed much later based on the experience with apache such as adding configuration parts.

In the examples app there are servlet examples, JSP examples, WebSocket, and so on. Tomcat comes with basic examples in the installation package. Servlets examples folder has a few servlets apps. The source that we see is not actually all the source code.
Go to Developer tools of the browser and reload the example to observe GET request and response headers. The response has the server name.

helloWorld example is of the equivalent of the cgi-bin that comes with the CGI files. This source application is actually served by a get request and the get request manufactures this file. It is mentioning essential parts of the actual source code.

1. `public class HelloWorld extends HttpServlet`
2. `HelloWorld` class has:

   `public void doGet(HttpServletRequest request, HttpServletResponse response);`

   which corresponds to the *Get()* method call that we actually see in **Slide Time: 03:18.**
3. The *request* object and the *response* object in **Slide Time: 03:18** is the `HttpServletRequest` object and the `HttpServletResponse` object. This is the somewhat standard verbosity of Java.
4. `response.setContentType("text/html")`, in its essence (not in the actual code) is to set the Content-Type to `text/html` in the response headers .
5. `PrintWriter()` is used to write to the response object using `out.println`. This writes to the HTML file that needs to be rendered eventually to the user, similar to `system.out.println` in CGI.
6. The actual code (HelloWorld.java) has quite a few more details like **serial version ID, locales**, etc which are standard things to do to make sure that the execution is according

to the expectations of users. Briefly, locales help to adjust things in such a way that they are friendly to people who are not necessarily from your country. It has some of the more nuanced things like background color, extra information written in HTML file such as images, hyperlinks, colors and other formatting options like width, height, etc.

Overall, the code that we have seen is laid out inside xampp/Tomcat/webapps/examples where there are servlets, JSP, WebSocket, etc. we would not be looking at that much. Servlets folder has *helloWorld.html* similar to the htdocs folder in the world of CGI bin. This is the summary of the code that they showed us. And the application itself (HelloWorld.java) is in WEB-INF/classes.

WEB-INF has a bunch of standard things like *web.xml*, *classes*, *JSP* and *libs,* which has various libraries.

Now let us take a look at some of the other things that are included in Tomcat. So we have more servlet examples. In particular, these show you how, for example, the information in a request in this case it analyzes the URL i.e. examples/servlet/servlets/RequestInfoExample.html, and outputs the URL, GET method, protocol used (in this case, HTTP/1.1), etc similar to the various shell scripts and environments, except that it is done through Java objects.

Third example is requestHeaders example which displays the request headers. This is done by calling `getHeadernames()` of `HttpServletRequest` object in `doGet()` method. `response.getWriter()` returns a `PrintWriter` object that can send character text to the client and for each one of header names, ask the request object to get its value.

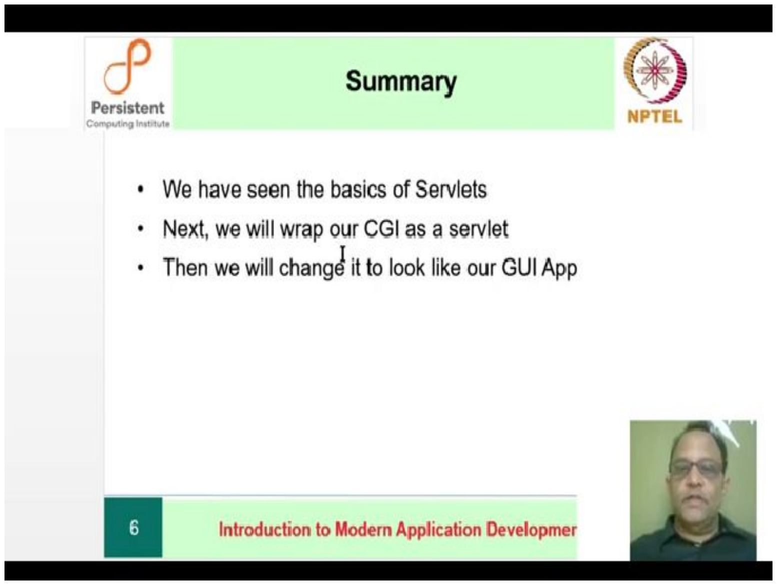
postdata.sh file collects all the data in the post. This has a number of things in the form `:HTTPH[]`. We print it out through the environment array. The request object corresponds to the environment array. And it contains all of these different elements. It is very easy to learn what the system is doing from this file. And as we develop our app further, we will use these examples as our starting points and show how these kinds of things get implemented.

To summarize:

1. Basics of Servlets
2. We have seen Tomcat setup
3. We have learnt about `HttpServletRequest` objects, `HttpServletResponse` objects, `doGet()` and `doPost()`.
4. We have seen a few examples of servlets.
5. We saw a hint of what a servlet API is, which is exactly what those `doGet()`, `doPost()`, `request` and `response` objects are.

We will take a little bit of a look at how URLs and servlets are mapped. And what the rest of the management pages contain. Then we will wrap our existing CGI file as a servlet. And finally change it to look like our GUI app. **(Video Ends: 27:36)**

(Refer Slide Time: 27:37)



Summary

- We have seen the basics of Servlets
- Next, we will wrap our CGI as a servlet
- Then we will change it to look like our GUI App

6 Introduction to Modern Application Developmer