**What is Servlet**

Web application တစ္ခုကို ဖန္တီးရာမွာ အသံုးၿပဳတဲ့ technology တစ္ခုၿဖစ္တယ္။ အဲ့မွာ web applicaton တစ္ခုဆိုတာ ဘာလဲဆိုေတာ့ sever side ရယ္ dynamic web page တစ္ခုရွိတယ္ဆိုရင္ အဲ့တာကို web application လို႔ေခၚတယ္။

Servlet ဆိုတာကလည္း java ၿဖစ္တဲ့အတြက္ robust နဲ႔ scalable ၿဖစ္တယ္။ servlet မတိုင္ခင္တုန္းက CGI (Common Gateway Interface) scripting language ကို server side programming language တစ္ခုအေနနဲ႔ သံုးခဲ့ရတယ္။

Servlect API မွာ မ်ားၿပားလွတဲ့ interfaces နဲ႔ classes’ ေတြရွိတယ္ ဥပမာ Servlet, GenericServlet, HttpServlet, ServletRequest, ServletResponse, etc. တို႔ၿဖစ္ၾကတယ္

**Servlet ဆိုတာဘာလဲ**

Servlet ဆိုတာကို မ်ိဳးစံုအဓိပၸါယ္ဖြင့္လို႔ရတယ္။

* Web application တစ္ခုကို ဖန္တီးလို႔ရတဲ့ technology တစ္ခု။
* Documentation ပါ၀င္တဲ့ interfaces နဲ႔ classes မ်ားကို ေပးႏိုင္တဲ့ API တစ္ခု။
* မည္သည့္ servlet ကိုမဆိုဖန္တီးမယ္ဆိုရင္ servlet interface ကို implements လုပ္ကိုလုပ္ရမယ္ အဲ့တာေၾကာင့္ သူက interface တစ္ခုဆိုလည္း ဟုတ္တယ္။
* Servlet ဆိုတာ class တစ္ခုလည္း ဟုတ္တယ္ သူက မည္သည့္ request ကိုမဆို response ၿပန္ႏိုင္တယ္။
* Dynamic web page တစ္ခုကို ဖန္တီးဖို႔ရန္အတြက္ server ေပၚမွာ deployed လုပ္ၿခင္းခံရတဲ့ web component တစ္ခုလည္း ၿဖစ္တယ္။



**Web Application ဆိုတာဘာလဲ**

အလြယ္ေၿပာရမယ္ဆိုရင္ေတာ့ Web ေပၚမွာ run ႏိုင္တဲ့ application ကို web application လို႔ေခၚတယ္။ သူ႕ကို web components မ်ားၿဖစ္ၾကတဲ့ Servlet, JSP, Filter အစရွိတဲ့ web components မ်ားရယ္ HTML, CSS, and JavaScript အစရွိတဲ့ elements မ်ားနဲ႔ ေပါင္းစပ္ဖြဲ႕စည္းထားၿခင္း ၿဖစ္ပါတယ္။ အဲ့ web components ေတြက web server ေပၚမွာ execute လုပ္တယ္ ၿပီးရင္ HTTP request ကို response ၿပန္တယ္။

**CGI (Common Gateway Interface)**

CGI ဆိုတာကလည္း server နဲ႔အလုပ္တြဲလုပ္တဲ့ technology တစ္ခုၿဖစ္တယ္။ program ဆီကို http request မ်ားပို႔လႊတ္မယ္ process ေတြလုပ္မယ္ အစရွိသၿဖင့္ server side လုပ္ငန္းမ်ားကို လုပ္ေဆာင္ႏိုင္တယ္။



**CGI ရဲ႕ မေကာင္းတဲ့အခ်က္မ်ား**

* Client မ်ားလာရင္ response time ပိုၾကာလာေရာ။
* သူက request တစ္ခုခ်င္းစီတိုင္းအတြက္ process တစ္ခုစတယ္ ဒါေပမယ့္ web server မွာက စတဲ့ process limit က အကန္႔အသတ္ရွိတယ္။
* သူ႕ကို platform dependent language ေတြၿဖစ္တဲ့ C, C++, perl တို႔က သံုးၾကတယ္။ platform independent ၿဖစ္တဲ့ java အတြက္ သိပ္မကိုက္လွေပဘူး။

**Servlet ရဲ႕ ေကာင္းတဲ့အခ်က္မ်ား**

Servlet မွာက ေကာင္းတဲ့အခ်က္အမ်ားၾကီးရွိတယ္။ servlet container ကို လာတဲ့ multiple request မ်ားကို handle လုပ္ဖို႔အတြက္ web container က threads မ်ားကို ဖန္တီးလိုက္တယ္။အဲ့မွာ ေကာင္းက်ိဳးေတြ မ်ားစြာၿဖစ္ေပၚလာတယ္ ဘာေတြလဲဆိုေတာ့ thread ၿဖစ္တဲ့အတြက္ေၾကာင့္ common memory area တစ္ခုကို share ၿပီးသံုးႏိုင္တယ္ lightweight ၿဖစ္တယ္ thread ေတြၾကားထဲက communication cost က ေလ်ာ့က်သြားတယ္။



* **Performance ပိုေကာင္းလာတယ္**: ဘာလို႔လဲဆိုေတာ့ process မဟုတ္ပဲ thread ေတြကို သံုးထားလို႔ၿဖစ္တယ္။ request တစ္ခုခ်င္းစီတိုင္းအတြက္ thread တစ္ခုစီကို servlet က ဖန္တီးေပးလိုက္လို႔ၿဖစ္တယ္။
* **Portability**: java language ကို သံုးထားလို႔။
* **Robust**: JVM က servlet ကို manage လုပ္တဲ့အတြက္ ကၽြန္ေတာ္တို႔က memory မေလာက္တာတို႔ garbage collection တို႔အတြက္ စိတ္ပူစရာမလိုေတာ့ဘူး။
* **Secure**: Java language ကို သံုးထားလို႔။

