

CARA CEPAT
MENGEMBANGKAN
SOLUSI JAVA ENTERPRISE
DENGAN METODE MVC

Frans Thamura

Leo Heryanto

Endy Muhardin

PENERBIT BAMBUMAS

SERI ENTERPRISE OPENSOURCE

CARA CEPAT MENGEMBANGKAN SOLUSI JAVA ENTERPRISE DENGAN METODE MVC

Frans Thamura

Leo Heryanto

Endy Muhardin

PENERBIT BAMBUMAS

Cara Cepat Mengembangkan Aplikasi Java dengan Metode MVC

Hak Cipta © 2006 pada penulis

Hak Cipta dilindungi Undang-Undang. Dilarang memperbanyak atau memindahkan sebagian atau seluruh isi buku ini dalam bentuk apapun, baik secara elektronis maupun mekanis, termasuk memfotocopy, merekam atau dengan sistem penyimpanan lainnya, tanpa izin tertulis dari Penulis dan Penerbit.

ISBN: 979-25-5950-7

Cetakan pertama : November 2006

Ketentuan pidana pasal 72 UU No. 19 tahun 2002

1. Barang siapa dengan sengaja dan tanpa hak melakukan kegiatan sebagaimana dimaksud dalam pasal 2 ayat (1) atau pasal 49 ayat (1) dan ayat (2) dipidana dengan pidana penjara paling singkat 1 (satu) bulan dan/atau denda paling sedikit Rp. 1.000.000 (satu juta rupiah) atau pidana penjara paling lama 7 (tujuh) tahun dan/atau denda paling banyak Rp. 5.000.000.000,00 (lima miliar rupiah).
2. Barang siapa dengan sengaja menyiarkan, memamerkan, mengedarkan, atau menjual kepada umum suatu Ciptaan atau barang hasil pelanggaran Hak Cipta atau Hak Terkait sebagaimana dimaksud pada ayat (1), dipidana dengan pidana penjara paling lama 5 (lima) tahun dan/atau denda paling banyak Rp. 500.000.000,00 (lima ratus juta rupiah)

Sekilas Mengenai Buku dan Penulis

Buku ini adalah sebuah buku yang dikembangkan yang menerangkan proses integrasi dari projek Open Source paling populer dimuka bumi ini menjadi sebuah projek Open Source baru, yang merupakan integrasi dari projek yang telah dipilih. Projek tersebut adalah Cimande, PASIR dan Postila.

Diskusi dengan penulis dapat dilakukan melalui forum di website Bambumas dengan URL:

<http://www.bambumas.com/forum>

Adapun profile penulis adalah sebagai berikut:

Frans Thamura

Frans Thamura adalah pendiri Java User Group Indonesia, Java Champions pertama di Indonesia, sebuah posisi bergengsi dari seseorang yang gerakannya diakui dunia, CEO dari beberapa perusahaan IT, saat ini beliau khusus melayani pendanaan gerakan Open Source berbasis Java untuk berbagai industri terutama *manufacturing* ataupun *egovovernment*, pendiri BlueOxygen, pusat pengembangan projek Open Source berbasis Java.

Beliau telah lebih dari 10 tahun malang melintang didunia IT memberikan konsultansi untuk berbagai jenis masalah manajemen operasional menggunakan IT, termasuk juga konsultansi *cyberbranding* dan *community building*.

Beliau dapat dihubungi melalui emailnya frans@intercitra.com

Leo Heryanto

Leo Heryanto adalah konsultan Java dan instruktur Meruvian Management College, *training center* Open Source Java pertama di Indonesia, aktif berkontribusi untuk pengembangan projek di dalam BlueOxygen. Hampir semua contoh kode didalam buku ini adalah karya beliau. Beliau dapat dihubungi melalui emailnya leo@meruvian.com

Endy Muhardin

Endy Muhardin adalah konsultan IT di Indonesia, mengelola beberapa projek Open Source berbasis Java diantarnya adalah Playbilling, sebuah projek Open Source *billing* warnet yang mengadopsi teknologi Spring, Freemarker, serta iBatis.

Beliau aktif didalam Java User Group sebagai pelopor adopsi Spring untuk segala solusi. Beliau saat ini *full time* bekerja di perusahaan BaliCamp sebagai konsultan IT. Beliau dapat dihubung melalui emailnya endy@artivisi.com

Kata Pengantar

Tidak terasa hampir 6 tahun berlalu, tepatnya sejak January 2001, projek BlueOxygen telah dijalani, banyak sekali solusi dan teknologi yang digunakan dan diganti dalam proses pengembangan dan sampai akhirnya menjadi sebuah buku, dan dapat terbit. Buku ini juga adalah sebuah perjalanan panjang yang ide pengembangannya dimulai pada tahun 1999 di sebuah kota bernama Kisaran dan sebuah kamar kost di sekitar Tawakal, Grogol, Jakarta. Buku yang juga dapat dikatakan sekelumit jawaban dari sebuah perjalanan memecahkan sebuah nilai kekecewaan kehidupan. