



WSRP analyse

Analyse af integrationsmetoder ved brug af WSRP



IT- og Telestyrelsen

Ministeriet for Videnskab
Teknologi og Udvikling



WSRP analyse
Analyse af integrationsmetoder ved brug
af WSRP

Udarbejdet af IT- og Telestyrelsen i
samarbejde med EOS – Eastfork Object
Space ved Joakim Recht.

Udgivet af:

IT- & Telestyrelsen
Holsteinsgade 63
2100 København Ø
Telefon: 3545 0000
Fax: 3545 0010

Publikationen kan hentes
på It- & Telestyrelsens hjemmeside:

<http://www.itst.dk>

Forside foto: Thomas Yde

>

WSRP analyse

Analyse af integrationsmetoder ved brug af WSRP

Indhold

>

Problemstilling	6
Rapportens opbygning	7
Generelt om WSRP 1.0 og 2.0	8
Kort producentoversigt	11
Simpel WSRP	12
Opsummering	12
Sikkerhed i WSRP	14
Logning	15
Opsummering	15
Context Read og Write	17
Opsummering	17
Portlet Præferencer	19
Opsummering	19
HTML Forms	20
Opsummering	20
Javascript og AJAX	21
Opsummering	21
Fejlhåndtering	22
Opsummering	22
Notifikation og hændelser	23
Ændring af view	24
Opsummering	24
Skins og struktur	25
Opsummering	25
Styring af flowcontrol	26
Performance og skalerbarhed	27
Late binding	27
Parallel rendering	27
Caching	27
Skalering af producenter	27
Erfaringer	28
Interoperabilitet	28
Overordnet konklusion	29

Leverandøruafhængighed	29
Opsamling	29
Litteraturoversigt	31

Problemstilling

>

I forbindelse med udviklingen af offentlige portaler, er der behov for at klarlægge hvilke muligheder der er for at integrere forskellige myndigheders services i en samlet portal. Denne rapport fokuserer på Web Services for Remote Portlets-specifikationen (WSRP), der er en standard udgivet af OASIS.

En offentlig portal som borger.dk er karakteriseret ved, at den skal samle information for den enkelte borger, og borgeren skal kunne interagere med det offentlige, i princippet uden at forlade portalen. Det stiller en række krav til integration, da portalen i sig selv ikke stiller noget information til rådighed. I stedet skal informationerne hentes fra de enkelte myndigheder, og da borgeren ikke nødvendigvis er interesseret i det offentliges opbygning, skal portalen sørge for, at selvom information kommer fra forskellige myndigheder, skal det stadig fremstå som en helhed.

Rapportens opbygning

>

Denne rapport er en analyse af WSRP 1.0 og 2.0 og standardens muligheder for at integrere indhold fra forskellige myndigheder i en samlet portal. Rapporten tager udgangspunkt i en beskrivelse af forskellige problemstillinger [SAF], hvor hver af disse problemstillinger behandles i hver sit afsnit. Enkelte afsnit er slået sammen, da analysen har vist, at de er en del af samme problemstilling.

Fokus for analysen har ikke været en praktisk sammenligning af de forskellige produkter og teknologier, men i stedet en mere teoretisk gennemgang, baseret på de informationer, der er gjort tilgængelige på nettet samt indsamlede erfaringer.

For at vurdere WSRP, indeholder hvert afsnit en opsummering af fordele og ulemper ved hhv. brug af forskellige teknologier:

- > WSRP 1.0: Et website opbygges gennem en portal, som er stykket sammen af et antal komponenter (portlets). Portlets er implementeret uafhængigt af portalen, potentielt på en helt anden platform, og de enkelte portlets hentes via webservices over nettet.
- > WSRP 2.0: Samme princip som WSRP 1.0, men med flere muligheder.
- > IFrames: Browserbaseret teknologi, som gør det muligt at indlejre andre sider i en webside.
- > Normalt website: Websitet opbygges gennem et normalt framework, fx Struts, JSF eller lignende, og komponenter udefra vises ved at implementere en visning ovenpå webservices.

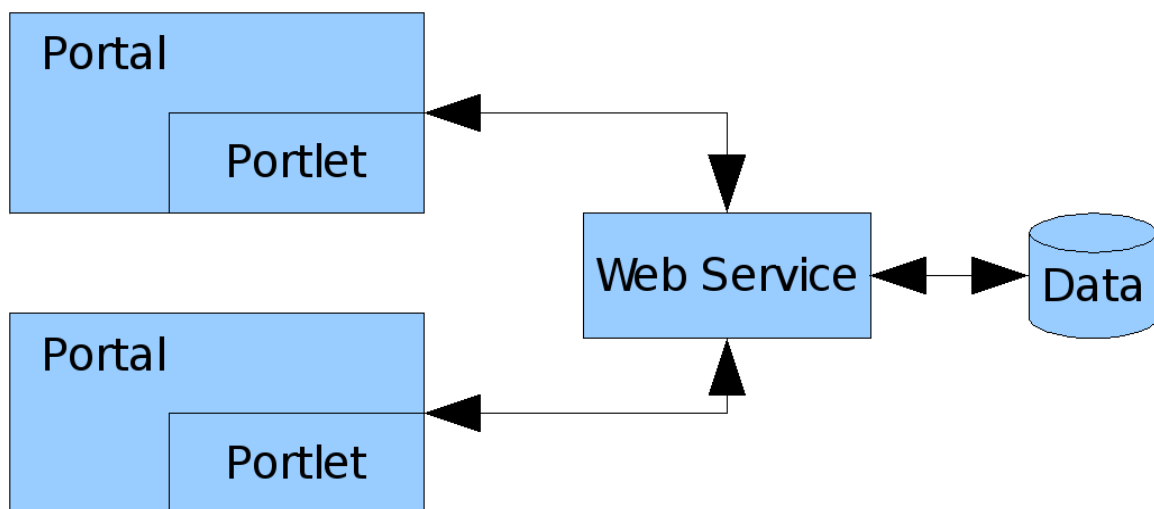
En sidste mulighed, JSR-168-portlets, er ikke inkluderet i analysen, da denne mulighed ligner WSRP meget. Den primære forskel er, at WSRP introducerer et ekstra webservice-led, der kan komplicere arkitekturen. I mange implementationer af WSRP, er der rent faktisk tale om, at portlets implementeres i JSR-168 og derefter udstilles via WSRP. Endelig er JSR-168 Java-specifik, og kan dermed ikke bruges på tværs af platforme.