**Web Terminology မ်ား**

1. **Website**
2. **HTTP**
3. **HTTP Requests**
4. **Get vs Post**
5. **Container**
6. **Server: Web vs Application**
7. **Content Type**

**1. Website**

Website မွာက ၂ မ်ိဳးရွိတယ္။

1. **Static website:** static website လုပ္ရတာက လြယ္ပါတယ္။ programming knowledge တို႔ database design တို႔ဘာမွ သိစရာမလိုဘူး။ static website ဆိုတာက page ေတြက အေသေတြ သူ႕ထဲက information က ဘာမွမေၿပာင္းလဲသြားဘူး။

2. **Dynamic website:** dynamic ၿဖစ္ေနတဲ့ web pages မ်ားကို စုစည္းထားၿခင္းၿဖစ္တယ္။ သူ႕ရဲ႕ content ေတြက dynamically ေၿပာင္းလဲေနၾကတယ္။ သူက content ေတြကို access လုပ္ေပးေနတယ္။အဲ့ content ေတြက ဘာေတြ

ထြက္လာလဲဆိုေတာ့ Database ဆီကို CMS ကတစ္ဆင့္ ထြက္လာၾကတာ။ အဲ့တာေၾကာင့္ တစ္ခုခုကို update လုပ္လိုက္မယ္ဆိုရင္ အဲ့တာေတြက လိုက္ေၿပာင္းသြားလိမ့္မယ္ ဆိုလိုတာက အေသမဟုတ္တဲ့အတြက္ေၾကာင့္။

Dynamic website ေတြက client side scripting သို႔မဟုတ္ server side scripting တစ္ခုခု ဒါမွမဟုတ္ ႏွစ္ခုစလံုးကို သံုးတယ္။

Client side scripting ဆိုတာက သူက user ရဲ႕ computer ေပၚမွာပဲ user ရိုက္ထည့္လိုက္တဲ့ input ေတြ အေပၚမွာပဲ content ေတြကို ထုတ္ေပးတယ္။ဆိုလိုတာက web browser က server ဆီကေန web page ကို download ဆြဲလိုက္တယ္ ၿပီးေတာ့ user ကိုၿပမယ့္ information ေတြကို page ထဲမွာပဲ code ေတြကို render လုပ္လိုက္ၿခင္းကို ဆိုလိုတယ္။ Server side scripting မွာက်ေတာ့ software က server ေပၚမွာ run တယ္။server ထဲမွာ processing လုပ္လို႔ၿပီးရင္ plain pages ေတြကို server ဆီကို ပိုု႔လႊတ္လိုက္တယ္။



|  |  |
| --- | --- |
| Static Website | Dynamic Website |
| Page တက္တိုင္းတက္လိုင္း ၿပတာ အတူတူပဲ | Page တက္တိုင္း ေၿပာင္းလဲေနတယ္ |
| Html သိရင္ ရၿပီ | Technology မ်ိဳးစံုနဲ႔ ေရးလို႔ရတယ္ |
| Request တိုင္းမွာ တူညီတဲ့ response ကိုပဲၿပန္တယ္ | Request တစ္ခုစီတိုင္းမွာ မတူညီတဲ့ response ကိုလည္း ထုတ္ေပးႏိုင္တယ္ |
| Update လုပ္ခ်င္တယ္ဆိုရင္ file ကို update လုပ္ၿပီး server ထဲသြားထည့္ရမယ္ | Page ထဲမွာပါတဲ့ server side code ေတြက page load လုပ္တိုင္းမွာ မတူညီတဲ့ content ေတြကို ထုတ္ေပးေနတယ္ |
| Static website က fix ၿဖစ္တယ္ အဲ့တာက သူရဲ႕ တစ္ခုတည္းေသာ ေကာင္းတဲ့အခ်က္ပဲ | သူ႕ရဲ႕ ေကာင္းတဲ့အခ်က္က CMS ကို ေကာင္းစြာလုပ္ေဆာင္ႏိုင္တယ္ |

**Http (Hyper Text Transfer Protocol)**

Client နဲ႔ Server ၾကားထဲက communication လုပ္ေပးတဲ့ data communication protocol တစ္ခုၿဖစ္တယ္။ Http ဆိုတာက TCP/IP ကိုအေၿခခံထားတဲ့ communication protocol တစ္ခုၿဖစ္ၿပီး image files, query results, HTML files အစရွိတာေတြကို world wide web ေပၚမွာ ေပးပို႔ဖို႔ အတြက္ သံုးတယ္။ သူ႕ရဲ႕ default port ကေတာ့ TCP 80 ၿဖစ္တယ္။



**Http ရဲ႕အေၿခခံက်ေသာ ၀ိေသသမ်ား**

* Web servers မ်ားနဲ႔ browser မ်ားၾကားထဲမွာ data မ်ားကို exchange လုပ္ေပးတဲ့ protocol တစ္ခုၿဖစ္တယ္။
* Request Response Protocol တစ္ခုၿဖစ္တယ္။
* သူက ယံုၾကည္ရတဲ့ TCP Connection မ်ားကို သံုးပါတယ္။သူ႕ရဲ႕ default protocol ကေတာ့ TCP 80 ပါ။
* သူက stateless ၿဖစ္တယ္။ဆိုလိုတာက request တစ္ခုပို႔လိုက္တိုင္း request အသစ္လို႔ပဲ စဥ္းစားခံရတယ္။server က user ကို ဘာမွအသိေပးထားၿခင္း မရွိဘူး။

**Http ရဲ႕ အေၿခခံက်ေသာ အခ်က္မ်ား**

သူ႕မွာ ၃ ခ်က္ရွိတယ္။

* **HTTP is media independent: ဆိုလိုတာက ဘယ္ media type မ်ိဳးၿဖစ္ပါေစ သူက handle လုပ္ႏိုင္တယ္။**
* **HTTP request ပို႔လိုက္ၿပီးတဲ့ေနာက္ server နဲ႔ user က connection ၿပတ္ေတာက္သြားတယ္ဆိုရင္ေတာင္ response ကို ေစာင့္ေပးတယ္။**
* **HTTP Stateless**

**HTTP ရဲ႕ အေၿခခံ architecture**



**Http Requests**

Http requests ဆိုတာက computer တစ္လံုးကေန web server တစ္ခုဆီသို႔ ပို႔လိုက္တဲ့ request မွာ စီထားတဲ့ interesting information ေတြပါ၀င္တယ္ အဲ့တာကို http requests မ်ားလို႔ေခၚတယ္။

User တစ္ေယာက္က server ဆီကို request တစ္ခုပို႔တဲ့အခါ ေအာက္ေဖာ္ၿပပါ information မ်ားပါ၀င္ပါတယ္။

* The Request-line
* The analysis of source IP address, proxy and port
* The analysis of destination IP address, protocol, port and host
* The Requested URI (Uniform Resource Identifier)
* The Request method and Content
* The User-Agent header
* The Connection control header
* The Cache control header

Http request method က **Requested URI (Uniform Resource Identifier) လုပ္ခိုင္းတဲ့ အတိုင္းလုပ္ေဆာင္ရတယ္။ အဲ့ method က case sensitive ၿဖစ္ရမယ္ upper case ၿဖစ္ရမယ္။**

**GET vs POST**

GET နဲ႔ POST ၾကားထဲမွာ မ်ားစြာေသာ ကြဲၿပားၿခားနားခ်က္မွာ ရွိပါတယ္။

|  |  |
| --- | --- |
| GET | POST |
| Data ကို limit amout နဲ႔ပဲ ေပးပို႔လို႔ရတယ္ ဘာလို႔လဲဆိုေတာ့ data ကို header ထဲမွာေပးပို႔ရလို႔။ | Data ကို large amount ေပးပို႔လို႔ရတယ္ ဘာလို႔လဲဆိုေတာ့ data ကို body ထဲမွာေပးပို႔ရလို႔။ |
| Data ေတြ စိတ္မခ်ရဘူး ဘာလို႔လဲဆိုေတာ့ url မွာအကုန္ၿမင္ေနရလို႔။ | Data ေတြက စိတ္ခ်ရတယ္ဘာလို႔လဲဆိုေတာ့ url မွာ မၿမင္ရလို႔။ |
| သူ႕ကို bookmark ထဲမွာ ထည့္သိမ္းထားလို႔ရတယ္။ | သူကေတာ့ bookmark ထဲမွာ ထည့္သိမ္းထားလို႔မရဘူး။ |
| သူက idempotent ၿဖစ္တယ္ အဲ့တာဘာေၿပာတာလဲဆိုေတာ့ second request ပို႔လိုက္လည္း first request ကို အေႏွာက္အယွက္မေပးဘူး first request က သူ႕အလုပ္ကို သူဆက္လုပ္ႏိုင္တယ္။ | သူကေတာ့ idempotent မၿဖစ္ဘူး။ |
| သူက ပိုထိေရာက္တယ္ အဲ့တာေၾကာင့္ post ထက္ပိုသံုးၾကတယ္။ | သူကေတာ့ သိပ္မထိေရာက္ဘူး အဲ့တာေၾကာင့္အသံုးနည္းတယ္။ |

သူတို႔ရဲ႕ အားသာခ်က္ အားနည္းခ်က္ကို သိသြားတဲ့အခါ သူတို႔ရဲ႕အသံုး၀င္ပံုကိုလည္း ပိုနားလည္သြားႏိုင္တယ္။

အဲ့တာက ဘာလဲဆိုေတာ့……

**GET** ကို resource ဆီက data မ်ားကို request လုပ္တဲ့အခါ သံုးတယ္။

**POST** ကို resource ဆီသို႔ processed လုပ္ထားေသာ data မ်ား submit လုပ္တဲ့အခါ သံုးပါတယ္။

**GET Request ကိုခြဲၿခမ္းစိပ္ၿဖာၿခင္း**

တစ္ကယ္တမ္းက်ေတာ့ query string ကို GET Request ရဲ႕ url အတြင္းမွာ ေပးပို႔ၿခင္းပဲၿဖစ္တယ္။

GET/RegisterDao.jsp?name1=value1&name2=value2

အဲ့တာက dataကို request header ထဲမွာ ပို႔တယ္လို႔လည္း ေၿပာလို႔ရတယ္။ဒါက default request တစ္ခုလည္း ၿဖစ္တယ္။



အခ်ိဳ႕ေသာ GET method ရဲ႕ feature ေတြကေတာ့…..

1. Browser history က်န္တယ္။
2. Bookmark ထဲမွာ ထည့္သိမ္းထားလို႔ရတယ္။
3. Cached က်န္တယ္။
4. Length restrictions ရွိတယ္။
5. Sensitive ၿဖစ္တဲ့ data ေတြအတြက္ မသံုးသင့္ဘူး။
6. Data ေတြယူဖို႔အတြက္ေလာက္ပဲ သံုးသင့္တယ္။

**POST Request ကိုခြဲၿခမ္းစိပ္ၿဖာၿခင္း**

တစ္ကယ္တမ္းက်ေတာ့ query string ကို GET Request ရဲ႕ HTTP message body အတြင္းမွာ ေပးပို႔ၿခင္းပဲၿဖစ္တယ္။

POST/RegisterDao.jsp HTTP/1.1

Host: www. javatpoint.com

name1=value1&name2=value2

အခ်ိဳ႕ေသာ POST method ရဲ႕ feature ေတြကေတာ့…..

