**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 တစ္ခုလည္း ဟုတ္တယ္။
* Servlet ဆိုတာ class တစ္ခုလည္း ဟုတ္တယ္ သူက မည္သည့္ request ကိုမဆို response ၿပန္ႏိုင္တယ္။
* Dynamic web page တစ္ခုကို ဖန္တီးဖို႔ရန္အတြက္ server ေပၚမွာ deployed လုပ္ၿခင္းခံရတဲ့ web component တစ္ခုလည္း ဟုတ္တယ္။



**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 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 dynamic change ေနၾကတယ္။ သူက content ေတြကို access လုပ္ေပးေနတယ္။အဲ့ content ေတြက

ထြက္လာလဲဆိုေတာ့ Database သို႔ CMS က 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။ 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**

