Servlet & JSP – 01 – Giris ve Temel Kavramlar

injavawetrust.com/servlet-jsp-01-giris-ve-temel-kavramlar/

Levent Erguder 15 March 2014

Merhaba Arkadaslar,
Bir suredir planladigim Servlet&JSP(Java Server Pages) ile ilgili yazilarima bu yazi ile basliyorum. Sonunu getirmek nasip olur insallah.

“Ve ma tevfîkî illa billah aleyhi tevekkeltü ve ileyhi ünîb”
“…Başarım ancak Allah’tandır, O’na güvendim; O’na yöneliyorum.”
Hûd suresi /88

Neden Servlet&JSP?

Gunumuzde Java ile calisan sirketler (telekom,finans,e-ticaret,sigortacilik sektoru vs) projelerini **JSP** veya **JSF** tabanli yapmaktadir. JSP teknolojisi , JSF e gore daha eski bir teknolojidir fakat guncelligini hala korumaktadir .

Java Web teknolojisinin mantigini anlamak icin oncelikle **Servlet** ozellikle bilinmelidir. Sektorel acidan JSP ve JSF in de bilinmesi faydali olacaktir.

Suanki isimde Servlet+JSP kullandigim icin ve Oracle Certified Expert, Java EE 6 Web Component Developer sinavina hazirlandigim icin 2014 yilinda ozellikle bu konulara agirlik verecegim.

Guncelleme : Bu sinavi 30 Agustos 2014 tarihinde gectim , daha fazla bilgi icinOracle Certified Expert, Java EE 6 Web Component Developer Sinavi Hakkinda

Ilerde Java EE kariyeri isteyen arkadaslarin mutlaka Servlet,JSP JSF gibi yapilari ogrenmesi gerekmektedir.

Bu bolum giris niteliginde olacaktir , temel kavramlari anlamak en onemli adimdir. Ilerleyen bolumlerde yeri geldikce daha detaylica inceleyip farkli bakis acilarindan tekrar bakacagiz.

Java Web dunyasina ilk adimimizi atalim ..

Servlet Nedir ?

Servlet her seyden once bir Java sinifidir. Dolayisiyla bildigimiz sularda yuzuyor olacagiz. **Servlet** kavramini daha iyi anlayabilmek icin oncelikle**Sunucu**/**Server** kavramini ve Sunucunun(Server) sorumluluklarini anlamamiz gerekmektedir.

Server(Sunucu) Nedir Sorumluluklari Nelerdir ?

Sunucu(Server) hizmet veren bir fiziksel makine olabilecegi gibi bir **web server application** da olabilir (software).

Sunucunun(Server) iki temel sorumlulugu vardir ;

- Istemcinin(Client) istegini(request) karsilamak(handle)
- Bu istege karsilik gelecek cevabi(**response**) geri gondermektir.
  (Bi nevi para-cokomel parayi ver cokomeli al para cokomel para cokomel)

Server istegi alir(**handle request**), istenen kaynagi(**resource**) bulur ve istemciye geri gonderir dedik peki bu kaynaklar(**resource**) neler olabilir ?

Istenen kaynak ; bir HTML sayfasi , PDF dosyasi , resim dosyasi vb olabilir bu onemli degil, istemci(client) kaynagi ister ve Sunucu(Server) da bu kaynagi(resource) istemciye(client) gonderir tabi istenen kaynak mevcut oldugu surece.

Client(Istemci) Nedir Sorumluluklari Nelerdir ?

Istemci(Client), Sunucudan(Server) kaynak (resource) isteginde bulunur(request) ve gelen cevabi(response) uygun formatta gosterir.

Client(istemci) den kastimiz **browser**(tarayici) ve/veya **kisidir**.

Browser/tarayici ; Server ile nasil iletisim kurabilecegini bilen programlardir, evet bildigimiz (Chrome, Firefox vb.)