Bookmark ထဲမွာ မွတ္ထားလို႔မရဘူး။

သူ႕မွာ Data length restriction မရွိဘူး။

Cached လံုး၀မရွိဘူး။

Browser history ထဲမွာ သူ႕ကိုလံုး၀ၿပန္ေဖာ္လို႔မရဘူး။



**Servlet Container**

Java EE applications မ်ားအတြက္ run time environment ကို ထုတ္လုပ္ေပးတယ္။ client က static web pages မ်ားမွာသာ server ဆီကို request ပို႔ႏိုင္တယ္။

Servlet Container ဆိုတာက web server ရဲ႕အစိတ္အပိုင္းတစ္ခုၿဖစ္ၿပီး သူ႕ကို သီၿခား process တစ္ခုအေနနဲ႔ run လို႔ရတယ္။



**Servlet Container States**

၃ ပိုင္းရွိပါတယ္။

1. Standalone: သူက typical java based server ေတြၿဖစ္တယ္။သူထဲမွာ ရွိတဲ့ servlet container နဲ႔ web server's တို႔ေတြက single program တစ္ခုရဲ႕ ၿပီးဆံုးေအာင္ လုပ္ေဆာင္ရမယ့္ အစိတ္အပိုင္းမ်ားၿဖစ္တယ္။ဥပမာ Tomcat က ကိုယ့္ဟာကုိယ္ run ႏိုင္တယ္။
2. In-process: သူကေတာ့ web server က separate ၿပဳလုပ္ၿခင္းခံထားရတယ္ ဘာလို႔လဲဆိုေတာ့ main server တစ္ခုရဲ႕ address space တစ္ခုတည္းမွာ မတူညီတဲ့ program တစ္ခုကို run ထားလို႔ၿဖစ္တယ္ (plugin တစ္ခုအေနနဲ႔) ။ဥပမာ JBoss ထဲမွာ Tomcat ကို run ၿခင္း။
3. Out-of-process: web server နဲ႔ servlet container တို႔ဟာ မတူညီတဲ့ program ေတြၿဖစ္ၾကတယ္။သူတို႔ထဲမွာ မတူညီတဲ့ program ေတြ run ေနၾကတယ္။ သူတို႔ၾကားထဲမွာ ဆက္သြယ္ဖို႔အတြက္ web server က plugin ကိုသံုးပါတယ္။အဲ့ plugin က servlet container ကို ေပးစြမ္းတယ္။

**Servlet Container လုပ္ေဆာင္ႏိုင္ေသာ operations မ်ား**

1. Life Cycle Management
2. Multithreaded Support
3. Object Pooling
4. Security

**Server: Web vs Application**

Server ဆိုတာ တကယ္ေတာ့ computer တစ္လံုး သို႔မဟုတ္ device တစ္ခုၿဖစ္တယ္။ သူက request ေတြကို လက္ခံႏိုင္တယ္ response ၿပန္ႏိုင္တယ္ ဘယ္သူ႕ဆီကိုလည္းဆိုေတာ့ client ေတြဆီကို။ network resources မ်ားကို manage လုပ္ဖို႔အတြက္နဲ႔ program သို႔မဟုတ္ software ေတြ run ဖို႔အတြက္ သူ႕ကိုသံုးတယ္။

သူ႕မွာ ၂ မ်ိဳးရွိတယ္။

1. Web Server
2. Application Server

**Web Server**

Web server မွာ web or servlet container တစ္ခုသာလွ်င္ပါ၀င္ပါတယ္။ သူ႕ကို servlet, jsp, struts, jsf အစရွိတဲ့ technology မ်ားနဲ႔ သံုးစြဲႏိုင္တယ္။ EJB မွာေတာ့ မသံုးပါဘူး။

အဲ့တာက computer တစ္လံုးၿဖစ္ၿပီးေတာ့ အဲ့ထဲမွာ web content ေတြကို သိမ္းဆည္းထားတယ္။ေယဘုယအားၿဖင့္ web server ကို website မ်ား host လုပ္ဖို႔ရန္အတြက္ သံုးတယ္ ဒါေပမယ့္ ဒီေနရာမွာ တစ္ၿခား web server မ်ားကိုလည္းသံုးႏိုင္ပါတယ္ FTP, email, storage, gaming အစရွိတဲ့ web server မ်ားၿဖစ္ၾကတယ္။

Web server မ်ားရဲ႕ ဥပမာကေတာ့ Apache Tomcat ၿဖစ္ပါတယ္။

**Web Server ရဲ႕အလုပ္လုပ္ပံု**

သူက နည္းလမ္း ၂ မ်ိဳးနဲ႔ client ဆီက request မ်ားကို response လုပ္ႏိုင္ပါတယ္။

Web server က script ကို အသံုးၿပဳၿခင္းအားၿဖင့္ database နဲ႔အလုပ္တြဲလုပ္ၿခင္းအားၿဖင့္ response ကို ထုတ္လႊတ္ပါတယ္။

Requested url နဲ႔ စပ္ဆက္ေနတဲ့ file မ်ားကို client ဆီကို ပို႔ေပးတယ္။



**အေရးၾကီးေသာအခ်က္မ်ား**

* အကယ္လို႔မ်ား requested Web page ကို client side မွာမေတြ႕ေတာ့ဘူးဆိုရင္ web server က 404 ဆိုတဲ့ http response ကို response ၿပန္ေပးလိမ့္မယ္။
* Web server က requested web page ကုိရွာေသာအခါ page ကို ေတြ႕ခဲ့မယ္ဆိုရင္ client ကို http response နဲ႔ page ကို ၿပန္ေပးလိမ့္မယ္။
* တစ္ၿခား Resource ဆီက client requests မ်ားလာခဲ့မယ္ဆိုရင္ web server က application server ဆီကို ခ်ိတ္ဆက္ၿပီး HTTP response ၿပဳလုပ္ဖို႔ရန္အတြက္ data မ်ားကို သိမ္းဆည္းပါတယ္။

**Application Server**

Application server မ်ားတြင္ web နဲ႔ EJB container မ်ားပါ၀င္ပါတယ္။သူကိုလည္း servlet, jsp, struts, jsf, ejb စတဲ့ technologies မ်ားနဲ႔ တြဲဖက္သံုးစြဲႏိုင္ပါတယ္။သူက component based product တစ္ခုၿဖစ္ၿပီးေတာ့ server centric architecture တစ္ခုရဲ႕ middle-tier ထဲမွာ အလုပ္ပါတယ္။

သူက State maintenance နဲ႔ security အတြက္ persistence နဲ႔ data access မ်ားနဲ႔အတူ middleware services မ်ားကို ေပးစြမ္းပါတယ္။



ထင္ရွားတဲ့ Application server မ်ားကေတာ့….

* JBoss - JBoss community ရဲ႕ open-source server တစ္ခုၿဖစ္တယ္။
* Glassfish - Oracle ရဲ႕ server ၿဖစ္တယ္။
* Weblogic - Oracle ရဲ႕ server ပဲ ဒါေပမယ့္ သူက ပိုၿပီး secure ၿဖစ္တယ္။
* Websphere - IBM ရဲ႕ server ၿဖစ္တယ္။

**Content Type**

Content type ကို MIME type (Multipurpose internet Mail Extension) လို႔လည္း သိထားၾကတယ္။သူက HTTP Header တစ္ခုၿဖစ္ၿပီးေတာ့ browser ကိုပို႔လႊတ္လိုက္တဲ့ အရာေတြက ဘာေတြလဲဆိုတာကို ေရးထားတဲ့ description မ်ားကို ေပးပါတယ္။

MIME က internet standard တစ္ခုၿဖစ္ၿပီးေတာ့ mail ရဲ႕ extending limited capabilities မ်ားအတြက္ အသံုးၿပဳပါတယ္။ဘယ္လိုနည္းလမ္းေတြနဲ႔လည္းဆိုေတာ့ sounds မ်ား images မ်ားထည့္သြင္းၿခင္း message ထဲတြင္း text မ်ားရိုက္ၿခင္း စတဲ့နည္းလမ္းေတြနဲ႔အသံုးၿပဳပါတယ္။

MIME ရဲ႕ features ေတြကေတာ့…

* non-ASCII characters မ်ားကို support ေပးပါတယ္။
* single message တစ္ခုတည္းမွာ multiple attachment ကို support ေပးပါတယ္။
* attachment ထဲမွာဆိုရင္လည္း executable audio, images နဲ႔ video files မ်ားအစရွိသည္မ်ားကို support ေပးပါတယ္။
* သူက unlimited message length ကို support ေပးပါတယ္။

**List of Content type**

သူ႔မွာ Content type မ်ားစြာရွိတယ္။ေတာ္ေတာ္ေလးကို မ်ားတယ္။အဲ့တာေၾကာင့္မေရးၿပေတာ့ပါ။

**Servlet Api**

 javax.servlet နဲ႔ javax.servlet.http packages မ်ားက servlet api အတြက္ interfaces နဲ႔ classes မ်ားကို ကိုယ္စားၿပဳပါတယ္။

javax.servlet package မွာ interfaces နဲ႔ classes အမ်ားၾကီး ပါ၀င္ပါတယ္။သူတို႔ေတြကို servlet သို႔မဟုတ္ web container က အသံုးၿပဳပါတယ္။သူတို႔မွာ ဘယ္ protocol ၿဖစ္ရမယ္လို႔ တိတိက်က်မရွိဘူး။

javax.servlet.http package မွာဆိုရင္ေတာ့ အေပၚက ေကာင္လိုပဲ interfaces နဲ႔ classes မ်ားစြာပါ၀င္ပါတယ္ ဒါေပမယ့္ သူကေတာ့ http requests မ်ားကိုပဲ တာ၀န္ယူလုပ္ေဆာင္ပါတယ္။

**javax.servlet package မွာရွိတဲ့ interfaces မ်ား**

1. Servlet
2. ServletRequest
3. ServletResponse
4. RequestDispatcher
5. ServletConfig
6. ServletContext
7. SingleThreadModel
8. Filter
9. FilterConfig
10. FilterChain
11. ServletRequestListener
12. ServletRequestAttributeListener
13. ServletContextListener
14. ServletContextAttributeListener

**javax.servlet package မွာရွိတဲ့ classes မ်ား**

1. GenericServlet
2. ServletInputStream
3. ServletOutputStream
4. ServletRequestWrapper
5. ServletResponseWrapper
6. ServletRequestEvent
7. ServletContextEvent
8. ServletRequestAttributeEvent
9. ServletContextAttributeEvent
10. ServletException
11. UnavailableException

**javax.servlet.http packages တြင္ပါ၀င္ေသာ interfaces မ်ား**

1. HttpServletRequest
2. HttpServletResponse
3. HttpSession
4. HttpSessionListener
5. HttpSessionAttributeListener
6. HttpSessionBindingListener
7. HttpSessionActivationListener
8. HttpSessionContext (deprecated now)

**javax.servlet.http packages တြင္ပါ၀င္ေသာ classes မ်ား**

1. HttpServlet
2. Cookie
3. HttpServletRequestWrapper
4. HttpServletResponseWrapper
5. HttpSessionEvent
6. HttpSessionBindingEvent
7. HttpUtils (deprecated now)

**Servlet Interface**

Servlet interface က method မ်ားကို define လုပ္တယ္ သူ႕ကို servlet အကုန္လံုးက implements လုပ္ရတယ္။