Indholdet af rapporten er af en ret teknisk karakter, men hvert afsnit afsluttes med en kort konklusion, som også kan læses af personer uden dybere kendskab til teknologierne

Generelt om WSRP 1.0 og 2.0

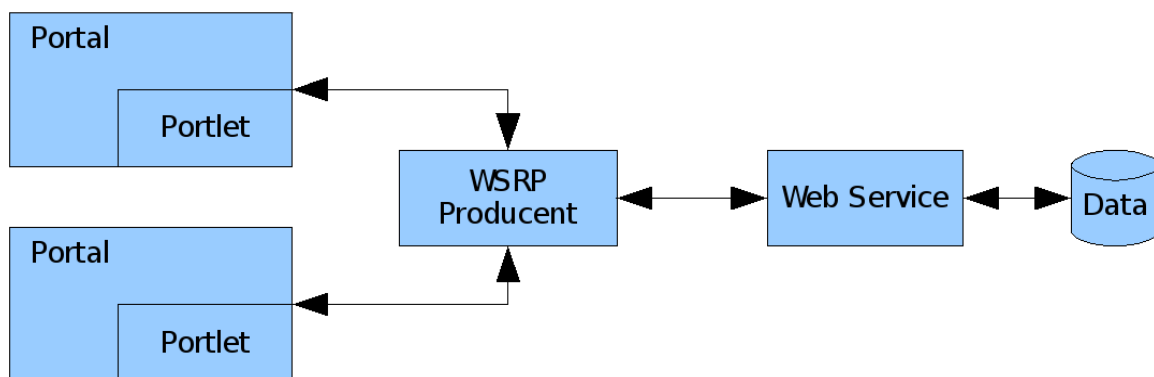
>

WSRP er en relativt simpel standard til remoting af portlets. Standarden har været i version 1.0 i længere tid, og version 2.0 er på vej ud. Oprindeligt skulle WSRP 2.0 have været klar i 2006, men den oprindelige deadline er overskredet. Lige nu foreligger 2.0 i en Committee Draft fra oktober 2006, og der er ikke fastlagt nogen dato for den endelige release. Seneste melding [TCMIN] er, at der skal være afstemning om vedtagelse af standarden omkring maj eller 2-3 måneder derefter, afhængigt af om der kommer flere ændringer.



Figur 1: Traditionelle portlets

Figur 1 viser hvordan en traditionel portal, fx med JSR-168 realiseres. Her udstilles data via en webservice, som anvendes i to forskellige portlets i to forskellige portaler. De to portlets genbruger dermed kun datakilden, selve brugergrænsefladen udvikles to gange. Figuren nedenfor viser, hvordan dette ser ud i WSRP. Her er igen to portaler, men i stedet for to forskellige portlets, der er udviklet hver for sig, er der i stedet to WSRP-portlets, der viser brugergrænsefladen, som er genereret af en WSRP-producent.



Figur 2: Portal med WSRP-portlets

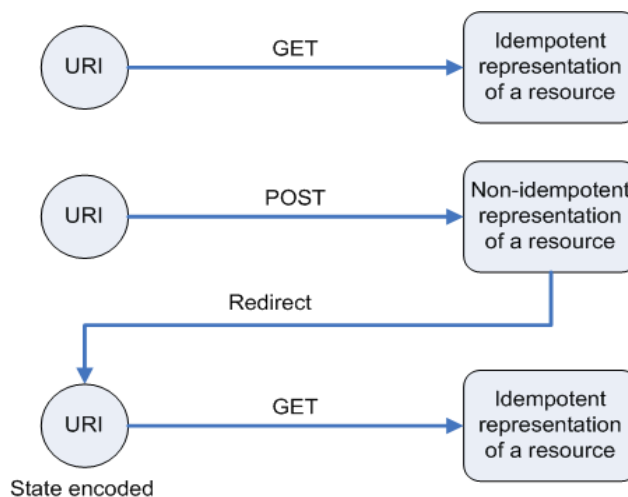
>

Der er naturligvis varianter over dette – fx behøver WSRP-producenten ikke at anvende en webservice, men kan i stedet hente data direkte. Da der i mange tilfælde alligevel skal udvikles en webservice, vil det dog være nærliggende at bygge WSRP-producenten ovenpå denne i stedet.

WSRP definerer to aktører, en producent og en aftager, typisk en portal. Producenten udvikler og udstiller portlets, som aftageren kan anvende.

Hovedfeatures i WSRP 1.0:

- > Discovery og registrering: Der er et standardiseret interface til at fremsøge og registrere brugen af portlets. Aftagere registrerer sig som bruger af en portlet og får ved den lejlighed udstedt et unikt id, der persisteres og bruges ved fremtidige requests som unik identifikation af en bestemt portlet.
- > 2-fase-strategi til at interagere med portlets: I stil med normal http interageres der med portlets i to trin: Et trin til tilstandsændring og et trin til at hente markup. Dette er illustreret i figuren nedenfor.



I kaldene til producenten, kan aftageren medsende information om den aktuelle bruger, session, portalspecifikke præferencer og andet. Disse informationer kan bruges af producenten til at tilpasse udseende og tilstand, så den valgte portlet præsenteres på en passende måde.

Portlets er karakteriseret ved, at de fungerer som selvstændige komponenter, der som udgangspunkt ikke har nogen relation til evt. andre komponenter. De kan konfigureres og tilpasses på forskellige måder: Ved registreringen kan aftageren medsende parametre, og ved alle kald til portlets medsendes argumenter i form af session-identifikation og portlet-tilstand.

>

WSRP 2.0 udvider den forrige version ved at indføre et ekstra led i strategien, så der nu er tre trin:

1. Kald metode der kan ændre tilstand (`performBlockingInteraction`) og returnere genererede events
2. **Håndter hændelser (`handleEvents`)**
3. Hent markup (`getMarkup`)

Det nye er altså, at der er lagt hændelser og notifikationer ind. Dette kan bruges i de tilfælde, hvor en portlet ikke er helt isoleret, men gerne vil vide noget om omverdenen. Dette har ikke været muligt i WSRP 1.0 uden leverandørspecifikke udvidelser.

Kort producentoversigt

>

En række af de kommercielle portalleverandører understøtter WSRP 1.0. De fleste har implementeret en del af WSRP 2.0 i form af proprietære udvidelser, og vejen til WSRP 2.0 burde derfor ikke være særlig lang.

- > Oracle AS understøtter allerede WSRP 2.0, med forbehold for de ændringer, der måtte komme i final release. Endelig support skulle komme i R11.
- > BEA WebLogic understøtter WSRP 2.0 i version 10, der netop er udkommet.
- > IBM WebSphere Portal 6.0 understøtter WSRP 1.0, og der er ikke nogen publiceret plan for 2.0-support. Lige som andre leverandører, er de fleste 2.0-features implementeret som proprietære udvidelser, så et godt bud er, at WSRP 2.0 er understøttet i næste version. Hvis afstanden mellem versioner er den samme som tidligere, kommer den næste version i starten af 2008. WebSphere Application Server 6.1 har indbygget JSR-168 portlet container, og det er også en mulighed, at denne udvides til at understøtte WSRP.
- > Microsoft er tilbageholdende med at understøtte WSRP, da det er i konflikt med deres egne WebParts, og meget tyder i det hele taget på, at SharePoint, som er Microsofts portalløsning, mest fokuserer på at være aftager. SharePoint understøtter ikke WSRP umiddelbart, men kræver at det installeres som en udvidelse. Udvidelsen [SPWP] er kun kommet i en enkelt version i 2004, og er markeret som en betaversion. SharePoint 2007 skulle have understøttelse for visning af WSRP-portlets indbygget, men kan som udgangspunkt ikke udstille WSRP-portlets til andre [SPDS] [SPWS], så portlets kan ikke udvikles i SharePoint 2007 uden ekstra udvidelser. Det vides ikke om udvidelsen til SharePoint 2003 også fungerer med 2007. Det fremgår ikke, om 2007-udgaven er baseret på WSRP version 1.0 eller 2.0, men 2.0 har ikke været nævnt nogen steder, så det er sandsynligt at det kun er 1.0.
- > NetUnity har et lignende framework til .NET for WSRP 1.0. Her foreligger heller ingen plan for WSRP 2.0-understøttelse.

Denne analyse har ikke systematisk gennemgået de enkelte portalimplementationer for interoperabilitet, men det anbefales på det kraftigste at undersøge, om der er problemer med at få nogle af dem til at køre sammen. Der skulle i forbindelse med WSRP-specifikationsarbejdet være foretaget interop-tests, men der er kommet mange nye portalversioner siden da, og det kan have givet anledning til nye problemer. Der eksisterer desuden et testframework, der kan bruges til at teste WSRP-implementationer [WCTK].

Endelig skal det nævnes, at open source-implementationerne i flere tilfælde har problemer med interoperabilitet [IWOS]. Selvom der kun fokuseres på kommercielle portalprodukter, er det ikke hensigtsmæssigt hvis der ikke er nogen open source-muligheder. Især er det et problem, hvis producenter tvinges til at investere i kommercielle løsninger, da de kan være både tunge og dyre. Der bør derfor være fokus på at inddrage open source-løsninger, så disse også kan indgå i den samlede løsning.

Simpel WSRP



Den simpleste brug af WSRP er i det tilfælde, hvor der blot vises et stykke statisk html uden konfigurationsmuligheder. Her er der, bortset fra udvidet fejlhåndtering og større latency, ingen forskel fra normale portlets.

I forhold til visning af portlets, er det i høj grad overladt til portalen. Ensartet visning kræver, at styles mv. standardiseres over portlets. WSRP definerer nogle forskellige tilstande, portlets kan være i (visning/redigering/preview), og nogle visningsformater (minimeret, maksimeret, alene). Disse kan bruges som hints i forhold til, hvordan portlets placeres i forhold til hinanden. Den konkrete visning er dog stadig op til portalen, og her varierer mulighederne fra leverandør til leverandør.

Når de enkelte portlets lever i hver deres verden, kan det være svært at få dem til at fremstå som om at de indgår i en større sammenhæng. Dette understreges især af, at de fleste portaler som standard sætter portlets ind i vinduer med rammer og knapper.

Brugen af WSRP i forhold til fx JSR-168-portlets gør ikke den store forskel. Som nævnt, så vil det mest have indflydelse på performance, især på portlets der ikke kan caches. Det betyder, at det vil være hensigtsmæssigt at holde antallet af portlets på en side på et minimum for at mindske den tid det tager at vise siden.

Udviklingsmæssige konsekvenser

En konsekvens af at bruge WSRP er, at ansvaret for udviklingen af portlets kan flyttes rundt. I det ekstreme tilfælde, er portalen kun en tynd skal, der udelukkende trækker portlets ind og præsenterer dem. Det betyder, at selve udviklingen af portalen potentielt kan gå meget hurtigt, hvis de enkelte portlets allerede er udviklet.

I den aktuelle situation er der dog allerede en række myndigheder, der har udviklet selvstændige sites, mange af dem med tilhørende webservices. Det betyder, at udviklingen af portlets potentielt er en ekstra byrde, der ikke umiddelbart giver udbytte for andre end portalen. Det er derfor vigtigt at indtænke udviklingsprocessen i planlægningen af en sådan portal, da det er nødvendigt med et bredt samarbejde for at implementere en sammenhængende portal.

Opsummering

	WSRP 1.0	WSRP 2.0	IFrames	Site
Forbindelse til producer	Via portal	Via portal	Direkte fra browser	Via site
Visninger	Forskellige views: Minimeret, maksimeret, normal, fuld skærm. Portal definerer rammer og muligheder for at flytte rundt på portlets.	Se WSRP 1.0	Indlejret	Ingen begrænsninger, men intet er standardiseret, og skal dermed laves fra bunden.
Rendering	Afhængig af portal	Afhængig af portal	Parallel	Ingen begrænsninger
Ansvarlig for udvikling	Producent	Producent	Producent	Portal

>

Tabel 1: Muligheder i forhold til simpel integration

Simpel WSRP er baseret på selvstændige portlets, der ikke har behov for kommunikation med andre portlets. Her giver WSRP følgende:

- > Standardiseret protokol til at hente visning af portlets
- > Standardiserede visninger og skift mellem visningerne
- > Adskillelse mellem portal og portlet

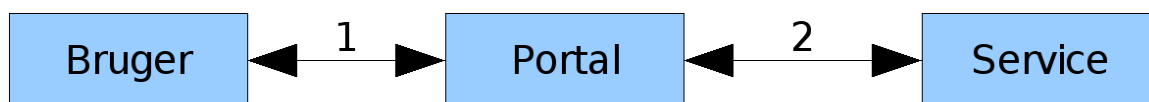
Det giver her god mening at bruge WSRP, hvis ellers der er behov for remoting og understøttelse af flere platforme. Almindelige JSR-168-portlets giver stort set samme fordele som WSRP, med den forskel at portlets skal ligge på samme server som selve portalen, og at portalen skal være baseret på Java.

For de simple tilfælde, har IFrames og manuel integration ingen videre fordele. IFrames er simple og hurtigere, men har ikke standardfeatures som forskellige visninger.

Sikkerhed i WSRP

>

Den væsentligste problemstilling i forhold til sikkerhed og WSRP (og webservices generelt) er, hvordan der sikres tillid (trust) mellem producent og aftager. Scenariet er som vist på figur 3:



Figur 3: Kommunikation i portalscenarie

Her ligger en service bag portalen – det kan enten være en normal webservice eller en WSRP-portlet – som er udviklet og hosted uafhængigt af portalen. Den ligger ikke nødvendigvis på samme netværk, og der er dermed ikke umiddelbart nogen sikkerhed for, at det portalen fortæller om brugeren. Oftest skal portalen sige noget om, hvem brugeren er, i form af brugernavn eller cpr-nummer, og det er denne oplysning, der ikke er garanteret.

WSRP indeholder i sig selv ikke nogen sikkerhedsaspekter, da det antages, at sikkerhed håndteres af andre protokoller. I forbindelse med WSRP skal der tages hensyn til flere former for sikkerhed:

- > Transportsikkerhed mellem producent og aftager. Dette kan løses simpelt ved at kommunikere over https eller mere kompliceret ved at anvende WS-Security. Ingen af delene understøttes direkte i portalerne, så det er noget, der skal konfigureres på et lavere niveau, fx i Axis eller lignende. Mulighederne er dermed meget afhængige af, hvilken portal og hvilken portletcontainer, der er tale om. Det sikre valg er derfor at bruge almindelig https.
- > Tillid mellem producent og aftager. Her er spørgsmålet hvordan producenten er sikker på, hvem brugeren hos aftageren er. I den anledning er det relevant først at se på, om det er nok, at producenten blot stoler på, at aftageren har foretaget den påståede autentifikation. I så fald, kan aftageren blot medsende fx cpr-nummer til producenten, der så returnerer personoplysninger på baggrund af det. Hvis det er nødvendigt med en yderligere garanti for, at det rent faktisk er en konkret bruger, der sidder i den anden ende, så er der brug for en anden mekanisme, fx SAML (version 2.0, som der også refereres til i det følgende), hvor producenten i princippet laver en selvstændig verifikation af den udførte autentifikation.
- > Udveksling af personfølsomme oplysninger. Problemet her er især hvordan en bruger får kontrol over, hvem, i form af services, der får adgang til data. Igen kan SAML og lignende teknologier, fx Shibboleth [SHIB], hjælpe, men her kræves der endnu mere af portalen. Hvis brugeren skal bekræfte hver gang, fx cpr-nummer kommunikerer til en ny myndighed, så skal portalen indeholde funktionalitet til at holde styr på dette. Så vidt vides, understøtter ingen af portalerne dette i øjeblikket, så det er noget, der skal udvikles selvstændigt, formentligt til den konkrete portal.
Et alternativ er, at producenten selv holder styr på godkendelser. Når en portlet vises første gang, kunne indholdet være en godkendelse af, hvad der skal udveksles for at bruge den.

Alt efter hvilken portal, der er aftager, vil der være forskellige muligheder for at definere en sikkerhedskontekst. Flere portaler understøtter SAML som sikkerhedsmekanisme, fx BEA WebLogic. Hvis en mere kompliceret løsning kræves, så kan WSRP 1.0 også udvides i nogen grad. Det drejer sig primært om, at det er muligt at sende vilkårlige attributter med i kald fra aftageren til producenten, og disse attributter kan anvendes til at etablere en sikkerhedskontekst. Dette kræver dog, at aftagere og producenter er enige om disse udvidelser og deres betydning, og at portal og portlet skrives på en sådan måde, at de tager hensyn til udvidelserne. WSRP siger i øvrigt ikke noget om, hvad der kræves af autentifikation hos aftageren. I requests til producenten medsendes information om, hvordan brugeren er autentificeret, og det kan producenten bruge efter behov. Det betyder også, at det ikke er inkluderet i standarden, at nogle portlets kun skal vises hvis brugeren er logget ind, og dermed er det en del af de features, der udbydes af de enkelte portaler.

Logning

WSRP siger i sig selv ikke noget om logning, men når portlets kræver personfølsomme informationer, fx cpr-nummer, skal der logges når de bruges. Logning kan ske to steder: På portalen og i de enkelte portlets.

Som udgangspunkt vil logning på portalen oftest bestå af en normal access log, der blot viser sidevisninger. Det er dermed ikke på portlet-niveau, så de formelle krav opfyldes ikke på den måde. Logning af visning af de enkelte portlets skal dermed konfigureres på portalserveren, og her vil der være forskellige muligheder alt efter hvilken leverandør der er tale om. Selv da kan det blive et problem at få alle relevante oplysninger med i loggen, fx oplysninger om hvem den nuværende bruger er.

Logning kan (og bør) også ske på de enkelte portlets. Her er det muligt at logge hvem der bruger de enkelte portlets hvornår. Logning på dette niveau giver dog ikke noget samlet overblik over en brugers færden, da de enkelte logfiler vil ligge hos et antal forskellige myndigheder.

Opsummering

	WSRP 1.0	WSRP 2.0	IFrames	Site
Transportsikkerhed	Håndteres i transportlaget	Håndteres i transportlaget	Afgøres af producenten	Ingen begrænsninger
Trust	Portalen repræsenterer brugeren. Portalen håndterer login. Det er muligt at implementere trust via fx SAML.	Portalen repræsenterer brugeren. Evt. brug af SAML assertions.	Direkte kommunikation med brugeren. Brugeren autentificeres hos producenten.	Sitet repræsenterer brugeren og håndterer login

>

Udveksling af oplysninger	Portal håndterer udveksling	Portal håndterer udveksling	Ingen udveksling. Hvis de forskellige producenter kører på forskellige domæner, kan cookies heller ikke udveksles.	Site håndterer udveksling
Logning	Logning både på portal og producent. Log på portal kræver yderligere konfigurationsmuligheder.	Se WSRP 1.0	Kun logning hos producent	Log på sitet kan kodes ind. Log hos producenten foregår på webservice-niveau.

Tabel 2: Oversigt over muligheder i forhold til sikkerhed

Sikkerhed er et af de store problemområder i WSRP. Det skyldes primært, at WSRP delegerer sikkerhedsspørgsmålet til andre relevante standarder, og interaktionen mellem WSRP og de pågældende standarder er dermed overladt til de enkelte implementationer.

Selvom WSRP i sig selv ikke siger noget om det, er det vigtigt at producent og aftager er enige om følgende, uanset hvilken teknologi der ligger nedenunder:

- > Etablering af et tillidsforhold, formentligt vha. SAML 2.0
- > Transportsikkerhed. Almindelig SSL er den simpleste løsning, og WS-Security vil formentligt kun komplicere arkitekturen.
- > Sikring af dataudveksling, så brugeren får kontrol og indsigt i, hvilke data der sendes frem og tilbage.
- > Hvordan der logges, og hvilke logs der kan/skal/må ikke samkøres.

Denne analyse kommer ikke med konkrete anbefalinger til, hvordan disse punkter implementeres, da området er så omfattende, at det bør gennemgå en dybere analyse, herunder konkrete eksperimenter med at implementere en dækkende sikkerhedskontekst på tværs af platforme.

En dybdegående analyse vil være nødvendig uanset hvilken tilgang der anvendes, da problemstillingerne er de samme, som det også kan ses af tabel 2.

Context Read og Write



En portal består typisk af flere portlets, der sættes sammen på en side. Nogle portlets lever helt i deres egen verden, men andre vil gerne enten modtage eller publicere informationer. Det kan fx være, at en portlet indeholder en komponent til at vælge kommune, mens en anden portlet viser information om den aktuelle kommune. Dette kræver, at vælgeren publicerer information om, hvilken kommune der er valgt, og at viseren får den information stillet til rådighed. Den bedste måde at lave denne kommunikation på er ved, at portlets kan udsende notifikationer om, at deres tilstand er ændret. Andre portlets kan derefter lytte efter disse ændringer og reagere på dem. Denne metode sikrer, at portlets ikke direkte kender til hinanden, men bare reagerer på hændelser. Hændelserne formidles af portalserveren, og dermed opnås der en lav kobling mellem de konkrete portlets.

WSRP 1.0 (og JSR-168) giver ingen videre hjælp til at lave kommunikation mellem portlets. Det er ikke muligt at portlets kan udsende og modtage notifikationer, når der sker ændringer i systemet i WSRP 1.0, og det er derfor ikke muligt at lave løst koblede portlets i WSRP 1.0 alene. WSRP 2.0 retter op på dette, og de fleste leverandører har allerede implementeret en eller anden form for IPC (Inter Portlet Communication).

Alternativet er at anvende session state i portalen til at udveksle tilstanden. Det betyder, at når der udføres en operation på en portlet, skal portalen lægge resultatet i brugerens session. Når portlets derefter vises, skal portalen tage de korrekte attributter fra brugerens session og sende med som argumenter til de enkelte portlets. Dette skaber en relativt høj kompleksitet i selve portalen, og de enkelte portlets bliver bundet hårdt op på hinanden.

Portalen bliver altså koordinator for de enkelte portlets. Brugerens session bliver ikke direkte kommunikeret til de enkelte portlets, da der eksisterer en session pr. portlet. Portalen er dermed nødt til at være involveret i processen.

Rent konkret er der tale om, at navigationState skal sendes rundt til de enkelte portlets. NavigationState fungerer lige som normale GET-parametre, og angiver hvor brugeren befinder sig. Det kan fx dreje sig om, hvilken bopælskommune der er valgt.

Hvis der skal integreres portlets, der skal kommunikere med hinanden, anbefales det at anvende WSRP 2.0, da alternativet vil give en masse udfordringer og en høj kobling mellem portlets.

Opsummering

	WSRP 1.0	WSRP 2.0	IFrames	Site
Overførsel af tilstand til portlet	Via navigationState. Dette kræver dog kendskab til den semantiske betydning af indholdet.	Se WSRP 1.0	Via GET-parametre	Ikke relevant

>

Overførsel af tilstand fra portlet	Via navigationState. Dette kræver dog kendskab til den semantiske betydning af indholdet.	Se WSRP 1.0	Muligt via Javascript, men giver en meget tæt kobling.	Ikke relevant
Kommunikation mellem portlets	Mulig via session-tilstand, koordineret af portal. Skaber tæt kobling.	Ja, via notifikationer.	Nej	Håndteres internt. Vil formentligt skabe tæt kobling.

Tabel 3: Muligheder for at kommunikere mellem portlets og portal

Muligheden for at lave kommunikation mellem portlets bliver hurtigt en nødvendighed. WSRP 1.0 er i den sammenhæng for mangelfuld, og det giver ikke mening at prøve at arbejde sig udenom problemerne i denne version. WSRP 2.0 retter op på dette, og det bør derfor være denne version, der tages i anvendelse. Hvis der tages udgangspunkt i brugen af IFrames eller manuel integration, opstår de samme problemer som med WSRP 1.0, så bortset fra JSR-268 (portlet 2.0), har WSRP 2.0 nogle klare fordele her, som gør IPC muligt på en standardiseret måde.

Portlet Præferencer



For at kunne genbruge portlets bedst muligt, er det nødvendigt at de kan tilpasses i en sådan grad, at de passer ind i de sammenhænge, de anvendes i. WSRP giver en række muligheder, bl.a. er det i forbindelse med registreringsprocessen muligt at angive en række parametre, der kan bruges af producenten til fx at tilpasse udseendet. Det er op til producenten at definere hvilke parametre, der er mulige at angive, og hvad de skal bruges til.

Parametrene knytter sig til en bestemt registrering. Det betyder, at hvis parametrene er brugerspecifikke, skal en portlet registreres en gang pr. bruger. Dette scenario er understøttet af WSRP-standarden.

Opsummering

	WSRP 1.0	WSRP 2.0	IFrames	Site
Overførsel af parametre	I forbindelse med registrering	Se WSRP 1.0	Kun via GET-parametre	Ikke relevant

Tabel 4: Tilpasningsmuligheder

WSRP giver mulighed for at specificere vilkårlige parametre til en portlet, og dermed kan portlets tilpasses mere eller mindre i det uendelige. Udviklingsmæssigt bliver portlets dog også tilsvarende mere komplicerede at vedligeholde, så der er en grænse for, hvor meget portlets i praksis kan tilpasses.

I langt de fleste tilfælde vil det formentligt være sådan, at for at opnå en tilpas integration, skal portlets udvikles til den konkrete portal. Det vil også skabe mere tryghed ved opgraderinger af portlets, da der kun er en aftager at tage hensyn til. Spørgsmålet om præferencer er dermed indsnævret en del, og i stedet for at fokusere på at kunne tilpasse en portlet fuldstændigt, kan der fokuseres på at kunne tilpasse en portlet efter den konkrete kontekst indenfor en portal.

HTML Forms



HTML forms er en integreret del af de fleste websites, og de vil også skulle optræde i portlets. Grundlæggende er der ikke noget i vejen for, at der anvendes HTML forms i portlets. Der er dog visse steder hvor det er nødvendigt at være opmærksom:

- > Det er ikke muligt at kombinere flere forskellige portlets til en samlet form. En form sendes til en, og kun en, portlet.
- > Hvis en portlet har upload af filer, skal filen først sendes til portalen, som skal pakke den ind i et soap-request. Alt efter hvilken webservice-stack der er tale om, kan det være tungt og kræve store mængder hukommelse.
- > Nogle frameworks, fx nogle JSF-implementationer, går ud fra, at der kun er én form pr. side, hvilket kan være et problem for portlets, der går ud fra, at de selv skal definere en form.

Den største udfordring i forhold til forms er formentligt at holde et konsistent layout og udseende. Det dækker fx over, at felter og hjælpetekster er placeret på samme måde, at de kommer i en ensartet rækkefølge, og at indtastninger af fx datoer foregår på samme måde.

Opsummering

	WSRP 1.0	WSRP 2.0	IFrames	Site
Simple forms	Muligt	Muligt	Muligt	Muligt
Forms på tværs af portlets	Ikke muligt	Ikke muligt	Ikke muligt	Ja
Filupload	Ja	Ja	Ja	Ja
Standardiseret visning	Ingen	Ingen	Ingen	Muligt

Tabel 5: Muligheder for at bruge forms

Så længe der ikke er behov for, at forms fra forskellige portlets kombineres til en samlet form, opfylder WSRP de gængse krav rent teknisk. Det største problem vil ligge i at opnå en standardiseret visning. Igen er problemet ikke unikt for WSRP, men det gør det ikke mindre vigtigt.

Det er altså nødvendigt at definere nogle fælles regler for, hvordan forms sættes op, hvordan felter ser ud, og hvordan fælles datatyper indtastes.

Javascript og AJAX

>

AJAX og Javascript er blevet nogle af kerneteknologierne til at gøre websites mere interaktive og attraktive generelt. Det drejer sig fx om at kunne auto-udfylde forms med data på baggrund af tidligere indtastninger, og generelt at kunne ændre en sides tilstand uden at skulle genindlæse hele siden.

AJAX er ikke tænkt ind i WSRP 1.0, og det giver nogle begrænsninger i, hvordan det kan bruges. Først og fremmest kan normal Javascript kun bruges i begrænset omfang. En portlet lever i princippet isoleret, men i Javascript deles et globalt namespace. Det vil sige, at definerede funktioner kan være i konflikt med funktioner defineret i andre portlets. Ligeledes kan der være problemer hvis forskellige portlets anvender forskellige versioner af eksterne libraries som fx Dojo. Den eneste sikre form for Javascript er dermed inline scripts, fx i onmouseover på et html-element.

For AJAX er der nogle flere problemer, idet browseren gerne vil kommunikere direkte med en service. Det skal ske gennem en portal, men portalen understøtter ikke den form for direkte kommunikation. Det er godt nok muligt at sende et request til en bestemt portlet, men svaret vil blive behandlet af portalen, og selvom den enkelte portlet returnerer fx et svar i XML, så vil portalen tage svaret og sætte ind i den normale ramme, og derved returnere en normal html-side til browseren.

En mulig vej udenom dette er at anvende ressource urls til at kommunikere AJAX, men det åbner op for nogle sikkerhedsproblemer, idet browseren kommunikerer direkte med producenten, og nogle browsere tillader ikke AJAX-requests til andre domæner end det, portalen ligger på. Se Subbu Allamaraju, der har været med til at specificere WSRP 1.0, har en yderligere gennemgang af problemstillingerne [SUB].

