PROGRAMMERINGSSPRÅKET PYTHON

OCH

WEBPROGRAMMERING

Jouni Peltonen

version: 2.5 3.4.2011

Kandidatavhandling i datateknik
Handledare: Annamari Soini

Institutionen för informationsteknologi
Åbo Akademi
4 april 2011 Åbo

Referat

Peltonen, Jouni: Programmeringsspråket Python och webprogrammering.

Kandidatavhandling utfört under handledning av Annamari Soini.

I denna avhandling presenteras först kort webprogrammeringens historia och nuvarande riktning, hur programmering för webben har ändrat genom tiden och vilka krav det ställer på programmeringsspråk, som används för webprogrammering.

Sedan introduceras Python-språket och dess egenskaper allmänt. Därefter koncentrerar avhandlingen på språkets lämplighet för webprogrammering. Python-språkets styrka och svaghet som programmeringsspråk för webapplikationer diskuteras och Python jämförs med några av de mest använda webprogrammeringsspråken, som C#, Java och PHP.

Det huvudsakliga målet med denna avhandling är att presentera ett modernt webprogrammeringsspråk och att ge en inblick i det sammanhang i vilket språket används. I andra hand ges en insyn i de krav som ställs på moderna webprogrammeringsspråk.

Sökord: Python, programmeringsspråk, webprogrammering, webapplikationer, ramverk

Innehåll

1 Inledning

Under de senaste åren har övergången av både privata och offentliga tjänster till internet ökat exponentiellt. I dag möter webapplikationer flera krav, som skiljer sig avsevärt från både traditionella mjukvara krav och dem, som gamla statiska websidor ställde. Websidor har förändrats till interaktiva webapplikationer. Användarnas och deras användningsområdens mångfald ställer särskilda krav på webapplikationer. När kraven på webapplikationer ökar, ökar också kraven på verktyg som används för att skapa dem.

Tidigare var största delen av dynamiska websidor HTML-sidor med kodbitar, som producerade dynamiskt innehåll. På serversidan var detta implementerat med särskilt gränssnitt för webservrar och webapplikationer, som exekverade program på servern. På klientsidan användes oftast appletprogram, som exekverade i webläsaren. Idag är dynamiska websidor oftast implementerade med skriptspråk, som exekveras på servern. I modern webutveckling har programmeringsspråket stigit till samma prioritetsnivå, som den har vid traditionell programmering. Webben sätter vissa unika begränsningar för webprogrammeringspråket. Det stora urvalet av webprogrammeringsspråk, som nu finns har uppstått ur en önskan att utveckla ett fullständigt och perfekt webprogrammeringsspråk, som möter des krav.

Programmeringsspråk skapas ofta genom att vidareutveckla existerande språk för att förbättra vissa områden, som upplevs viktiga i webprogrammering. Nya programmeringsspråk är utvecklade för att möta de förändrade kraven. Eftersom det inte är troligt att någon av dessa konkurrerande språk skulle vinna och ensamt bli det enda rätta, kommer nya webprogrammeringsspråk fortsättningsvis dyka upp.

Så länge som webapplikationer har olika behov, kommer också språkens varierande egenskaper att göra vissa språk mera lämpliga för ändamålet än de andra. Att hitta rätt språk kan vara svårt, eftersom på samma sätt som med traditionell programmering, finns det flera språk som kan passa tillämpningsområdet. Vid valet av programmeringsspråk, måste man ofta göra kompromisser och betona vissa egenskaper på bekostnad av andra. Det finns inte programmeringsspråk, som utför alla uppgifter bättre än sina konkurrenter och fyller alla webprogrammerarens krav. Vid val av språk måste man ta hänsyn till egenskaperna hos språket och kraven på den önskade webtjänsten.

Python är ett mångsidigt högnivåspråk. Python har utvecklats på önskan av att skapa ett språk som är lätt att läsa, lätt att lära sig och roligt att använda. Språket lämpar sig för alla typer av programmering och skapande av applikationer av alla sorter, också webapplikationer. Många av Python-språkets funktioner, såsom renhet, utvecklingshastighet och användarvänlighet, gör att webprogrammerare snabbt kan bygga webapplikationer, som är eleganta och lätta att underhålla. Precis vad utvecklarna önskar.

Python har redan i flera år varit ett av de snabbast växande webprogrammeringsspråken enligt Tiobe Software -företagets programmeringsspråksmätare. Språkets ökande popularitet har i år första gången anammat ett av Öppen källkod -utvecklarnas favoritbarn PHP. (Tiobe, 2011) Trots detta är Python fortfarande en utmanare inom webprogrammeringsspråken och är ännu klart efter de populäraste språken, som till exempel Java och C. Orsaken till den starka förändringen i popularitet kan förklaras i utvecklingen av tekniken hur webapplikationer exekveras på webservern.

Python-programmets exekvering genom CGI (Common Gateway Interface), standardgränssnittet för webservrar och webapplikationer, har varit möjligt sedan början av språkets historia, men det har varit ganska komplicerat och arbetskrävande jämfört med konkurrenterna, som har haft alla nödvändiga inställningar redan färdigt på servern. Med Python WSGI (Web Server Gateway Interface), språkets eget gränssnitt, har programmerarnas arbete blivit mycket lättare. WSGI har möjliggjort programmering utan en förutbestämd serverplattform och färdiga program kan lätt överföras från en serverplattform till en annan.

Ännu större hjälp för snabb och enkel webprogrammering har varit utvecklingen av webapplikationramverk. Ramverk erbjuder en samling av genvägar och beprövade lösningar på olika uppgifter, som underlättar webutvecklarens arbete. Först och främst gör ramverk det enklare och snabbare att installera applikationer på webservern. Genom att installera ett ramverk säkerställer man också, att de mest komplexa gränssnittinställningar går rätt.

Python-språkets stora urval av mångsidiga ramverk med högkvalitet, har lyft Python, från “lillebrorstatuset” den hade före ramverktiden till en av de främsta webprogrammeringsspråk, som används i dag. Python tillsammans med sina ramverk

tävlar i dag helt jämnstarkt med webbens mest populära programmeringsspråk C, Java och PHP.

2 Bakgrund

Även om Internet skapades redan på 60-talet har utvecklingen av World Wide Web (WWW) skett ganska nyligen. Under de senaste tjugo åren har utvecklingen av webben ändå varit snabb och ändringen såväl i användarändamålet, som teknologin drastiskt.

Enligt World Wide Web Consortium (W3C), var World Wide Web ursprungligen tänkt för länkande och bläddring av dokument enligt specifikationer skrivna av Tim Berners-Lee 1989. (W3C, 2009) Under 1990-talet växte WWW eller webben, såsom den nu också var kallad, till en global plattform för flera typer av tjänster. Företagen utnyttjade webben främst för att marknadsföra sin egen verksamhet. Offentlig förvaltning började också flytta tjänster, som sina blankettbibliotek till webben. Websidorna var statiska och de uppdaterades sällan om alls, eftersom att det krävdes professionella webproggrammerare, att ändra på sidorna.

I slutet av 1990-talet började webbens kommersialisering. (Deitel, Deitel, & Nieto, 2001) Nya kommersiella webtjänster föddes, till exempel webbutiker och webauktioner. Google bildades och annonsering och betald reklamföring började sprida sig. Ett stort behov av interaktivitet och dynamiska sidor uppstod och detta behov besvarades av små bitar skriptspråk, till exempel JavaScript, som var inbäddade i HTML-koden och exekverades i webläsarens tilläggsdel. Ett annat alternativ var små program, så kallade applets, som exekverades på klientsidan av andra program. (Deitel, Deitel, & Nieto, 2001) Webläsarkriget var som hetast mellan Netscape och Microsofts Internet Explorer. De var inte förenliga med varandra och använde mycket olika tekniker. HTML- och CSS-standarderna stöddes dåligt. Allt detta gjorde utveckling av webapplikationer på klientsidan mycket svår och större program började implementeras på serversidan.

I början av 2000-talet lanserades begreppet Web 2.0, som hänvisar till en övergång till mera interaktiva webapplikationer och ett mera socialt tillvägagångssätt för produktion och distribution av innehåll, som betonar öppen kommunikation, decentralisering av beslutsfattande samt fritt informationsutbyte och återanvändning. (TechPluto, 2010)

Övergången av både privata och offentliga tjänster till webben ökade kraftigt. Webben blev ett socialt medium. Trots de problem, som följde av att webben inte var avsedd för applikationer, förändrades en ökande andel av statiska websidor till interaktiva webapplikationer. Webapplikationerna erbjöd dynamiskt och interaktivt innehåll till slutanvändare och lättare underhåll för utvecklarna. Det skapades webtjänster, som sammanställer information och funktionalitet från flera oberoende källor.

2.1 Webprogrammering idag

I stort sätt alla offentliga och kommersiella websidor är egentligen webapplikationer. Till och med för att publicera statiska sidor används ofta webapplikationer för att underlätta uppdateringen. Företag har alltmer börjat använda Software as a Service (SaaS) -modellen och flyttat sina egna applikationer till webben. (Ho, 2008) Webben surfas med flera olika terminaler förutom datorer, till exempel med mobiltelefoner. Kriget mellan webläsare har allt flera parter, som Microsoft IE, Mozilla Firefox, Google Chrome, Apple Safari och diverse mobilawebläsare, men samtidigt följs HTML- och CSS-standarderna noggrannare.

Programmeringen på klientsidan sker i allt starkare HTML-, XHTML-, XML- och JavaScript-språk. Genomslaget av HTML5 kommer ytterligen att bredda möjligheterna för vad man kan utföra på klient-sida. Med Flash-, JavaFX- och Silverlight-plattformarna, skapar man så kallade Rika Internet Applikationer (Rich Internet Applications), som har ersatt applets som klient-sidans webapplikationer. (Stamos, Thiel, & Osborne, 2008) Några nya asynkrona tekniker, till exempel AJAX har uppfunnits för att förenkla och försnabba interaktionen mellan webläsaren, webapplikationen och det bakomliggande systemet. (Garrett, 2008) Det finns alltså flera webarkitektur till många olika användningsområden.

Enligt Web Engineering (Gerti Kappel... [et al.] dpunkt.verlag GmbH 2003) är den traditionella klient-server-arkitekturen också lönsammare än någonsin tidigare. Nätverksanslutningarnas stigande hastighet och servrarnas allt större resurser har gjort det lönsammare att köra webapplikationer på servrar, vilket i sin tur har förbättrat applikationernas säkerhet och pålitlighet. (Kappel, Pröll, Reich, & Retschitzegger, 2003) Nu får ett stort antal användare samtidigt tillgång till applikationer, som kräver en

hel del resurser. Flera användare kan komma åt och dela information samtidigt från olika delar av världen. Detta har gjort det möjligt för traditionella desktopprogram, som kartapplikationer, att flytta över till webben. När applikationer körs på servrar är de också lättare att underhålla och uppgradera.

Förbättring av säkerhet och effektivitet samt underlättande av underhållning och uppdatering har för sin del påskyndat övergången av bank- och handelstjänster till webben, men också skapat helt nya, såsom stora socialnätverkstjänster. Frågan är nu hur man utvecklar moderna, dynamiska och interaktiva webapplikationer idag för en plattform som inte är tänkt för dem? Vilka krav finns det för moderna webapplikationer? Vilka verktyg finns det för att underlätta utvecklingsarbetet?

3 Kraven för webapplikationer

Kraven på webapplikationer idag skiljer sig avsevärt från både traditionella mjukvarakrav och de krav statiska websidor ställde. Webben har blivit en viktig plattform för komplexa och sofistikerade applikationer, gjorda för varierande användningsändamål. Användarens och dess användningsmiljös mångfald, tillsammans med applikationernas tillämpningsområden ställer särskilda krav, men det finns också många gemensamma krav för webapplikationer. De viktigaste gemensamma kraven kan karaktäriseras som följande:

3.1 Implementbarhet och kostnadseffektivitet

Webapplikationen måstes ofta utvecklas snabbt, enkelt och billigt. Webben är allt mer en informationskanal och anpassning till hastigt förändrade situationer är viktigt. (Kappel, Pröll, Reich, & Retschitzegger, 2003) Ibland måste webapplikationer kunna utvecklas och implementeras mycket snabbt, men samtidigt bör utvecklingskostnaderna hållas låga.

3.2 Prestanda

En av de kritiska kvalitetsegenskaperna av en webapplikation är prestanda. (Kappel, Pröll, Reich, & Retschitzegger, 2003) Webapplikationer måste vara effektiva och

snabba. De måste kunna klara av stora beräkningar i acceptabel tid. (Koskenniemi, Saastamoinen, & Eerola, 2007) Jämfört med traditionella applikationer är svarstiderna för webapplikationer längre och mera slumpmässigt varierande. Webapplikationernas svarstid (Response Time) kan delas i sidostorlek (page size), turer (turns), bandbredd (bandwidth), tur och returtid (round trip time), serverns behandlingstid (server processing time) och klientens behandlingstid (client processing time). (Savoia, 2001) Sidostorlek och turer anger hur mycket data överförs mellan servern och klienten. Bandbredd och tur och returtid, anger hur snabbt data överförs och behandlingstider anger hur snabbt uppgifterna beräknas. Oberäknelig variering av alla dessa sätter en särskild utmaning för utveckling av webapplikationer. En webapplikation ska vara skalbar för olika slags nätverksanslutningar och antal användare, så att den är tillgänglig i alla situationer, både för mobila terminaler och kabelanslutningar. (Peltomäki, 1998)

3.3 Säkerhet

Säkerhet är viktigt för webapplikationer, emedan de är tillgängliga för en stor mängd användare. (Kappel, Pröll, Reich, & Retschitzegger, 2003) Webapplikationen måste kunna identifiera användaren och säkerställa att endast den har tillgång till data, som kan innehålla en hel del viktig information. Det är viktigt att användaren kan lita på att hans data är skyddat och användarens egen dator är skyddad då applikationen används. (Koskenniemi, Saastamoinen, & Eerola, 2007) Information måste kunna skickas mellan användarens dator och servern, utan att någon annan kan läsa eller ändra det.

3.4 Skalbarhet och portabilitet

Under sin livstid måste webapplikationen kunna överföras till olika serverplattformar, och utökas så att den möter ändrande behov. Den måste också vara användbar med växande utbud av olika terminaler, såsom dator, mobiltelefon, surfplatta eller TV. När man utvecklar webapplikationer är det oftast svårt, om inte helt omöjligt, att göra antaganden om användarens terminal, nätverksanslutningar eller användningssituationer. Webapplikationen måste alltså vara skalbar för många olika anslutningshastigheter, användningsbehov och varierandeutrustning. (Koskenniemi, Saastamoinen, & Eerola, 2007)

3.5 Underhållbarhet

Webapplikationen kräver ständig uppdatering samt underhållande hela sin livscykel. Tekniken förändras snabbt och webapplikationen måste ständigt kunna uppdateras och utvecklas. Detta måste vara möjligt även om utvecklarna har bytts ut. De nya utvecklarna måste kunna lära sig webapplikationen snabbt och kunna fortsätta med underhållande och utveckling. (Kappel, Pröll, Reich, & Retschitzegger, 2003)

3.6 Kvalitet och pålitlighet

Webapplikationer bör vara av högkvalitet och pålitliga. Webapplikationer har blivit skyltfönster för företag och organisationer och inverkar på den bild ett företag ger av sig. Företagets hela utkomst också vara beroende av webapplikationens pålitlighet. Det får således inte förekomma långa avbrott i tjänsten eller andra problem i användning av webapplikationen. (Koskenniemi, Saastamoinen, & Eerola, 2007)

3.7 Användbarhet

Webapplikationerna måste vara användbara. (Koskenniemi, Saastamoinen, & Eerola, 2007) Är en applikation inte användbar kommer den inte att användas.

3.8 Sammanfattning

Moderna webapplikationer ställs krav, som skiljer sig från de traditionella mjukvarakraven och de krav som gamla statiska websidor hade. När kraven på webapplikationer ökar, ökar också kraven på verktyg som används för at skapa webapplikationerna.

4 Kännetecken för bra webprogrammeringsspråk

Alla dessa särskilda krav som webapplikationer har, ställer krav också på verktygen som väljs för att utveckla webapplikationerna. Med de valda verktygen måste man kunna hantera en mängd olika protokoll och format. Man måste klara av olika programmeringsuppgifter och kunna hantera prestanda, säkerhet, portabilitet, samt ha

förmåga att infoga andra webverktyg och språk. För utveckling av stora och komplexa webapplikationer behövs plattformsoberoende och säkra programmeringsspråk.

4.1 HTML

HTML (HyperText Markup Language) är ett internet baserat språk, som alla webläsare förstår. Det är ett märkningsspråk (markup language) i stället för ett fullt programmeringsspråk. Ett HTML-dokument är text, som har inbäddade märkningar (markup), som påverkar hur texten ser ut i webläsaren. HTML-implementerings grundfunktionssätt är att ladda ner dokumentet utgående från dess plats och namn (t.ex. URL), tolka HTML-märkningarna och visa dokumentet enligt dem. (Ford, Wells, & Wells, 1998) Processen och språket är mycket enkla och det är HTML-språkets styrka. Det finns bara två brister i HTML: dess bristande förmåga att utföra vissa uppgifter och dess låga prestanda i utförandet av uppgifterna. (Peltomäki, 1998) Enbart HTML uppfyller alltså inte kraven för de språk man utvecklar moderna webapplikationer med. HTML måste alltså få hjälp av starkare programmeringsspråk.

4.2 Hur kan man utvärdera programmeringsspråk?

I boken Concepts of Programming Languages (Robert W. Sebesta 4th ed.1999) framställs att programmeringsspråk har sina skilda områden (domains) de är utvecklade för och därmed specialiserade, att passa vissa uppgifter bättre än andra. Utvärderingen av ett visst programmeringsspråk är således mycket beroende av det användningsområdet den ska användas i. (Sebesta, 1999)

Ett programmeringsspråk kan passa flera områden, men varje område har sin egen samling av utvärderingskriterier enligt vilka språkets lämplighet för domänen är dömd. Vilka kriterierna är, kan naturligtvis diskuteras, men det finns vissa allmänna kriterier, som man kan enas om att är viktiga. Tillsammans med de kriterier, som introducerats för webapplikationer, kan man komma fram till vissa unika krav för ett bra webprogrammeringsspråk.

Ett bra webprogrammeringsspråk bör alltså innehålla följande egenskaper:

4.3 Läsbarhet och skrivbarhet

Ett av de viktigaste kriterierna för att bedöma programmeringsspråk är hur lätt koden kan läsas och förstås. (Sebesta, 1999) Detta beror på att underhållning är en viktig del av applikationernas livscykel. Eftersom applikationens underhållbarhet är beroende av kodens läsbarhet, är programmeringsspråkets läsbarhet mycket viktigt. Med skrivbarhet menas hur lätt det är att skriva program med språket. Skrivbarhet har många likadana kriterier som läsbarhet.

4.3.1 Enkelt och precist

Mindre är bättre. Desto färre komponenter och sätt att göra saker desto snabbare och lättare är det att lära sig använda språket effektivt. Språket ska vara uttrycksfullt (Expressiv). Det är svårt att mäta, men det innebär att språket bör ha ett lämpligt sätt att specificera beräkningar med språket. Språkets syntax bör likna mänskospråk, till exempel engelska så mycket som möjligt. Symboler, förkortningar och programmeringsjargong bör undvikas. Meningen (Semantik) bör följa direkt formen (Syntax). Egenskaper som överlagring av operatorer borde inte vara möjligt, då det leder till kod som är svår att förstå, om man själv inte skrivit den. Språket ska ha ortogonalitet (Orthogonality). Desto mera ortogonalitet, desto färre undantag. Man måste kunna definiera datatyper och identifierare får inte ha onödiga begränsningar. Speciella ord måste kunna skiljas från varandra och ska inte kunna användas som variabler. Språket ska ha stöd för både process abstraktion och data abstraktion.

4.4 Pålitlighet

Ett program är pålitligt om den genomför enligt sina specifikationer, under alla omständigheter. (Sebesta, 1999) Programmeringsspråket ska stöda utveckling av koder av högkvalitet och pålitlighet. Bra testverktyg hjälper utvecklaren att skriva högklassig kod.

4.4.1 Typkontroll

Typkontroll (Type checking) är granskning av skrivfel i koden. Processen varvid typerna kontrolleras, kan äga rum under kompilering (statisk kontroll) eller exekvering

(dynamisk kontroll). Då kontrollprocessen är kostligt (processorn och minne), ju tidigare fel i koden hittas desto mindre kostar (pengar) det att korrigera dem. Där för anses type checking under kompilering vara bättre. Ett språk med typkontroll ger pålitligare program. (Sebesta, 1999)

4.4.2 Undantagshantering

Med undantaghantering (Exception handling) menas programmets förmåga att hantera feltillstånd. Det är ett praktiskt sätt att rapportera och göra ändringar i programflödet, när en procedur inte kan utföras på normalt sätt. Undantagshantering är en viktig del av pålitlighet och många programspråk har därför inbyggt stöd för undantagshantering.

4.4.3 Läsbarhet och skrivbarhet

Både läsbarhet och skrivbarhet påverkar pålitligheten. Ett program som är skrivet med ett läsbart och skrivbart program är enklare och mer sannolikt pålitligt. (Sebesta, 1999)

4.5 Säkerhet

Man måste kunna övervaka en webapplikations tillgång till filer, lösenord och anslutningar. Språket måste erbjuda säkerhet för innehåll och kod. Språket måste stöda säkerhetsstandarder och ha inbyggda funktioner för hantering av och skydd mot vanligaste attackerna.

4.6 Kostnad

Kostnader (Cost) betyder här resurser, som förbrukas. Det kan vara tid, pengar eller dataresurser.

4.6.1 Utvecklings kostnader

Kostnaderna beror i stor del på hur lätt och snabbt det är att skriva program. Utvecklingsverktyg har också stor roll. Språket måste stöda snabb, enkel och billig webapplikationsutveckling. Språket måste vara tillgängligt och lätt att plocka upp och använda för olika uppgifter. Språket ska alltså ha bra, men billiga, helst gratis, verktyg och helst vara av högnivåspråk, då de har klart lägre skrivkostnader än lågnivåspråk.

Bra verktyg och språkets goda skrivbarhet resulterar också i lägre kostnader i programmerarskolning, för att kostnaden för utveckling är i förhållande till kostnaden för utvecklare. Ett språk som är lätt att lära sig har också lägre utvecklingskostnader.

4.6.2 Kostnader för att exekvera webapplikationen

Språket måste kunna utföra beräkningar snabbt och effektivt. Kompileringen eller tolkningen av språket får inte kräva för mycket resurser. Språket bör ha inbyggda funktioner för att hantera hårt trafikerade webapplikationer. Språket måste möta webapplikationernas prestandakrav.

4.6.2.1 Serversidans språk

När beräkningarna körs på servern, är det lättare att bedöma hur lång tid vissa uppgifter tar. Man behöver heller inte ta lika mycket hänsyn till användarens terminalens prestanda. Två saker kan påverka svarstiden som användaren upplever: För det första applikationer, som är interaktiva använder också mycket nätkapacitet, vilket ökar betydelsen på nätanslutningens hastighet. Ju långsammare nätanslutning desto längre svarstid har applikationen. För det andra när största delen av beräkningarna körs på serversidan belastar det serverns kapacitet och betydelsen på samtidiga användare ökar. Ju fler användare desto längre svarstid har applikationen.

4.6.2.2 Klientsidans språk

Då största delen av beräkningarna körs på klientsidan, som med Rika Internet Applikationer (RIA), måste man ta i beaktande vad man skickar till klienten och vad som körs. För båda frågorna finns det tre alternativ: källkod (source code), delvis kompilerad kod (byte-kod) och binärkod (binary code). (Ford, Wells, & Wells, 1998) Eftersom kompileringen sker på klientsidan, är det inte säkert att det som skickades, är samma sak som det som körs. Det är omöjligt för utvecklaren att vara säker på att beräkningarna, som utförs på klientsidan är korrekta, och detta kan leda till allvarliga säkerhetsproblem. Kompilatorn kan även ha kompatibilitetsproblem för vissa plattformar och terminaler. Terminalen måste också uppdateras kontinuerligt.

4.6.2.3 Serversidan eller klientsidan?

Vad som anses vara det bättre sättet att köra webapplikationer har växlat mellan serversidan och klientsidan genom hela webbens historia. Idag, på grund av snabba nätförbindelser och billiga server-resurser anses säkerhet och portabilitet ge serversidan klar fördel för vissa typer av webapplikationer, till exempel bank-applikationer. Klientsidans RIA och AJAX tekniken passar bättre applikationer som behöver en snabb interaktion med grafiska element, till exempel spel. (Wikipedia, 2011)

Enligt nuvarande tendens kan man allmänt säga, att om webapplikationen kan implementeras på serversidan, så lönar det sig att göra så. RIA och AJAX bör användas först när serversidans lösningar inte längre räcker. (Kyrnin, 2011) (Best Ajax Scripts, 2007) (Aggarwa, 2009)

4.6.3 Kostnader för implementation av webapplikationen

Språket måste möta webapplikationskrav på att vara enkelt och billigt att implementera. Webserverna måste stöda språket och det måste gå relativt lätt att installera applikationer på servern. (Sebesta, 1999)

4.6.4 Kostnader av underhållbarhet

Webapplikationen skall kunna underhållas och uppdateras hela sin livscykel. Detta gäller också programmeringsspråket, men samtidigt måste språket vara bakåtkompatibelt. Språket måste vara läsbart så att underhållningen och utvecklingen kan fortsätta även om utvecklarna ersätts. (Sebesta, 1999) Det måste också ha omfattande utvecklarstöd och dokumentation.

4.7 Portabilitet

Med tanke på mångfalden av operativsystem och hårdvaruplattformar som används på webben, måste dagens webprogrammeringsspråk stöda skalbarhet och portabilitet. (Sebesta, 1999) Det är inte nog att man kan utveckla applikationer med språket, de måste också kunna överföras till andra serverplattformar och applikationerna måste vara användbara med flera olika terminaler. Ju bättre olika plattformar, språk och terminaler är förenliga, desto populärare blir språket. Desto populärare ett språk blir, desto större

antal webhotell stöder det. Språket ska passa både stora projekt, som e-handelsplatser eller CRM-lösningar och små projekt, som en blogg.

4.8 Sammanfattning

HTML har visat sig vara otillräckligt för utveckling av moderna webapplikationer. Websidornas funktionalitet har förbättrats och förstärkts med hjälp programmeringsspråk, som används tillsammans med HTML. Dessa språk har möjliggjort den moderna dynamiska och interaktiva webben.

Nya programmeringsspråk är utvecklade för att möta de förändrade kraven. De skapas ofta genom att vidare utveckla existerande språk för att förbättra vissa områden som upplevs viktiga i webprogrammering. Eftersom det inte är troligt att någon av dessa konkurrerande språk skulle vinna och ensamt bli det enda rätta, kommer nya webprogrammeringsspråk fortsättningsvis dyka upp. Vid val av språk måste man ta hänsyn till egenskaperna hos språket och kraven på önskade webtjänsten. Det finns inte programmeringsspråk som utför alla uppgifter bättre än sina konkurrenter och fyller alla webprogrammerarens krav. Där för måste man ofta göra kompromisser och betona vissa egenskaper på bekostnad av andra.

5 Python allmänt

Python är ett modernt, objektorienterat programmeringsspråk, med stöd för flera olika programmeringsparadigm. Det är designat för att vara tydligt och kraftfullt. Python utformades av Guido van Rossum i slutet av 1980-talet. Namnet Python kommer från brittiska komediserien Monty Pythons flygande cirkus (Monty Python's Flying Circus). (Swaroop, 2008) Program skrivna i Python ska kunna vara lätta att skriva och läsa, vilket gör det lättare att snabbt utveckla program som fungerar. Python är idealiskt för prototyputveckling och andra ad hoc-programmeringsuppgifter, men det lämpar sig även för mer allmänna tillämpningar. (Python Wiki, 2008)

Python är ett versionsberoende programmeringsspråk. Python 3.0 är den nyaste versionen av språket. Skillnaderna mellan version 2.6 och 3.0 är relativt stora. Den främsta orsaken till en ny stor version av Python är att eliminera alla små problem och

konstigheter, som har samlats under årens lopp och att göra språket ännu renare. Python 2.x kod kan konverteras till 3.x med hjälp av ett verktyg. (Python documentation, 2009)

5.1 Egenskaper och särdrag

En av van Rossums mål när han utformade Python var att ha ett språk vars kunnande skulle leda till att behärska programmering allmänt. (Swaroop, 2008) Python har alltså flera egenskaper som gör det ett utmärkt ”nybörjarspråk” och således är också lämpligt för undervisande av programmering. År 2007 bytte Institutionen för informationsteknologi vid Åbo Akademi från Java till Python, som första programmeringsspråk, som undervisas. (Practical Programming, 2011)

5.2 Syntax och Semantik

Python-språkets syntax kallas "elegant" och ”rent”. (Python Wiki, 2008) Python syftar att vara ett mycket läsbart språk. Den är designad att ha en ren visuell uppställning, ofta använda engelska ord, där andra språk använder skiljetecken. Jämfört med språk som Java eller C++ har Python en mer intuitiv syntax.

Python påtvingar också ett indenterat och strukturerat sätt att skriva program på och koden liknar därför mycket en pseudo-kod. Python använder indentering med blank tecken, istället för klamrar eller nyckelord som begin eller end, för att avgränsa kodblock. En djupare indentering visar var kodblocket börjar, och en minskning innebär slutet på kodblocket. Många utvecklare tycker inte om Pythons sätt att använda indentering för att märka var kodblock. (Pilgrim, 2004) Då dessa osynliga markeringar ges en stor roll, kräver det ett noggrant och vant programmeraröga att hitta intendentfel.

Python möjliggör bättre återanvändning av kod än traditionella strukturerade språk, och har ett mindre antal syntaktiska undantag och särskilda fall än andra språk. Python är skiftlägeskänsligt. Versaler och gemener åtskiljs, det vill säga variablerna ”ORD”, ”Ord” och ”ord” är alla olika och kan ha olika värden.

5.3 Typsystem

Ett typsystem definierar hur ett programmeringsspråk klassifierar värden och uttryck som typer. Python använder ”Anka-typning” (Duck-typing), där objekten är typade, men variabelnamnen är typlösa. Anka-typning är en typ av dynamisk typning (dynamic typing), där ett objekts aktuella uppsättning av metoder och egenskaper bestämmer den giltiga semantiken istället för att ärva från en viss klass. Python låter programmerare definiera sina egna typer med hjälp av klasser. Dynamisk typning betyder också att största delen av typgranskning sker under exekvering. Python är också starkt typad (strong typing), vilket innebär, att i stället för att försöka exekvera kod som inte är väldefinierade, märker Pythontolken felen och meddelar om dem. Vidare förenklar dynamisk typning syntaxen när en variabel kan användas till flera syften, men missförstånd mellan utvecklare kan också lätt uppstå när det finns många sätt att göra saker. (Wikipedia, 2011)

5.4 Uttryck

Pythons uttryck liknar språk som C och Java, men Python skiljer sig från andra språk i och med att det inte har en inbyggd syntax för reguljära uttryck, utan erbjuder istället modulen RE. Skillnad mellan uttryck och uttalanden är strikt tillämpad, vilket kan leda till överlappande funktioner.

5.5 Metoder

Python metoderna har en self-parameter man kan fritt namnge för att förfoga över data, i motsats till implicit-self -metoden vissa andra objektorienterade programspråk har. (Python documentation, 2011)

5.6 Fri och öppen källkod

Python är ett exempel på en FLOSS (Free/Libre och Open Source Software). Det möjliggör att man kan läsa källkoden, göra ändringar i den och fritt distribuera exemplar av den. Tröskeln för att börja med Python är låg när man inte behöver köpa dyra utvecklingsverktyg och man har en stor utvecklargemenskap (community) redo att hjälpa genom att dela råd och egna lösningar. (Wikipedia, 2011) Det stora

utvecklargemenskapet har också kontribuera till en omfattande dokumentation, som är tillgänglig på nätet. Den kommer med en texteditor (IDLE) och det finns en stor mängd av tutorialer, böcker, kursmaterial och övningar.

5.7 Högnivåspråk

Högnivåspråk är konstruerade att så långt som möjligt motsvara det sätt på vilket utvecklaren ser på problem som skall lösas, inte på det sätt datorns maskinvara skall hantera problemet. Jämfört med lågnivåspråk är det mycket lättare att programmera i ett högnivåspråk. Program skrivna i högnivåspråk blir kortare och tar mindre tid att skriva. Termerna ”högnivå” och ”lågnivå” är relativa till vad det jämförds med. Pythons abstraktionsnivå anses vara högre än till exempel C-språkets, som i sin tur är högnivå jämfört med Assembler-språket.

Vid Pythons abstraktionsnivå är processor- och minnesdetaljer gömda från programmeraren och detta förhindrar buggar, som annars är vanliga vid utvecklingen stora system. Detta bidrar till en snabbare utveckling då testningen inte tar lika lång tid. Det gör det är också lättare att skriva stora, avancerade program, när mängden fel minskar och felsökning och förbättring av programmen underlättas.

Högnivåspråk är portabla, vilket innebär att man inte behöver bry sig om lågnivådetaljer och samma skript kan köras på många olika datorer och operativsystem med få eller inga förändringar. (Swaroop, 2008) Pythonprogram kan köras på diverse datorer och operativsystem utan att det kräver några ändringar i systemet. På grund av att Python är FLOSS har den faktiskt portats till många olika plattformar. Python kan köras på allt från Windows till PlayStation. (Swaroop, 2008)

Det är också den största nackdelen med högnivåspråk av Pythons nivå. Det är alltså svårare att exakt kontrollera hur programmet uppföra sig. Program skrivna i högnivåspråk är därför inte kanske lika snabba, kompakta eller resurssnåla, som program skrivna i lägrenivå, till exempel C.

5.8 Tolkat språk

Python är ett tolkat språk i motsats till kompilerat språk. I ett tolkat programmeringsspråk, eller skriptspråk som det också kallas, tolkas koden samtidigt som programmet körs, till skillnad för ett kompilerat språk, där koden översätts i en separat process före exekveringen. Det innebär också att det krävs en skild exekveringsmiljö för att kunna köra program skrivna i skriptspråk.

Python har implementerats med både kompilatorer och tolkar. Det huvudsakliga sättet som Python använder i dag är en mellanliggande teknik, som kombinerar kompilering och tolkning. Allmänt räknas Python dock som ett tolkat språk. Under utvecklingsarbetet, ger Python-tolken direkt begripliga kommentarer av eventuella fel, programmeraren kan göra prototyper av funktioner och ändra på små saker snabbare, utan att behöva kompilera programmet på nytt efter varje ändring, vilket kan ta en avsevärd tid. Allt detta gör Python till ett enklare och snabbare språk att utveckla program med.

Den största nackdelen med tolkningen är dock mycket sämre prestanda av programmet jämfört med kompilerat språk. Tolkning kräver tid och minne, som färdigt kompilerat språk inte behöver. En annan nackdel är att varje skriptspråk behöver en egen exekveringsmiljö, oftast för flera operativsystem. Ett program som ska säljas och spridas till många användare bör kunna köras som ett vanligt exekverbart program utan någon särskild exekveringsmiljö. Linux kommer färdigt inställd med Python, men om man vill att andra kan använda Python-applikationen, bör de installera Python-tolken först.

5.9 Python stöder många olika programmeringsstilar

Python är ett dynamiskt språk och erbjuder funktioner, som är gemensamma för största delen av dynamiska språk. Det tvingar inte en utvecklare till ett visst paradigm. Python stödjer objektorienterad och imperativ programmering. Det finns också ett begränsat stöd för funktionell programmering.

I objektorienteradespråk, är programmet uppbyggt kring en varierande uppsättning objekt som interagerar med varandra och kombinerar data och funktioner. I

imperativaspråk, är programmet uppbyggt kring subrutiner. I stället för klasser används funktioner, strukturer och primitiva datatyper. Funktionell programmering använder matematiska funktioner för att konstruera program. Funktionerna utvärderas vid exekvering av programmet.

Python har ett mycket kraftfullt, men förenklat sätt att stöda objektorienterad programmering och imperativt programmering. Python främjar inte funktionell programmering även om den har alla funktioner som krävs för att göra det möjligt att programmera i funktionell stil i Python.

5.10 Utvidgbar och Inbäddbar

Pythons förmåga att utföra extra kod på platser som bestäms vid tolkning av koden, gör det möjligt för utvecklaren att skriva kod som är helt oberoende av de underliggande implementationerna. (Python documentation, 2011) Python kan alltså lätt utvidgas till exempel genom att lägga till nya moduler som är implementerade i andra språk. Detta gör Python inte bara mycket lätt att utvidga men också att upprätthålla och stöda.

Det är också möjligt att göra det tvärt om: att utvidga ett program skrivet i ett annat språk genom att inbädda Python i det. Inbäddande liknar utvidgandet, men är inte riktigt samma sak. Skillnaden är att när du utvidgar Python, är huvudprogrammet fortfarande Pythontolken, medan om du inbäddar Python, kallar vissa delar av programmet istället på Pythontolken föra att köra Pythonkod. (Python documentation, 2011) För att stöda utvidgbarhet och inbäddbarhet är all funktionalitet, utom grundläggande funktioner och datatyper, implementerad i form av ett rikt standardbibliotek.

5.11 Omfattande bibliotek

Python standardbibliotek är mycket omfattande och stöder många vanliga programmeringsuppgifter såsom reguljära uttryck, dokumentation, enhetstestning (Unit testing), utnyttjande av trådar (threading), databaser, webläsare, CGI, FTP, e-post, XML, XML-RPC, HTML, WAV-filer, kryptografi, GUI (grafiska användargränssnitt) och innehåller flera olika typer av komponenter och datatyper som normalt skulle betraktas som del av "kärnan", såsom siffror och listor. (Python documentation, 2011)

Allt detta finns tillgängligt överallt där Python är installerat. Förutom standardbibliotek, finns det många andra bibliotek som gör utvecklarens arbete lättare.

5.12 Sammanfattning

Python är inte det perfekta språket för varje programmeringsuppgift. Språket betonar utvecklings produktivitet och en kodläsbarhet över prestanda. Det anpassar inte till program som är gjorda för systemnivå, till exempel drivrutiner och kärnor, eftersom den är på högnivå för att ge bra kontroll över minnesallokering och andra uppgifter på lågnivå. Eftersom Python är relativt långsammare än kompilerade språk, är den inte vällämpad för beräkningsintensiva program. Ändå har Python börjat växa kraftigt i popularitet de senaste åren, och en del av orsaken är att språket är mycket flexibelt, men också kraftfullt.

6 Python och webben

Som redan tidigare konstaterades är Python lämpligt för ett brett spektrum av användningsändamål på grund av dess mångsidighet. Jag har redan gått igenom Pythons styrkor och svagheter, och de gäller även för webutveckling. I detta avseende skiljer sig Python inte från andra dynamiska scriptspråk, som dominerar webprogrammering, men Python är fortfarande en utmanare inom webprogrammeringsspråk och ligger efter de populäraste språken, såsom Java och diverse C-språk.

Jag undersöker nu i detalj Pythons lämplighet för olika aktiviteter inom den klassiska webserverarkitekturen. Språkets styrkor och svagheter, exempelvis i kommunikation mellan klienten och servern, presenteras och jag berättar ur Python sköter databas-integrering och URL-hantering. Det mest populära sättet att utveckla webapplikationer idag är med hjälp av ramverk. Deras lämplighet för uppgiften har stor inverkan på valet av programmeringsspråk. Jag undersöker vilka lösningar de kan erbjuda och presenterar Pythons populäraste webramverk.

6.1 TCP/IP -protokoll

Python erbjuder en mängd olika TCP/IP-nätverkprotokoll, bland annat följande moduler: httplib (HTTP) ftplib (FTP), soappy (SOAP), imaplib (IMAP), poplib (POP), smtplib (SMTP), nntplib (NNTP) gopherlib (Gopher) telnetlib (Telnet) samt urllib och urllib2, som använder flera protokoll, däribland http och FTP, som webservrar vanligtvis erbjuder.

6.2 Sockets

HTTP-protokollet definierar dataöverföringen mellan två parter: klienten och servern. Parterna skapar en TCP/IP-anslutning (socket). Klienten skickar en förfrågan till servern och servern svarar genom att skicka ett lämpligt svar. Klienten hanterar serverns svar, till exempel genom att skapa en websida. När sidan är klar, är interaktionen avslutade och anslutningen kan förkastas. Användning av sockets i Python är enkelt. Som med det mesta i Python, finns det ett bibliotek, som innehåller alla verktyg, som behövs för att arbeta med sockets. Python har stöd för både socket stream och socket datagram.

6.3 Server-applikation gränssnittet

Det finns egna protokoll för hur en webserver kör webapplikationer eller webramverk med argument och hur dessa anropas från en webläsare via http. Python-språket stöder flera av dessa protokoll.

6.3.1 CGI-protokoll

Common Gateway Interface (CGI) är ett protokoll, som stöds av all webservrar. CGI startar en ny process varje gång webapplikationen anropas. Detta kan fungera tillräckligt bra i situationer där webläsaren sällan kommer att anropa servern. Om applikation har en hel del trafik, kommer CGI att bli för långsam. Effektivare protokoll såsom FastCGI och SimpleCGI har utvecklats för att förbättra prestandan av CGI. Också de här protokollen är brett stödda. Till exempel Apache och den senaste IIS-servrar stöder CGI, FastCGI och SCGI. Idag anses CGI-protokollen vara något av en katastrof. Att köra en ny process från början är en av de dyraste enstaka åtgärder man kan utföra med

ett modernt operativsystem och kräva att det sker för varje enskild inkommande HTTP-anrop är helt enkelt fel. (Rhodes & Goerzen, 2010)

6.3.2 Mod_python

När det blev klart att CGI var både ineffektivt och oflexibelt, valdes man att använda en speciell modul som bäddar in Python tolken direkt i en webserver. (Rhodes & Goerzen, 2010) Mod_python är en sådan modul. Den integrerar Python-språket i Apache-servern. Inbäddade Python-tolken genomför Python-applikationer som en del av Apache HTTP-svar på anrop. Mod_python stoppar inte processen efter varje anrop såsom CGI, så till exempel databaskopplingarna lämnas öppna, vilket i sin tur gör datahämtning snabbare. Eftersom Mod_python modulen är integrerad med själva webservern, kan den direkt hantera många webserverfunktioner, till exempel genomföra olika TCP/IP -protokoll, filtrera anrop och svar och bestämma dokumenttyp. Mod_python kan också användas för Python Server Pages (PSP), en teknik där Python-kod kan bäddas in i HTML-sidor på samma sätt som i ASP, PHP och JavaServer Pages (JSP). Mod_python är alltså snabbare och säkrare i förhållande till CGI-protokollen och länge var det överlägset det populäraste sättet att ansluta Python till World Wide Web.

Att inbädda Python-tolken direkt i en Apache-process, kan orsaka problem. Speciellt om webservern är delad mellan flera webapplikationer. Att alla applikationer körs under samma användarnamn, orsakar sina egna säkerhetsproblem. Hela webservern måste till exempel startas igen om ett av applikationerna kraschar, uppdateras eller om det finns minnesläcka. Det är också långsammare och svårare att installera en Mod_python gränssnitt än ett gränssnitt som baserar sig på CGI-protokollet. Många av de mindre erfarna utvecklarna föredrog därför att välja ett CGI-gränssnitt. Alla webserverfunktioner, som man vill hantera, blev implementerade med ramverk istället. (Rhodes & Goerzen, 2010)

Idag är användningen av Mod_python begränsad. Den underhålls av Apache och dess vidare utveckling har upphört, men eftersom den fortfarande ger Python ett unikt gränssnitt till Apache-servern är den fortfarande ett lönsamt alternativ vad det angår Apache-server. (Rhodes & Goerzen, 2010)

6.3.3 WSGI

Integrering Python i webservrar med olika CGI-protokoll och Mod_python brukade vara vilt. Varje server stödde olika protokoll och använde olika anrophändelser. Webapplikationer måste skrivas speciellt för en server och skulle portas innan de kunde användas på en annan webserver. Webramverk hade ett separat gränssnitt för varje server vilket utvecklare kanske ville använda för att köra sina applikationer. Python fick ryktet som ett språk vars applikationer var svårt att installera.

Denna situation förbättrades mycket genom införandet av Python Web Server Gateway Interface (WSGI). WSGI är ett CGI-baserad standard-protokoll. Python-språkets mod_wsgi modul kombinerar CGI-protokollens och mod_pythons bäst egenskaper. WSGI införde ett enda anropssätt för alla webserver att implementera, vilket gör servern kompatibel med alla Python webapplikationer och webramverk, som också stöder WSGI.

Mod_wsgi modulens installation är enkel och snabb. Den är implementerad i C språk, och använder ingen Python-modul, så den tar upp mindre minne och exekverar mycket snabbare än Mod_python. Mod_wsgi kan också starta applikationer på nytt utan att man behöver starta om servern.

Mod_wsgi kan köras på två sätt, endera inbäddad i servern (embedded), likt mod_python, som kör programmet på servern direkt eller som en skild process (daemon), likt mod_fastcgi, där servern kommunicerar med en separat process. Med ISAPI-WSGI, ett ISAPI-tillägg ger WSGI mod_python egenskaper också för en IIS-server. Med WSGI behöver man alltså inte mera bestämma sig för en särskild webserver innan man börjar koda utan applikationerna är helt portabla mellan olika servrar.

Trots detta är det ännu inte väldigt många webhotell, som stöder mod_wsgi och FastCGI är ofta det enda alternativet om du inte har tillgång till webservern själv. WSGI-protokollet är inte färdigt utan vidareutvecklas och förbättras ständigt. WSGI anses vara en viktig standard i framtiden och WSGI stödets långsamma, men stadiga ökning på antalet servrar, påverkar starkt Python-språkets popularitet som webprogrammeringsspråk.

6.4 Python webserver

Python erbjuder tre moduler, BaseHTTPServer, SimpleHTTPServer och CGIHTTPServer, med vilka man kan bygga sin egen webserver. Dessa små Python webservrar gör det lättare att köra små applikationer, för vilka det inte lönar sig att installera Python på en webserver.

6.5 Webapplikationramverk

Jag har fokuserat mycket på nätprogrammering, men utveckling av webapplikationer i dag handlar mest om att hitta det rätta webapplikationramverket, som sköter nätinställningarna för oss. Ett ramverk är en samling av generella lösningar på olika uppgifter, som webapplikationer ska utföra. Det är alltså en mängd av genvägar och beprövade lösningar, som underlättar webutvecklarens arbete.

Jämfört med andra språk har Python avsevärt många ramverk. Eftersom Python är ett kraftfullt och flexibelt språk, är det lätt att skriva nya webramverk för det och många utvecklare har gjort just det. Det kan till och med vara möjligt att ramverk föds i misstag, när en liten webapplikation gradvis med tiden växer för att stöda en massa olika uppgifter. (Rhodes & Goerzen, 2010) Detta är både Python styrka och svaghet vad det gäller webutveckling.

Python-världen är full av ramverk, mikroramverk, metaramverk och liknande. Det finns flera alternativ för alla behov och de är ofta väldigt användbara verktyg, men ofta är alltför många alternativ en negativ sak. Ramverk är avsett att standardisera vissa aspekter av programmering, om det finns alltför många alternativ arbetar det mot detta mål. Python har en lång tid lidit av att samfundets utvecklings-resurser har varit uppdelade i många olika ramverk-projekt. Andra språk har däremot inriktat sig på ett par olika alternativ. Under de senaste åren, har Python-ramverk dock utvecklats med stormsteg och Python har att erbjuda några mycket populära och konkurrenskraftiga ramverk.

Ovan nämnda protokoll och moduler för server-applikation-gränssnittet kräver en hel del kunskap om webservrar och nätverksprotokoll. Majoriteten av webutvecklare idag vill börja så snabbt som möjligt att utveckla applikationer. De undviker vanligtvis att

skriva bara WSGI-applikationer och använder istället ramverk, som ger verktyg för hantering av inställningar, protokoll och gränssnitt.

HTTP-protokollen och process-hantering är uppgifter som till och med de allra lättaste och enklaste Python-ramverk klarar av. Python-ramverk har en samling av paket och moduler som gör det enkelt och snabbt att installera sina applikationer på webservern. Med hjälp av ramverk säkerställer man också att komplexa WSGI- inställningar går rätt. Ramverk kan även innehålla egna oberoende HTTP-servrar. Allt detta frigör utvecklaren att koncentrera sig på högnivå-programmering. Ett fullständigt Python-ramverk har många färdiga lösningar man kan använda för att implementera sin webapplikation. (Python Wiki, 2011)

Nedan beskrivna några av de mest använda lösningarna.

6.5.1 Databashantering

Vissa ramverk har ett databasgränssnitt, som gör det enkelt hantera databaser. För varje tabellrad definieras ett modell objekt, vilket utvecklaren kan använda för att definiera datafält, söka, skapa eller redigera objekt.

6.5.2 Webtjänster

Webtjänster kan allmänt betraktas som en interaktion mellan webapplikationer på webben med hjälp av standardiserade meddelandeformat. De flesta stora webapplikationer innehåller många dynamiska element, som gör detta. Python har moduler som stöder flera webserviceprotokoll: XML-RPC, JSON-RPC, SOAP, XMPP. Många ramverk implementerar dessa.

6.5.3 Användarverifiering

Användarverifiering kan vara viktigt för vissa webapplikationer. Några webramverk kan komma ihåg användarnamn och lösenord och har stöd för inloggning och sessionscookies, medan andra fordrar att utvecklaren själv programmerar dessa komponenter.

6.5.4 Model-View-Controller

Model-View-Controller (MVC) är ett designmönster, där data- och affärslogiken (Model) samt presentation och användarinteraktion (View) är avskilda med en mellanliggande komponent. (Controller). Nästan alla Python ramverk stöder detta då det är lätt att underhålla. Användning av vyer (view) och modeller (model), skiljer sig Python starkt från till exempel PHP-språket som fritt blandar HTML och PHP-kod för att producera blandning av koder. Med de flesta Python ramverken kan utvecklaren bestämma hur strängt han följer MVC till exempel med HTML validering.

6.5.5 URL-hantering

Diverse ramverk hanterar URL-adresser på helt olika sätt. Några mindre ramverk låter utvecklaren skapa små applikationer genom att dekorera anropade objekt med URL-mönster. Andra ramverk anser att varje applikation ska definiera sina webadresser på ett ställe, så att koden är i två nivåer. Sändning av URL:en händer på ett ställe och hanteringen på en annan. Ett annat sätt är att man definierar kontrollerklasser, som representerar någon punkt i URL hierarkin och sedan skriver metoder för dessa kontrollerklasser. Istället för att behöva matcha olika mönster eller använda kontroller är varje URL en väg från ett objekt till ett annat och varje URL-komponent pekar på följande objekt, som ska hanteras. Olika URL hanteringsmekanismer producerar samma resultat och vilken man väljer är en smaksak.

6.5.6 Några Python webramverk

Ramverkutbudet lever hela tiden och förändras mycket snabbt. Gamla ramverk vissnar bort och nya växer i stället för dem. De starkaste ramverken har stödet av en stor gemenskap, som ständigt korrigerar dess brister och utvecklar dem i den riktning den tycker är viktigaste. Detta innebär att stora ramverk kan skilja sig betydligt från varandra.

- ***CherryPy*** är ett enkelt ramverk som erbjuder basapplikationsserver, antingen som en egen oberoende webserver eller som en process via WSGI på en webserver. (CherryPy.org, 2011)

- ***Django*** hör till de populäraste kompletta ramverken. Det är ett högnivåramverk,

vars syfte är att göra det enkelt att utveckla avancerade databas-drivna webapplikationer. Det ska vara lätt att återanvända komponenter och det ska gå snabbt att skriva kod. Django genererar automatiskt administrations-gränssnitt utifrån data-modellen som anges i Python-kod. (Djangoproject.org, 2011)

- ***Zope*** är av Python ramverken en av de mognaste. Zope har vuxit till en familje av ramverk som främst används av innehållshanteringssystem (Content Management Systems) så-som Plone, Silva och ZMS. Zope Omfattar även ZODB, en objektorienterad databas och Medusa-http, en egen oberoende webserver. (Zope.org, 2010)

Största delen av Python-webramverken stöder enbart programmering på serversidan, men med genombrottet av AJAX-teknologin, har några ramverk börjat omfatta också AJAX-kod, vilket hjälper utvecklaren att programmera också på klientsidan. (Python Wiki, 2011)

Det urval av mångsidiga ramverk av högkvalitet, som Python har idag, har hjälp Python att stiga från ett lillebrorstatus, till et av de främsta webprogrammeringsspråk, som används i dag. Oavsett vilket ramverk man än väljer, är det dock Python med dess flexibilitet och elegans som möjlig-gör programmering av goda webapplikationer. (Rhodes & Goerzen, 2010)

6.6 Sammanfattning

Python webapplikationer är oftast distribuerade med hjälp av antingen en ren Python webserver för mindre applikationer eller genom att använda en skild webserver för att anropa dynamiskt innehåll från webapplikationen. Orsaken till den stora förändringen i Python-språkets popularitet under de senaste åren beror på server- utvecklingen av applikation gränssnittet och Python webramverken. Python-applikationer använde tidigare CGI-protokollet eller mod_python modulen, vilket krävde komplicerade och arbetskrävande installationer, samtidigt som konkurrerande språk redan hade alla nödvändiga inställningar färdigt på servern. Införandet av WSGI, Python-språkets standardgränssnitt för webservrar och webapplikationer, har varit ett stort framsteg i Python-språkets lämplighet för webprogrammering. Python har krävt mycket extra jobb av webapplikationutvecklare och dess prestanda har varit inkonsekvent mellan de olika

protokollen, men mod_wsgi har till stor del förändrat detta. WSGI har möjliggjort programmering utan en förutbestämd serverplattform och färdiga program kan lätt överföras från en serverplattform till en annan. Webservrar och webapplikationer kan nu kommunicera fritt utan att skilda moduler behövs programmeras för varje webserver man vill stödja.

Webramverk har en avgörande roll i modern webutveckling. De hanterar mycket av http logiken och de också ge flera viktiga abstraktioner: de kan sända webadresser till Python-kod, infoga Python variabler i HTML-modeller, hantera objekt i databasen och låta dem nås från webben genom protokoll som CRUD och REST. I dag väljer man oftast ett webramverk som passar projektet och låter dess bibliotek hantera de vanligaste uppgifter, men man kan också välja att göra allt själv utan hjälp av standard verktyg. Python webservrar kan vara mycket nyttiga i vissa fall. Det finns inte bara ett bra urval av färdiga webservrar, utan några små servrar är också inbyggda i Python biblioteket.

7 Sammanfattning

Python är ett mångsidigt programmeringsspråk, som passar utmärk för webutveckling. Många av Python-språkets grundegenskaper, som låga utvecklingskostnader och korta utvecklingstider, samt god skalbarhet, portabilitet och underhållbarhet möter de kraven, som ett bra webprogrammeringsspråk ställs.

Python gör det lättare att utveckla pålitliga och säkra webapplikationer med ett stort standardbibliotek, som stöder många säkerhet- och pålitlighetsfunktioner, till exempel enhetstestning och MVC-modellen.

Som med all tolkade språk Python-språkets största svaghet är i prestanda.

En prestanda, som räcker till flesta användningsändamål

Python har redan i flera år varit ett av de snabbast växande webprogrammeringsspråken.

WSGI har möjliggjort programmering utan en förutbestämd serverplattform och färdiga program kan lätt överföras från en serverplattform till en annan. Detta har hjälpt...

Ännu större påverkan på språkets stigande popularitet, har varit utvecklingen av dess webapplikationramverk. Python-språkets populäraste ramverk, som Django och Zope, hör till webbens mest omfattande och erbjuder en god samling av genvägar och beprövade lösningar på olika programmeringsuppgifter.

Jämfört med andra tolkade språk är Python också populär bland programmerare utanför webprogrammeringsområdet, vilket gör att bibliotek och begåvade programmerare är mera lättillgängliga.

Till exempel, på Google används Python i stor utsträckning och är enda programmeringsspråk på Google App Engine, som stöds.

8 Jämförelse

Ett flertal jämförelser har gjorts mellan de olika programmeringsspråken. Jämförelsen är ofta gjorda från en viss synvinkel och programmeringsspråkens lämplighet är övervägt endast inom ett särskilt användningsområde eller enskild uppgift. Dessa jämförelser är inte alltid så användbara, för att de inte avslöjar mycket om språkens allmänna styrkrelationer.

Jämförelser var alla möjliga användningsområden har tagits i beaktan, är naturligtvis mycket svåra att utföra. Men en bra tillgångs sätt är att se hur de möter kraven på bra webprogrammeringsspråk, som diskuterades om i kapitel 4.

8.1 Språken som jämförs

TIOBE Programming Index har använts för att välja några av de viktigaste programmeringsspråk, som används på webben. Indexen beräknar språkets popularitet, inte nyttighet eller passande för diverse användningsområden. Populariteten berättar om hur aktuell språket är och hur bra det svarar på utvecklarnas behov. Till exempel, C# var vald för jämförelsen på grund av att största delen av nya applikationer skrivna med Microsoft-verktyg är i C#. C++ blev inte vald för den skiljer sig inte tillräckligt från C# att det skulle löna sig ha båda språken med. C, även om den är ett av de mest använda språken på webben, har sitt eget speciellt användningsområde, som andra språken inte

tävlar i. För jämförelsen valdes system programmeringsspråken Java och C# samt skriptspråken Python och PHP.