အဲ့မွာ Life cycle method ၃ ခုနဲ႔ non-life cycle method ၂ ခုေပးပါတယ္။ life cycle method ၃ ခုက servlet ကို initialize လုပ္ၿခင္း request မ်ားကို service လုပ္ၿခင္း နဲ႔ servlet ကို destroy လုပ္ၿခင္းတို႔ၿဖစ္ၾကပါတယ္။

**Method of Servlet Interface**

သူတို႔ကို Web container က invoke လုပ္ပါတယ္။

|  |  |
| --- | --- |
| Method | Description |
| **public void init(ServletConfig config)** | Servlet ကို initialize လုပ္တဲ့ method တစ္ခုၿဖစ္ၿပီး web container က သူ႕ကိုတစ္ခါပဲ invoke လုပ္တယ္။ |
| **public void service(ServletRequest request,ServletResponse response)** | Request အတြက္ response ၿပန္ေပးပါတယ္။web container ကို request လာတိုင္းလာတိုင္း အဲ့ method ကိုေခၚၿခင္းးခံရတယ္။ |
| **public void destroy()** | Servlect ကို destroy လုပ္တဲ့ေနရာမွာ သံုးတယ္။သူ႕ကိုလည္း တစ္ၾကိမ္ပဲေခၚတယ္။ |
| **public ServletConfig getServletConfig()** | Servlet Config Object ကို return ၿပန္ေပးပါတယ္။ |
| **public String getServletInfo()** | Servlect informationကို return ၿပန္ေပးပါတယ္ ဘယ္လိုဟာေတြလည္းဆိုေတာ့ writer တို႔ copyright တို႔ version အစရွိသၿဖင့္ေပါ့။ |

**Servlet example by implementing Servlet Interface**

ဒီ class ကို ေရးမယ္ဆိုရင္ သူက ပထမဆံုး ေရးမယ့္ class ၿဖစ္တဲ့အတြက္ေၾကာင့္ သူ႕မွာ project မရွိဘူး အဲ့တာေၾကာင့္ maven project တစ္ခုယူမယ္။servlect ေရးမွာၿဖစ္တဲ့အတြက္ servlect jar file မ်ားလိုအပ္လိမ့္မယ္။ servlet အတြက္ pom.xml မွာ ဘယ္လိုေရးရမလဲဆိုေတာ့…..

<dependency>

<groupId>javax.servlet</groupId>

<artifactId>servlet-api</artifactId>

<version>3.0-alpha-1</version>

<scope>provided</scope>

</dependency>

**public** **class** First **implements** Servlet {

//servletConfig ဆိုတာက javax.servlet package ရဲ႕ interface တစ္ခုၿဖစ္တယ္။

ServletConfig config = **null**;

**public** **void** init(ServletConfig config) {

**this**.config = config;

System.***out***.println("servlet is initialized");

}

**public** **void** service(ServletRequest req, ServletResponse res) **throws** IOException, ServletException {

res.setContentType("text/html");

PrintWriter out = res.getWriter();

out.print("<html><body>");

out.print("<b>hello simple servlet</b>");

out.print("</body></html>");

}

**public** **void** destroy() {

System.***out***.println("servlet is destroyed");

}

**public** ServletConfig getServletConfig() {

**return** config;

}

**public** String getServletInfo() {

**return** "copyright 2007-1010";

}

}

**Generic Servlet**

**Generic Servlet Class**

Generic Servlet Class က Servlet, ServletConfig နဲ႔ Serializable interfaces မ်ားကို implement လုပ္ပါတယ္။ service method ကလြဲလို႔ interfaces မ်ားရဲ႕ method အားလံုးကို implementation မ်ားကို ရရွိတယ္။

Generic Servlet Class က မည္သည့္ request type မ်ိဳးကိုမဆို handle လုပ္ႏိုင္တဲ့အတြက္ေၾကာင့္ သူ႕ကို protocol independent လုိ႔ေခၚႏိုင္တယ္။

မင္းက Generic Servlet Class ကို inheriting လုပ္ၿခင္းနဲ႔ေရာ service method ရဲ႕ implementation ကိုေပးၿခင္းအားၿဖင့္ေရာ generic servlet ကို ဖန္တီးႏိုင္တယ္။

**Methods of Generic Servlet Class**

**public void init(ServletConfig config)** servlet ကို initialize လုပ္ဖို႔အတြက္ သံုးတာ။

**public abstract void service(ServletRequest request, ServletResponse response)** user request ပို႔လိုက္တဲ့အခ်ိန္တိုင္းအလုပ္လုပ္တယ္။

**public void destroy()** servlet ကို destroy လုပ္ဖို႔အတြက္သံုးတယ္။

**public ServletConfig getServletConfig()** servletConfig return ၿပန္ဖို႔အတြက္သံုးတယ္။

**public String getServletInfo()** writer, copyright and version ကဲ့သို႔ေသာ servlet ရဲ႕ information မ်ားကို return ၿပန္ဖို႔ရန္အတြက္ သံုးတယ္။

**public void init()**

**public ServletContext getServletContext()** servletcontext ကို return ၿပန္ဖို႔အတြက္သံုးပါတယ္

**public String getInitParameter(String name)** ေပးလိုက္တဲ့ parameter name အတြက္ parameter value ကို return ၿပန္ေပးလိမ့္မယ္

**public Enumeration getInitParameterNames()** web.xml မွာ သတ္မွတ္ထားတဲ့ parameters values အားလံုးကို return ၿပန္ေပးလိမ့္မယ္