Opsummering

	WSRP 1.0	WSRP 2.0	IFrames	Site
Javascript	Begrænset	Begrænset	Ja	Ja
AJAX-serverkald	Kun direkte til producenten	WSRP 2.0 tillader portlets at håndtere ressource urls, og dermed kan der laves serverkald via en portlet.	Ja	Ja
Brug af eksterne biblioteker	Frarådes kraftigt	Se WSRP 1.0	Ja	Ja

Tabel 6: Muligheder for at anvende Javascript og AJAX

Som det ses af tabel 6, giver både WSRP 1.0 og 2.0 en række begrænsninger, der ikke er lige til at løse. Problemerne opstår fordi Javascript bruger et enkelt namespace, og der kan meget nemt opstå navnekonflikter på funktioner og variabler. Derudover vil der være problemer hvis der anvendes frameworks i forskellige versioner. Det kunne fx dreje sig om populære frameworks som Scriptaculous eller Dojo.

Hvis der anvendes WSRP kræves det derfor en meget stram styring af navngivning og versionering eller at der ganske simpelt ikke bruges Javascript til andet end meget små ting. WSRP 2.0 har nogle ændringer, der muliggør AJAX-kald, men det løser ikke namespace- og versioneringsproblematikkerne.

Fejlhåndtering

>

Som tidligere nævnt, så gør WSRP det muligt at have portlets kørende på andre maskiner end selve portalen. Det åbner op for en række nye fejlmuligheder – hvad sker der fx hvis portalen kører mens en portlet er nede? Derudover er der behov for at håndtere andre fejlsituationer, fx hvis processeringen i en portlet fejler, hvis brugeren ikke er logget korrekt ind osv. Alt efter hvilken fejl der er tale om, kan der være forskellige muligheder for at komme videre. Er en portlet helt nede, kunne der fx blive vist en side med andre kontaktmetoder som telefon, email mv.

WSRP i sig selv definerer en række fejkoder, som portalen kan reagere på. Disse fejkoder er primært rettet på registreringsprocessen, og bør generelt ikke opstå når en portlet først er konfigureret. Fejkoderne er defineret i kap. 13 i WSRP 1.0.

Det betyder, at en fejl som ”korrekt svar med autoriseringsfejl” [SAF] ikke er en fejl i WSRP-termer, men blot en almindelig visning af markup fra en portlet, hvor markup angiver at der er opstået en fejl.

Sker der runtimefejl, fx i form af timeout eller andre kommunikationsfejl, tilbyder de forskellige portaler forskellige muligheder for at håndtere dette. Det er fx muligt i BEA WebLogic at definere en JSP-side, der vises når der opstår en fejl i en bestemt portlet. I disse tilfælde er det altså ikke producenten selv, der kan definere disse alternative sider.

Opsummering

	WSRP 1.0	WSRP 2.0	IFrames	Site
Fejlmuligheder	Et antal kendte fejlsituationer er beskrevet.	Samme som WSRP 1.0	Ingen	Frit
Visning af alternative sider	Portalspecifikt	Portalspecifikt	Nej	Frit

Tabel 7: Muligheder for at håndtere fejl

Håndteringen af fejl i WSRP er rimelig basal. Der er defineret et antal fejkoder, men disse er formentligt kun dækkende i de simpleste situationer. Håndteringen af fejlene er overladt til de enkelte portaler, og dermed er der relativt lidt kontrol over, hvad der sker når der opstår fejl.

Notifikation og hændelser



Som allerede nævnt, så er dette ikke en del af WSRP 1.0, til gengæld kommer det i WSRP 2.0. Notifikationer er en praktisk nødvendighed når portlets skal dele tilstand, og det er en af hovedårsagerne til, at WSRP 1.0 ikke er tilstrækkelig i andet end de mest simple tilfælde.

Ændring af view

>

WSRP-specifikationen definerer et antal forskellige tilstande, som en portlet kan være i. Det drejer sig om minimeret, maksimeret, normal og fuld skærm. Det er ikke nødvendigt for alle portlets at understøtte alle disse, og hvilke der er understøttet fremgår under registreringsprocessen.

Ud over de definerede tilstande, åbner standarden også op for, at der kan defineres yderligere tilstande hvis det er nødvendigt. I de fleste tilfælde vil de allerede definerede dog være tilstrækkelige.

Tilstandene er ment som hints til portalen, der så kan render de enkelte portlets i forhold til deres tilstande. For at det virker, er det altså nødvendigt, at aftager og producent er enige om, hvad de forskellige tilstande betyder.

Hvorvidt brugeren rent faktisk har mulighed for at skifte mellem de forskellige tilstande er op til den enkelte portal-implementation. De fleste implementerer dog denne funktionalitet i form af vinduer, hvor det i titel-feltet er muligt at maksimere og minimere vinduerne.

Opsummering

	WSRP 1.0	WSRP 2.0	IFrames	Site
Definerede views	Minimeret/maksimeret/normal/alene	Samme som 1.0	Ingen	Ingen

Tabel 8: Muligheder for at ændre visninger

Hvis der er behov for at kunne definere views og skifte mellem dem på en standardiseret måde, så er WSRP vejen frem, som det også fremgår af tabel 8. Alternativet er at lave det helt fra bunden, hvilket formentligt ikke er hensigtsmæssigt.

Skins og struktur

>

En af fordelene ved WSRP er, at der kommer markup direkte fra de enkelte portlets, og ud over evt. omskrivning af links, sker der ingen behandling af markup. Det betyder, at for at markup kan genbruges i forskellige sammenhænge, kræves det som det mindste, at struktur og præsentation er adskilt mest muligt. I HTML sker det normalt ved, at der anvendes CSS til at definere udseende. CSS defineres i eksterne stylesheets, der så kan tilpasses efter behov.

Samme princip gør sig gældende for WSRP. Her opfordres der til at bruge nogle standardklasser til at beskrive indholdet. Disse er beskrevet i WSRP-standarden [WSRP], og gør at samme type elementer har samme style-klasser. Dermed er det muligt for en portal at definere styles, der dækker alle de viste portlets. Det er ikke muligt for portlets at levere CSS-filer selv, da <link>-tags kun er tilladte i HTML <head>. Portlets kan bruge inline CSS, men dette frarådes generelt. Det betyder til gengæld, at hele præsentationsopgaven ligger hos portalen. Det vil formentligt være en god idé hvis de enkelte producenter offentliggør et stylesheet for de enkelte portlets, som portalen kan anvende som udgangspunkt.

De klasser der er defineret i WSRP-standarden er formentligt ikke dækkende i alle tilfælde, og selv når der er enighed om CSS-klasser, kan der være masser af variation i, hvordan selve HTMLen bliver opbygget. Det kan fx være et spørgsmål om, om der anvendes tabeller eller ej. Alt dette er udenfor selve WSRP, men for at opnå et ensartet udseende, er det nødvendigt at alle producenter er enige om nogle guidelines for, hvordan grafiske elementer opbygges.

Et alternativ (eller supplement) til ren CSS er, at en portlet kan tilpasses med parametre. Dette er allerede til dels dækket i afsnittet om præferencer. Det er dog værd at bemærke, at jo mere en portlet kan tilpasses, jo mere generel bliver den også. Det kan være godt til en vis grad, men på et tidspunkt kan den gå hen og blive for generel, så den ikke passer ordentligt ind nogen sammenhæng længere.

Opsummering

	WSRP 1.0	WSRP 2.0	IFrames	Site
Tilpasning af præsentation	Portal-CSS	Samme som 1.0	Styres af producenten	Ingen begrænsninger, alt skal laves fra bunden.