Browser/tarayicilarin bir diger gorevi tabiki HTML kodlarini yorumlamak(**interpreting**) ve kullanicilar icin bu kodu gorsel hale cevirmektir(**rendering**)

HTML ve HTTP

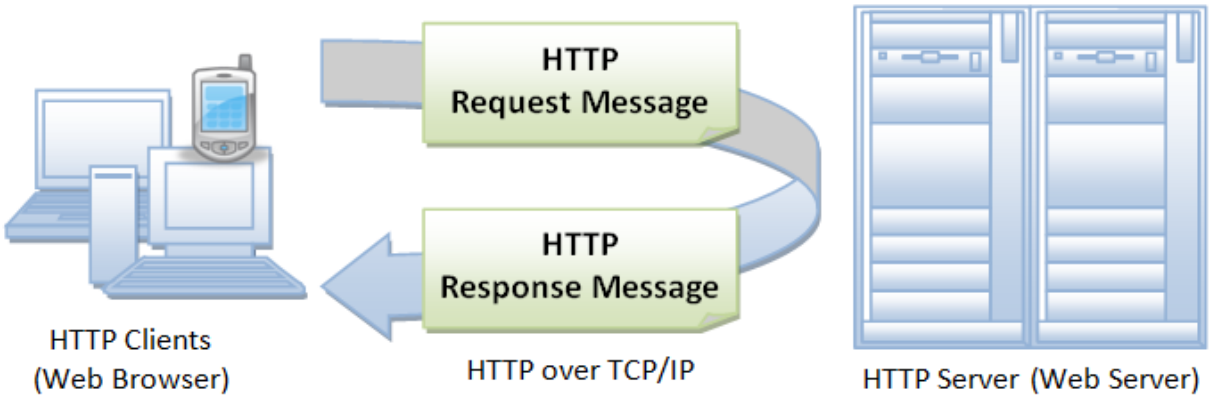
Client(Istemci) ve Server(Sunucu) **HTML** ve **HTTP** bilmektedir. Burada istemciden kastimiz tarayici/browserdir. Tabi bir yazilimci olarak bizim de **HTML** bilmemiz gerekmektedir.

HTTP bir TCP (Transmission Control Protocol) protokoludur acilimi **Hyper Text Transfer Protocol** ‘dur.

Burada bizim icin en onemli olan nokta HTTP nin istemci(client) ve sunucu(server) arasinda web uzerinden iletisim kurmasini saglayan bir protokol oldugunu bilmemizdir.

Sunucu, istemciye **HTML** dosyasini gondermek icin **HTTP** kullanir.

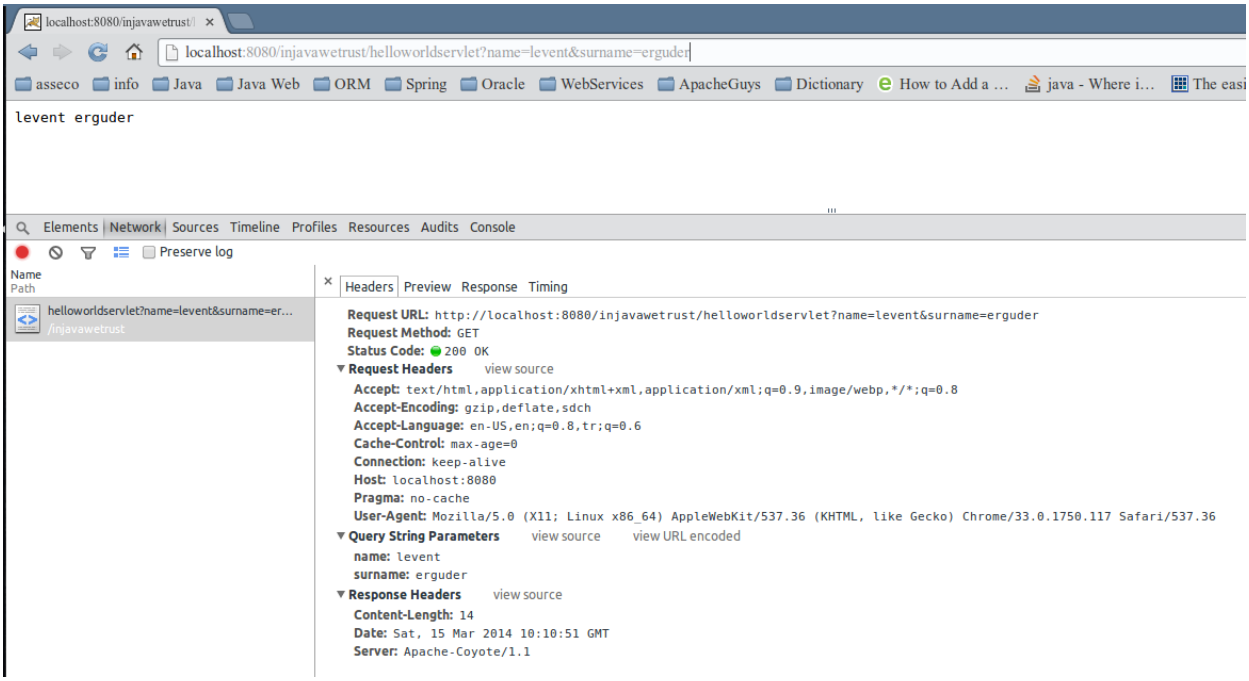
Istemci , sunucuya HTTP request(istegi) gonderir , sunucu da HTTP response(cevabi) gonderir. Kisacasi Sunucunun(Server) dili HTTP dir.



Burada kisaca HTTP' nin **GET** ve **POST** metotlarini inceleyecegiz.
GET metodu en temel HTTP metodudur. Temel gorevi Sunucuya(server) istenen kaynagi sormak ve bu kaynagi getirmektir. Hatirlayacagimiz gibi bu kaynak bir HTML, JPEG , PDF vb dosyalar olabilir.

POST metodunun temel amaci Sunucuya(server) , "form" datasi gondermektir.

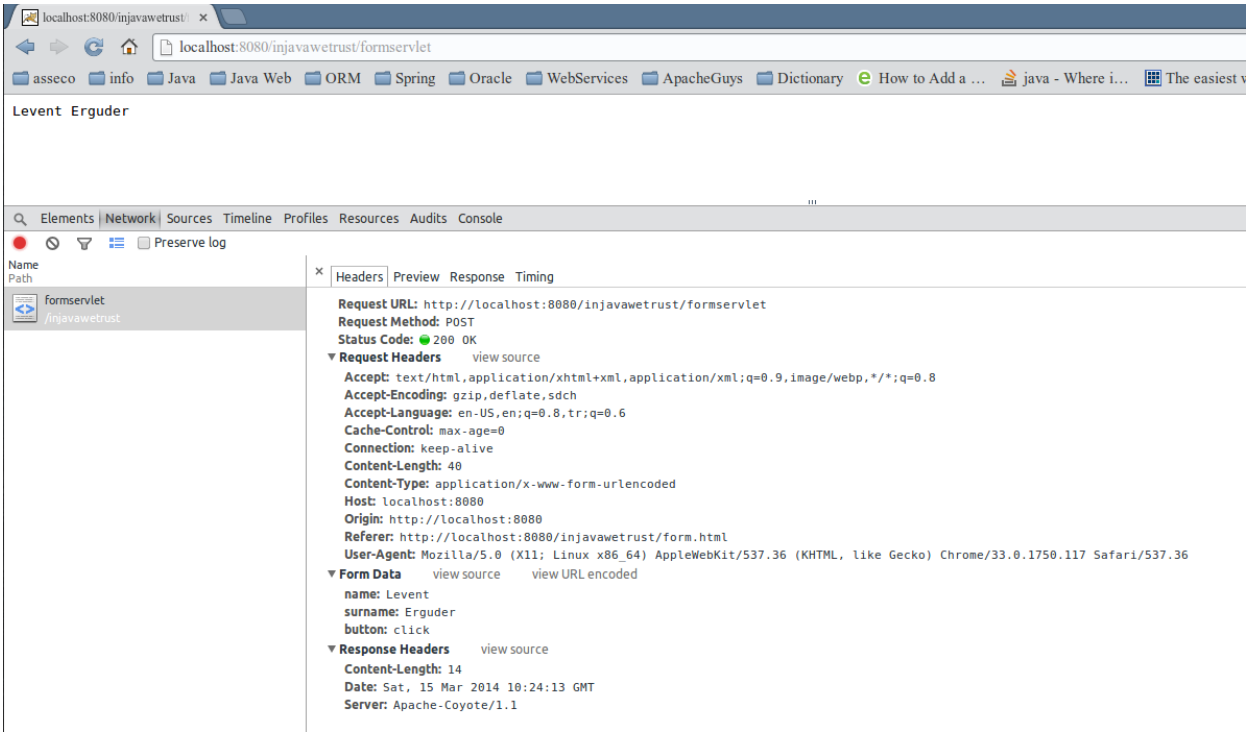
Bu iki metot haricinde baska HTTP metotlari da mevcuttur. Fakat bunlar en cok kullanilan metotlardir. Ilerleyen bolumlerde **GET** ve **POST** metotlarini daha detaylica inceleyecegiz diger metotlari da yuzeysel olarak taniyacagiz.



Google Chrome da HTTP mesajlarini inceleyebiliriz.
Bunun icin F12 ye basip Network sekmesine geldikten sonra mesaja tiklamamiz yeterlidir.