**public String getServletName()** servlet ရဲ႕ name ကို return ၿပန္ေပးလိမ့္မယ္

**public void log(String msg)** servlet log file ထဲမွာ ေပးလိုက္တဲ့ message ကို ေရးလိမ့္မယ္

**public void log(String msg,Throwable t)** servlet log file ထဲမွာ ရွင္းၿပတဲ့ message ကို ေရးလိမ့္မယ္

**public class First extends GenericServlet{**

**public void service(ServletRequest req,ServletResponse res)**

**throws IOException,ServletException{**

**res.setContentType("text/html");**

**PrintWriter out=res.getWriter();**

**out.print("<b>hello generic servlet</b>");**

**out.print("</body></html>");**

**}**

**}**

**HttpServlet**

**HttpServlet Class**

HttpServlet က GenericServlet class ကို depend လုပ္ထားတယ္ ၿပီးေတာ့ Serializable interface ကို implements လုပ္ထားတယ္။

**Method of HttpServlet Class**

|  |
| --- |
| **public void service(ServletRequest req,ServletResponse res)** request ကို protected service method ဆီကို ပို႔လႊတ္လိုက္တယ္ ဘယ္လိုနည္းနဲ႔လည္းဆိုေတာ့ request and response object ေတြကို http type အၿဖစ္ ေၿပာင္းလဲလိုက္ၿခင္းၿဖင့္ ပို႔လႊတ္လိုက္တယ္ |
| **protected void service(HttpServletRequest req, HttpServletResponse res)** service method က request ကိုလက္ခံတယ္ ၿပီးေတာ့ doXXX() method ဆီကို request မ်ားအား incoming http request typeအေပၚမူတည္ၿပီးေတ့ာ ပို႔လႊတ္လိုက္တယ္ |
| **protected void doGet(HttpServletRequest req, HttpServletResponse res) GET request ကို handle လုပ္တယ္ ၿပီးေတာ့ web container ကို invoke လုပ္တယ္** |
| **protected void doPost(HttpServletRequest req, HttpServletResponse res)** **POST request ကို handle လုပ္တယ္ ၿပီးေတာ့ web container ကို invoke လုပ္တယ္** |
| **protected void doHead(HttpServletRequest req, HttpServletResponse res) HEAD request ကို handle လုပ္တယ္ ၿပီးေတာ့ web container ကို invoke လုပ္တယ္** |
| **protected void doOptions(HttpServletRequest req, HttpServletResponse res) OPTIONS request ကို handle လုပ္တယ္ ၿပီးေတာ့ web container ကို invoke လုပ္တယ္** |
| **protected void doPut(HttpServletRequest req, HttpServletResponse res)** **PUT request ကို handle လုပ္တယ္ ၿပီးေတာ့ web container ကို invoke လုပ္တယ္** |
| **protected void doTrace(HttpServletRequest req, HttpServletResponse res) TRACE request ကို handle လုပ္တယ္ ၿပီးေတာ့ web container ကို invoke လုပ္တယ္** |
| **protected void doDelete(HttpServletRequest req, HttpServletResponse res) DELETE request ကို handle လုပ္တယ္ ၿပီးေတာ့ web container ကို invoke လုပ္တယ္** |
| **protected long getLastModified(HttpServletRequest req) HttpServletRequest ကို စၿပီး release လုပ္တဲ့ အခ်ိန္ကို ထည့္ေပးလိုက္တယ္(January 1,1970 GMT).** |

**Servlet Life Cycle**

Life Cycle of a Servlet

Servlet instance တစ္ခုရဲ႕ life cycle ကို web container က maintain လုပ္ပါတယ္။

1. Servlet class is loaded.
2. Servlet instance is created.
3. init method is invoked.
4. Service method is invoked.
5. Destroy method is invoked.



ပံုအရဆိုရင္ servlet တစ္ခုမွာ အဆင့္ ၃ ဆင့္ရွိတယ္(new,ready and end). Servlet instance ကို create လုပ္လိုက္တယ္ဆိုရင္ အဲ့တာကို new state လို႔ေခၚတယ္။init method() ကိုေခၚၿပီးတဲ့ေနာက္မွာ servlet ဟာ ready state ကို ကူးေၿပာင္းသြားတယ္။အဲ့ ready state မွာ servlet က task မ်ားကို လုပ္ေဆာင္တယ္။web container က destroy() method ကို လွမ္းေခၚလိုက္ေသာအခါ end state ကိုေရာက္ရွိသြားတယ္။

**1. Servlet Class is loaded**

Class loader က servlet class ကို load လုပ္ဖို႔အတြက္ တာ၀န္ယူတယ္။ web container က servlet အတြက္ ပထမဆံုး request ေရာက္လာတဲ့အခါ servlet class ေတြကို load ဆြဲတင္တယ္။

**2. Servlet instance is created**

Servlet class ကို load ဆြဲတင္ၿပီးတဲ့ေနာက္ Web container က servlet တစ္ခုရဲ႕ instance ကို create လုပ္လိုက္တယ္။ servlet life cycle မွာ servlet instance ကို တစ္ၾကိမ္တစ္ခါတည္းသာ create လုပ္တယ္။

**3. init() method is invoked**

Servlet instance ကို create လုပ္ၿပီးတဲ့ေနာက္ web container က init method ကို လွမ္းေခၚလိုက္တယ္။ init() method က servlet ကို initialize လုပ္ဖို႔အတြက္ သံုးတယ္။init method က javax.servlet.Servlet interface ရဲ႕ life cycle method တစ္ခုလည္းၿဖစ္တယ္။

**public** **void** init(ServletConfig config) **throws** ServletException

**4. Service method is invoked**

Servlet က request ေတြကို လက္ခံရရွိတဲ့အခါ web container က service method ေတြကို request ေတြလက္ခံရရွိတဲ့အခ်ိန္တိုင္း လွမ္းလွမ္း invoke လုပ္ေနတယ္။ servlet ကို initialized မလုပ္ထားဘူးဆိုရင္