Tabel 9: Muligheder for at tilpasse præsentation

Layout er en vigtig del af at integrere forskellige portlets i en samlet portal. WSRP hjælper ikke direkte på, hvordan det kommer til at se ensartet ud, men den giver nogle CSS-klasser, der med fordel kan anvendes.

Det er vigtigt at være opmærksom på, at CSS ikke er alt. Som fx nævnt i afsnittet om forms, er det vigtigt, at elementer er opbygget ens, og at elementer der går igen også ligner hinanden.

Styring af flowcontrol



Når der eksisterer et antal portlets, der hver især har et ansvarsområde, er det nærliggende at forestille sig at kombinere disse portlets til et større workflow. De enkelte portlets indgår dermed som komponenter i en proces, der mere eller mindre er defineret ved de portlets, der anvendes og deres rækkefølge.

Flowkontrol og kombination af portlets er helt uden for WSRP-specifikationens scope. Det vil stille meget høje krav til en portal, hvis det skal være muligt at implementere vilkårlige workflows, og derfor vil en mere praktisk løsning være at udstille de nødvendige parter som services og så bygge portlets, der implementerer de forskellige workflows.

Generelt vil det være en god ting at udarbejde nogle konkrete eksempler på brugbare workflows. Tages der udgangspunkt i det eksisterende borger.dk, er det ikke umiddelbart muligt at se, hvad der er behov for, og det gør det svært at sige noget konkret om workflows.

Det er vigtigt at det bliver afklaret, om der vil være genbrugelige elementer på tværs af workflows, eller om indholdet af de enkelte flows er så specifikt, at det alligevel skal fremstilles til det specifikke flow. Hvis det viser sig, at der er en høj grad af genbrug, kan det blive aktuelt at se på at udvikle en fælles ramme, hvor workflows kan udvikles i. Er der kun et begrænset antal workflows, eller er de meget forskellige, så kan omkostningerne ved at lave en fælles ramme hurtigt blive højere end ved at implementere hvert flow for sig.

Uanset hvad vil det kræve en betragtelig mængde udvikling for at skabe et workflow, og det er ikke sandsynligt, at det vil kunne genbruges i særlig høj grad på tværs.

Performance og skalerbarhed



Brugen af WSRP giver anledning til en række performance- og skalerbarhedsforhold, der også bør indgå i den overordnede vurdering af WSRP.

Late binding

I brugen af ”almindelige” webservices, anvendes ofte et princip om late binding for at undgå hårde bindinger til bestemte adresser og services. En måde at implementere dette på er ved at bruge UDDI som opslagsserver når en bestemt service skal bruges. I stedet for at kode adressen på en service ind, laves i stedet et opslag i UDDI, som har den korrekte adresse. Det betyder, at det altid er muligt at flytte en service eller at bytte den ud med en anden med samme interface, alt sammen uden at skulle ændre noget i den kørende applikation.

Samme princip kan bruges i forbindelse med WSRP. En gruppe under WSRP-gruppen har beskrevet hvordan UDDI kan bruges til dette formål [WSUD]. Igen skal det bemærkes, at det er portalspecifikt om UDDI understøttes og i hvilken grad. For eksempel understøtter WebLogic Portal UDDI i nogen grad, dog kun i version 2.0 – UDDI 3.0 er den nyeste.

Parallel rendering

En måde at optimere performance ved sidevisninger er at sørge for, at portlets hentes parallelt. Det gør i princippet, at brugeren ikke skal vente længere end den portlet, der tager længst tid om at svare. På sites med mange besøgende er dette dog ikke løsningen på alle problemer, da der kan opstå problemer med båndbredde i stedet. Eftersom WSRP kører over en normal webservice-stack, er der et vist overhead ved at hente portlets fra andre servere. Samlet set kan det overhead godt blive rimelig stort, og det kan sætte en stopper for, hvor mange parallelle portlets der kan vises.

Caching

I mange tilfælde kan caching være svaret på performanceproblemer, men der er en del tilfælde, hvor caching ikke er en mulighed. Det vigtigste er der hvor portlets viser personaliseret indhold. I sådanne tilfælde skal portlets være opdaterede, og caching besværliggøres en del. Det er i forbindelse med caching vigtigt at understrege, at caching ikke er den gyldne løsning på alle performanceproblemer, men det kan i visse tilfælde hjælpe.

Skalering af producenter

Et af de steder, hvor WSRP kan hjælpe er i forbindelse med skaleringen af producenter. I et traditionelt portalmiljø, deployes portlets sammen med portalen. Det betyder, at det er selve portalen, der skal trække hele belastningen, og det kan gå hen og blive et problem, både fordi portlets potentielt kan være performancekrævende, og fordi det kan være besværligt at administrere så mange portlets på den samme maskine.

WSRP gør det muligt at distribuere belastningen på en nem måde ved helt simpelt at deploye portlets på forskellige maskiner. Dermed kan performancekrævende portlets deployes for sig selv, uden at det skaber nogen egentlige problemer for integrationen. WSRP gør det også muligt at tage en portlet-container helt ned, fx i forbindelse med opgradering, uden at hele portalen går ned.

Erfaringer

>

Det er småt med konkrete erfaringer om WSRP rundt omkring. WSRP har været anvendt i et antal pilotprojekter, men ingen projekter har efter vores oplysninger brugt WSRP 1.0 problemfrit. Det er, givet den første versions begrænsninger, forståeligt og forventeligt.

Erfaringer med WSRP 2.0 er der endnu færre af, men det må forventes, at det vil blive brugt mere effektivt end 1.0, især fordi producenter som Oracle og BEA bakker kraftigt op om WSRP 2.0.

De problemer der oftest opstår med WSRP 1.0 er, at der mangler muligheder for at lave kommunikation mellem portlets (IPC), og at det er svært at etablere en fælles sikkerhedskontekst.

Den manglende understøttelse for IPC har gjort, at nogle projekter har forsøgt at implementere det selv, men det har kun givet anledning til endnu flere problemer. Andre steder har WSRP været evalueret som en mulighed i stedet for almindelige JSR-168-baserede portlets. De primære behov er ofte at kunne opdatere portalen uden at skulle gendeploye hele sitet, uden at der nødvendigvis er behov for remoting. Derfor er WSRP blevet fravalgt til fordel for en almindelig JSR-168-implementation, hvor portlets fx hostes i WebSphere portlet-container. WSRP har i disse situationer ikke tilstrækkeligt med fordele, i stedet ville det introducere større latency og kompleksitet.

Interoperabilitet

En stikprøvekontrol mellem et antal portaler og producenter (Oracle AS 10.1.3, BEA WebLogic 9.2.1 og StringBeans 3.2) viser, at der er en række problemer med interoperabilitet. Det er langt fra en dækkende liste, den illustrerer blot, at det er vigtigt ikke at tage interoperabilitet for givet.

- > BEA Workshop kan godt registrere en producent fra Oracle, men den viser ikke de tilgængelige portlets.
- > Oracle JDeveloper kan ikke registrere en producent fra BEA. Problemet skyldes, at JDeveloper medsender autentifikationsinformation når det ikke er nødvendigt, hvilket får BEA til at afvise registreringen.
- > Når en StringBeans-producent registreres i Oracle JDeveloper, kommer der fejl i WSDL/XSD-filerne.