8.2 Allmän jämförelse

Language	C#	Java	PHP	Python
Intended use	Application, Web	Application, Web	Web, Server-side	General, Application, Scripting, Web
Appeared in	2001	1995	1995	1991
Designed by	Microsoft	Sun Microsystems	Rasmus Lerdorf	Guido van Rossum
Developer	Microsoft	James Gosling & Oracle Corporation	The PHP Group	Python Software Foundation
License	Proprietary	OpenSource	OpenSource	OpenSource
Paradigms	imperative, functional, generic, reflective, structured, event-driven	imperative, generic, reflective, structured	imperative, procedural, reflective	imperative, functional, aspect-oriented, reflective
Object-Orientation	Hybrid	Hybrid	Hybrid	Hybrid
Garbage Collection	Mark and Sweep or Generational	Mark and Sweep or Generational	Reference Counting	Reference Counting
Type strength	strong	strong	weak	strong
Type safety	safe	safe		safe
Expression of types	explicit	explicit	implicit	implicit
Compatibility among composite types	name-based	name-based		property-based
Type checking	static	static	dynamic	dynamic
Standardized	2000, ECMA, ISO	De-facto standard through the Java Language Specification	No	No

9 Sammanfattning

Python är ett mångsidigt programmeringsspråk, som passar utmärk för webutveckling. Många av Python-språkets grundegenskaper, som låga utvecklingskostnader och korta utvecklingstider, samt god skalbarhet, portabilitet och underhållbarhet möter de kraven, som ett bra webprogrammeringsspråk ställs. Python gör det lättare att utveckla pålitliga och säkra webapplikationer med ett stort standardbibliotek, som stöder många säkerhet- och pålitlighetsfunktioner, till exempel enhetstestning och MVC-modellen. Som med all tolkade språk Python-språkets största svaghet är i prestanda, Pythons prestanda, räcker ändå till flesta användningsändamål. Python har redan i flera år varit ett av de snabbast växande webprogrammeringsspråken. WSGI har möjliggjort programmering utan en förutbestämd serverplattform och färdiga program kan lätt överföras från en serverplattform till en annan. Ännu större påverkan på språkets stigande popularitet, har varit utvecklingen av dess webapplikationramverk. Python-språkets populäraste ramverk, som Django och Zope, hör till webbens mest omfattande och erbjuder en god samling av genvägar och beprövade lösningar på olika programmeringsuppgifter. Jämfört med andra tolkade språk är Python också populär bland programmerare utanför webprogrammeringsområdet, vilket gör att bibliotek och begåvade programmerare är mera lättillgängliga. Till exempel, på Google används Python i stor utsträckning och är enda programmeringsspråk på Google App Engine, som stöds.

10 Referenser

Aggarwa, N. (den 27 1 2009). *Pagination: Server Side or Client Side?* Hämtat från JAVALOBBY: http://java.dzone.com/articles/pagination-server-side-or-clie den 1 4 2011

Best Ajax Scripts. (2007). *Why not AJAX.* Hämtat från Best Ajax Scripts: http://www.bestajaxscripts.com/why-not-ajax.html den 13 3 2011

CherryPy.org. (den 26 2 2011). *Welcome to cherrypy.org.* Hämtat från CherryPy.org: http://www.cherrypy.org/ den 3 3 2011

Deitel, H., Deitel, P., & Nieto, T. (2001). *Internet & World Wide Web: How to program.* New Jersey: Deitel & Associates Inc.

Djangoproject.org. (2011). *Meet Django.* Hämtat från Django: http://www.djangoproject.com/ den 3 3 2011

Ford, S., Wells, D., & Wells, N. (1998). *Web Programming Languages.* Hämtat från Object Services and Consulting, Inc.: http://www.objs.com/survey/lang.htm den 2 3 2011

Garrett, J. J. (den 18 2 2008). *Ajax: A New Approach to Web Applications*. Hämtat från Adaptive path: http://www.adaptivepath.com/ideas/e000385 den 2 4 2011

Graham, P. (9 2001). *The Other Road Ahead.* Hämtat från Paul Graham: http://www.paulgraham.com/road.html den 1 4 2011

Hand, M. (u.d.). *Application Extensibility in Python.* Hämtat från Python.org: http://www.python.org/workshops/1996-06/papers/m.hand-app-extensibility.html den 3 3 2011

Ho, V. (den 4 12 2008). *Gartner sees high levels of SaaS popularity.* Hämtat från ZDNet: http://www.zdnet.co.uk/news/it-strategy/2008/12/04/gartner-sees-high-levels-of-saas-popularity-39570765/ den 29 3 2011

Kappel, G., Pröll, B., Reich, S., & Retschitzegger, W. (2003). *Web Engineering.* Heidelberg: John Wiley & Sons Ltd.

Koskenniemi, H., Saastamoinen, M., & Eerola, P. (2007). *Kvalitetskriterier för webbtjänster.* Hämtat från www.finansministeriet.fi: www.finansministeriet.fi den 5 3 2011

Kyrnin, J. (2011). *When to Use Ajax and When Not To.* Hämtat från About.com: http://webdesign.about.com/od/ajax/a/aa092506.htm den 13 3 2011

Peltomäki, J. (1998). *WWW-ohjelmointi.* Jyväskylä: Teknolit Oy.

Pilgrim, M. (2004). *Dive Into Python.* Apress.

Practical Programming. (2011). *Practical Programming.* Hämtat från Improving Mathematics and Programming Education: http://www.imped.fi/wordpress/?page_id=11 den 3 3 2011

Python documentation. (den 14 2 2009). *Automated Python 2 to 3 code translation.* Hämtat från Python.org: http://docs.python.org/release/3.0.1/library/2to3.html den 13 3 2011

Python documentation. (den 3 4 2011). *Embedding Python in Another Application.* Hämtat från Python documentation: http://docs.python.org/extending/embedding.html den 3 3 2011

Python documentation. (den 3 4 2011). *Extending Python.* Hämtat från Python documentation: http://docs.python.org/extending/extending.html den 3 3 2011

Python documentation. (den 3 4 2011). *General Python FAQ.* Hämtat från Python documentation: http://docs.python.org/faq/general#why-must-self-be-used-explicitly-in-method- den 3 3 2011

Python documentation. (2011). *The Python Standard Library.* Hämtat från Python documentation: http://docs.python.org/library/ den 3 3 2011

Python Wiki. (den 15 11 2008). *Beginners Guide.* Hämtat från Python Wiki: http://wiki.python.org/moin/BeginnersGuide/Overview den 13 3 2011

Python Wiki. (den 28 3 2011). *Web Frameworks for Python.* Hämtat från Python Wiki: http://wiki.python.org/moin/WebFrameworks den 3 3 2011

Rhodes, B., & Goerzen, J. (2010). *Foundations of Python Network Programming (Second Edition).* New York: Apress.

Savoia, A. (2001). Measurement & Analysis Web Page Response Time 101. *STQE*, 48-53.

Sebesta, R. W. (1999). *Concepts of Programming Languages.* Colorado Springs: Addison Wesley Longman Inc.

Stamos, A., Thiel, D., & Osborne, J. (den 6 8 2008). *Living in the RIA World: Blurring the Line between Web and Desktop Security.* Hämtat från isecpartners.com: http://www.isecpartners.com/files/RIA_World_BH_2008.pdf den 30 3 2001

Swaroop, C. H. (den 8 10 2008). *A Byte of Python.* Hämtat från Swaroop C H: http://www.swaroopch.com/notes/Python den 31 3 2011

TechPluto. (den 10 6 2010). *Core Characteristics of Web 2.0 Services.* Hämtat från TechPluto: http://www.techpluto.com/web-20-services/ den 30 3 2011

Tiobe. (3 2011). *TIOBE Programming Community Index* . Hämtat från TIOBE Software: http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html den 10 3 2011

W3C. (2009). *W3C Help and FAQ.* Hämtat från The World Wide Web Consortium: http://www.w3.org/Help/ den 3 4 2011

Wikipedia. (den 31 3 2011). *Free and Open Source Software.* Hämtat från Wikipedia: http://en.wikipedia.org/wiki/Free_and_open_source_software den 3 3 2011

Wikipedia. (den 14 2 2011). *Pagination.* Hämtat från Wikipedia: http://en.wikipedia.org/wiki/Pagination_%28web%29 den 5 3 2001

Wikipedia. (den 3 4 2011). *Python (programming language).* Hämtat från Wikipedia: http://en.wikipedia.org/wiki/Python_(programming_language)#Typing den 3 3 2011

Zope.org. (2010). *Zope documentation.* Hämtat från Zope.org: http://www.zope.org/Documentation/ den 4 3 2011