**public** **void** service(ServletRequest request, ServletResponse response)

**throws** ServletException, IOException

**5. Destroy method is invoked**

Service မွ servlet instance ကို remove မလုပ္ခင္ web container က destroy method ကို invoke လုပ္လိုက္တယ္။ အဲ့ method က servlet ကို resource ေတြကို clean up လုပ္ခိုင္းလိုက္တယ္ resource ဆိုတာက ဘယ္လိုဟာမ်ိဳးလည္းဆိုေတာ့ memory တို႔ thread တို႔ကို ရွင္းပစ္တာ ကိုဆိုလိုတယ္္။

**public** **void** destroy()

**Servlet Example**

**Servlet Example တစ္ခုကို create လုပ္ဖို႔အတြက္ step မ်ား**

Servlet example တစ္ခုကို create လုပ္ဖို႔အတြက္ အဆင့္ ၆ ဆင့္ရွိတယ္။Servlet example တစ္ခုကို နည္းလမ္း ၃ မ်ိဳးနဲ႔ ဖန္တီးႏိုင္တယ္။

1. Servlet interface ကို implement လုပ္ၿခင္းၿဖင့္လည္းေကာင္း
2. GenericServlet class ကို inheritance လုပ္ၿခင္းၿဖင့္ေသာ္လည္းေကာင္း
3. HttpServlet class ကို inheritance လုပ္ၿခင္းၿဖင့္ေသာ္လည္းေကာင္း servlet ကို create လုပ္ႏိုင္ပါတယ္။

User ေတြအသံုးအမ်ားဆံုးနည္းလမ္းကေတာ့ HttpServlet class ကို inheritance လုပ္တဲ့နည္းလမ္းပဲၿဖစ္တယ္ ဘာလို႔လဲဆိုေတာ့ သူက doGet(), doPost(), doHead() ကဲ့သို႔ေသာ http request specific method မ်ားကို ထုတ္ေပးႏိုင္လို႔ၿဖစ္တယ္။

အခု Example မွာဆိုရင္ Apache Tomcat Server ကို သံုးပါမယ္။ အဆင့္ ၃ ဆင့္ကေတာ့….

1. Directory structure တစ္ခု create လုပ္ၿခင္း။
2. Servlet တစ္ခု create လုပ္ၿခင္း။
3. Servlet ကို compile လုပ္မယ္။
4. Deployment descriptor တစ္ခုကို create လုပ္မယ္။
5. Server start မယ္ ၿပီးရင္ project ကို deploy လုပ္မယ္။
6. Servlet ကို access လုပ္မယ္။

**1. Directory structure တစ္ခု create လုပ္ၿခင္း**

Directory structure တစ္ခု create လုပ္တယ္ဆိုတာ အဲ့ေနရာမွာ မတူညီတဲ့ file type မ်ားကို ထည့္လိုက္ၿခင္းအားၿဖင့္ web container က information မ်ားကို ရရွိၿပီး client ဆီကို response ၿပန္ပါလိမ့္မယ္။

Sun Microsystem ဆိုတဲ့ အဖြဲ႕အစည္းက unique ၿဖစ္တဲ့ standard တစ္ခုကို သတ္မွတ္ေပးခဲ့တယ္ အဲ့ standard ကို ေရးတဲ့သူေတြအေနနဲ႔ လိုက္နာရမယ္။

Servlet class file ေတြက class folder ထဲမွာ ရွိကို ရွိရမယ္ ၿပီးေတာ့ web.xml ကလည္း WEB-INF ေအာက္မွာ ရွိရမယ္။



2. Servlet တစ္ခုကို create လုပ္ၿခင္း

အေပၚမွာ ေၿပာခဲ့သလိုပဲ နည္းလမ္း ၃ ခုနဲ႔ create လုပ္လို႔ရတယ္။ေအာက္မွာ ေဖာ္ၿပထားတဲ့ example မွာဆိုရင္ httpServlet class ကို extends လုပ္ထားတယ္။ဘာလို႔လဲဆိုေတာ့ doGet() method ရဲ႕ implementation ကို လိုခ်င္လို႔။မွတ္ထားရမွာက get request ဆိုတာက default request တစ္ခုၿဖစ္တယ္။

**import** javax.servlet.http.\*;

**import** javax.servlet.\*;

**import** java.io.\*;

**public** **class** DemoServlet **extends** HttpServlet {

**public** **void** doGet(HttpServletRequest req, HttpServletResponse res) **throws** ServletException, IOException {

res.setContentType("text/html");// setting the content type

PrintWriter pw = res.getWriter();// get the stream to write the data

//writing html in the stream

pw.println("<html><body>");

pw.println("Welcome to servlet");

pw.println("</body></html>");

pw.close();// closing the stream

}

}

**3. Servlet ကို compile လုပ္ၿခင္း**

Servlet ကို compile လုပ္တဲ့အခါ jar file ေတြ လိုအပ္တယ္။server ေပၚမူတည္ၿပီး jar file မ်ားလည္း ကြဲၿပားတယ္။

|  |  |
| --- | --- |
| Jar file | Server |
| servlet-api.jar | Apache Tomcat |
| weblogic.jar | Weblogic |
| javaee.jar | Glassfish |
| javaee.jar | JBoss |

Jar file ကို load လုပ္ဖုိ႔လည္း နည္းလမ္း ၂ မ်ိဳးရွိပါတယ္။

1. Set classpath
2. JRE/lib/ext folder ထဲမွာ jar file ကို paste လုပ္လိုက္တဲ့ နည္းလမ္းၿဖစ္ပါတယ္။

Java file ကိုေတာ့ မည္သည့္ folder မွာမဆိုထည့္လိုက္။ java file ကို compile လုပ္ၿပီးတဲ့ေနာက္မွာ servlet ရဲ႕ class file ကို **WEB-INF/classes** ထဲမွာ paste လုပ္လိုက္ရမယ္။