Problemerne ser mest ud til at skyldes enten reelle fejl i udviklingsværktøjerne eller forskellige tolkninger af de involverede standarder. Der er ikke tale om uløselige problemer, men eftersom der er lukkede produkter involveret, er det ikke muligt at rette dem selv, og dermed er det op til leverandørerne at blive enige.

Overordnet konklusion

>

WSRP leverer følgende fordele:

- > Det er muligt at genbruge portlets over flere forskellige portaler, uafhængigt af lokation.
- > Portlets kan deployes på et antal forskellige maskiner.
- > Mulighed for interoperabilitet mellem portlets.
- > Ansvar for markup og data holdes samlet.

Til gengæld lever portlets i en smal verden, hvor der ikke eksisterer andet end en enkelt portlet. Det betyder, at kommunikation på tværs af portlets ikke er muligt uden at lave forskellige manuelle løsninger. Dette er formentlig den største ulempe, da det er nødvendigt for at skabe en god brugeroplevelse.

Derudover er det tvivlsomt, at interoperabilitet kan tages for givet. WSRP-specifikationen nævner en hel række relaterede standarder, og hver af disse bidrager med kompleksitet og steder, hvor der kan være afvigelser. Endelig lægges ansvaret for at lave markup hos dataleverandøren, og det kan være tvivlsomt, om det udstillede markup tilfredsstillende behøver.

WSRP 2.0 adresserer det første punkt, da notifikationer er en del af specifikationen. Eftersom det er en helt ny specifikation, der endnu ikke er ude i en færdig version, må det dog antages, at der går et stykke tid før alle leverandører implementerer den korrekt.

Leverandøruafhængighed

Et af de erklærede mål er at opnå uafhængighed af leverandører i forbindelse med udviklingen af WSRP-portlets. WSRP gør det i en vis grad muligt at opnå en sådan uafhængighed, men da WSRP kun løser en del af kravene, er der en del funktionalitet, der skal implementeres ved siden af. Behovet for SAML er et godt eksempel.

Specifikationen kræver ikke, at en leverandør implementerer SAML-understøttelse, og dermed indskrænkes de mulige leverandører allerede der.

Hertil kommer at det på nuværende tidspunkt er meget få webservice-standarder, der er 100% interoperable. Der er masser af tilfælde, hvor specifikationerne fortolkes lidt forskelligt, og det giver problemer når de forskellige implementationer skal tale sammen. Selvom WSRP er tilpas simpel, er det ingen garanti for, at den samme slags problemer ikke opstår her. Endnu værre bliver det når WSRP-specifikationen nævner 18 relaterede standarder. Hver af disse kommer med deres egne problemer og tolkningsmuligheder.

Endelig er der spørgsmålet om selve portalen. WSRP definerer nogle forskellige styles, visninger mv., men det er op til den enkelte portal at tolke disse. Det er også portalens ansvar at få etableret en korrekt sikkerhedskontekst samt at koordinere data mellem portlets. Alt dette stiller rimelig høje krav til portalen, og det er langt fra sikkert at portlets udviklet til brug i en bestemt portal også er anvendelige i andre portaler.

Opsamling

WSRP er en interessant teknologi, der giver nogle nye muligheder for at integrere informationer. Det er dog vigtigt at se WSRP som netop dette: En "enabling"

>

teknologi – nogle ting bliver mulige på en standardiseret måde, som ikke var mulige før:

> Det giver en standard for integration på brugergrænsefladeniveau

> Muliggør distribueret udvikling

I praksis er det dog ikke helt så let. Først og fremmest er WSRP 2.0 en nødvendighed i alt andet end de mest simple tilfælde. Derudover er der det faktum, at portlets skal tilpasses den enkelte portal i forhold til

> Sikkerhed

> Look-and-feel

> Interaktioner og processer

Det er kun de mest simple portlets, der ikke har behov for tilpasninger, fx de klassiske med World Clock og vejrudsigt.

Konklusionen er derfor, at WSRP (2.0) godt kan være den rigtige vej at gå, men det er vigtigt at være opmærksom på de begrænsninger og udfordringer det medfører. Det anbefales også, at der arbejdes videre med nogle af de væsentligste problemstillinger:

> Interoperabilitet. En stikprøvekontrol viste, at flere forskellige implementationer havde problemer med at snakke sammen. Det vil være nødvendigt at klarlægge, hvor store problemerne præcist er, og om det kan bringes til at køre sammen.

> Sikkerhed. Der kræves en høj sikkerhed på mange områder, og WSRP siger ikke noget om, hvordan denne sikkerhed skal implementeres. Konkrete eksperimenter med at etablere en fælles sikkerhedskontekst bør udføres for at finde mulige løsninger.

> Workflow. Det virker usandsynligt, at en portal kan koordinere flowet mellem forskellige portlets, men det vil være en god idé at undersøge mulige strategier for at implementere workflow.

Litteraturoversigt

>

- > SAF: Den Digitale Taskforce, Samarbejdsaftale om analyse af integrationsmetoder ved brug af WSRP, 2007
- > TCMIN: WSRP TC, Minutes: WSRP TC Meeting 29 March 2007, , http://www.oasisopen.org/committees/download.php/23316/TC_Minutes_20070329.html
- > SPWP: Microsoft, WSRP Web Part Toolkit for SharePoint Products and Technologies, 2004, <http://www.gotdotnet.com/workspaces/workspace.aspx?id=2e3d8a57ec9f4d169a81a395679d6392>
- > SPDS: Microsoft, Microsoft Office SharePoint Server 2007 datasheet, 2007, <http://office.microsoft.com/enus/sharepointserver/HA102063361033.aspx>
- > SPWS: Josh Holmes, Sharepoint Web Services For Remote Portlets (WSRP), <http://geekswithblogs.net/joshholmes/archive/2007/01/07/102946.aspx>
- > WCTK: WSRPtk, WSRP V1 Conformance Test Kit, <http://wsrptk.sourceforge.net/v1Conformance/index.html>
- > IWOS: X. Yang, X. D. Wang and R. Allan, Investigation of WSRP support in selected opensource portal frameworks, 2005
- > SHIB: Internet2, Shibboleth, , <http://shibboleth.internet2.edu/>
- > SUB: Subbu Allamaraju, Portlets and Ajax How Complete are the Standards?, 2006, http://www.subbu.org/weblogs/main/2006/02/portlets_and_aj.html
- > WSRP: , WSRP 2.0 Specification, 2006, <http://docs.oasisopen.org/wsrp/v2/wsrp2.0speccd04.html>
- > WSUD: Richard Jacob and Andre Kramer, WSRP UDDI Technical Note, 2003, <http://www.oasisopen.org/committees/download.php/15798/wsrppfbudditn1.0.pdf>

Center for Serviceorienteret Infrastruktur

Center for Serviceorienteret Infrastruktur (CSI) blev oprettet 1. december 2006 for en periode på tre år. Centeret har til opgave at lede og facilitere opgaven med at producere en åben, national infrastruktur.

Centeret har samlet en række af IT- og Telestyrelsens medarbejdere, der hidtil har arbejdet med forskellige aspekter af samme felt, herunder arbejdet med serviceorienteret arkitektur (SOA), brugerstyring og infrastruktur til e-handel.

<