Burada basit bir Servlet ornegi calismakta. GET mesajinin temel gorevi Sunucudan istenen kaynagi almak demistik bununla birlikte bir miktar veriyi(data) gonderebiliriz.

Soru isareti **(?)** path ve parametre bolumlerini ayirir ve **(&)** isareti yardimi ile birden fazla parametre bilgisi gonderebiliriz. Bu gonderilen bilginin URL de gonderildigini bilmemiz simdilik yeterli olacaktir. Resimde gorulen kisimlari ilerleyen bolumlerde detaylica incelecegiz. Bir HTTP request ve Response mesaji hangi alanlardan olusuyor goz asinaligi olmasi acisindan incelemek ilk asamada faydali olacaktir.

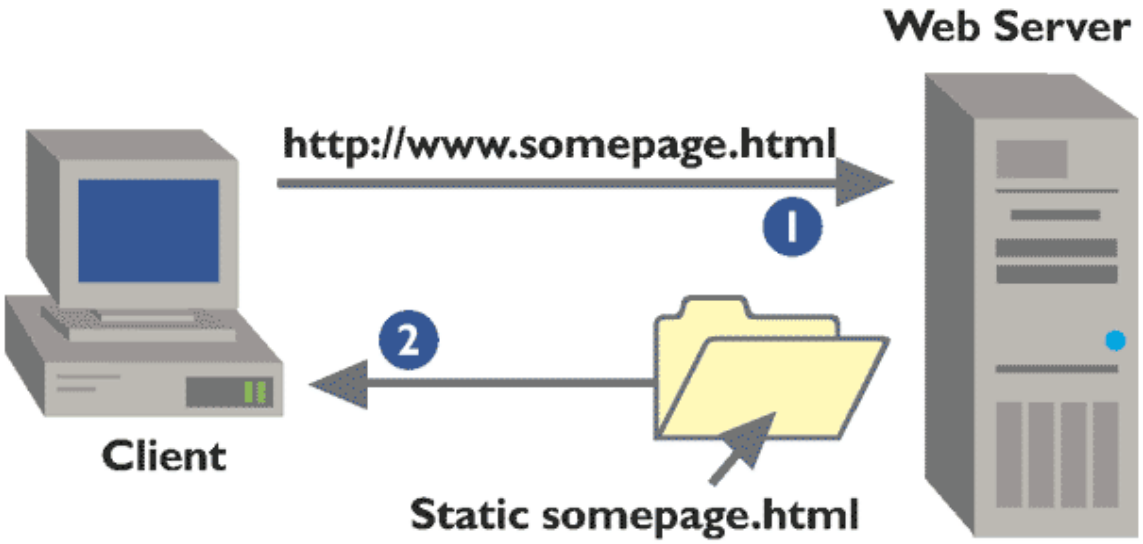


HTTP **POST** metodunda resimdeki gibi bir yapisi vardir. GET metodunun da Request Header alani varken POST metodunda **Request Header** alaniyla birlikte "**Form Data**" alani da yer almaktadir. Bu "Form Data" alanina **Message Body** ya da **Payload** denilmektedir.

Burada dikkat edecegimiz nokta "Form Data" da yer alan parametrelerin URL de yer almamasidir. Bu POST metodunu GET metodundan daha guvenli bir hale getirmektedir.

Sunucu ve Static Web Pages

Static sayfalar ilgili dizinde yer alirlar, Sunucu da bu static sayfalari bulur ve istemciye geri dondurur. Her istemci ayni seyi gorur.



Static sayfalar ilgili dizinde yer alirlar, Sunucu da bu static sayfalari bulur ve istemciye geri dondurur. Her istemci ayni seyi gorur.

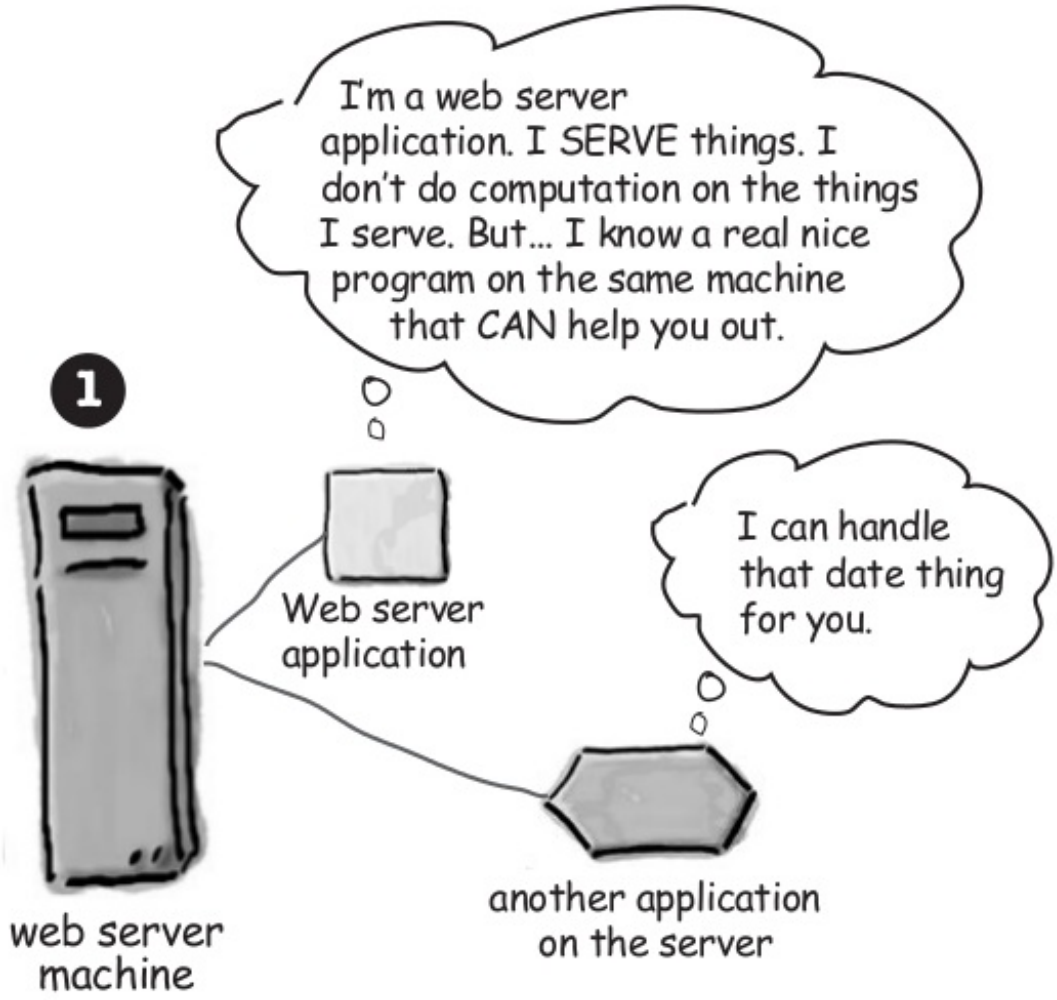
Fakat cogu zaman bundan daha fazlasi gerekir. Web server bize dinamik icerik (dynamic

content) ornegin database islemleri konusunda yardimci olamaz.

Servlet Container

Web Server sadece static sayfalar konusunda hizmet saglayabilir. Bununla birlikte yardimci bir uygulama(application) bize bu dinamik icerik hizmetini saglayabilir.

Web Server , **Servlet'leri** yuklemek(load) ve calistirmak(run) icin ayri bir modul kullanir. Bu ozellesmis ve **Servlet** yonetiminden sorumlu yapiya **Servlet Container** ya da **Servlet Engine** denilir.



Servlet Container ,**Web Server Application**'in bir parcasidir. Yapisi itibariyle bir kac cesiti mevcuttur. Standalone, In-process, Out-of-process gibi.

Servlet'ler bu yardimci uygulama/parcada yani **Servlet Container** tarafindan kontrol edilen Java siniflaridir. Bu tanim suan icin cok yuzeysel fakat simdilik yeterli olacaktir.

Apache Tomcat

Servlet ve JSP derslerimizde **Apache Tomcat** 'i kullanacagiz. **Apache Tomcat** acik kaynak kodlu bir web sunucu(**open source web server**) ve ayni zamanda servlet container ozelligine sahip bir yazilimdir.

Tomcat, Java Servlet ve JSP icin belirtimleri/spesifikasyon gerceklestirir(implementation)

ve HTTP Web Server olarak calisir.

Burada temel ve yuzeysel olarak bir giris yaptik. Ilerleyen bolumlerde tekrar Servlet ve Servlet Container konusunu ele alacagiz.

Yazimi burada sonlandiriyorum.
Herkese Bol Javali Gunler dilerim.
Be an oracle man , import java.;*
Levent Erguder
OCP, Java SE 6 Programmer
OCE, Java EE 6, Web Component Developer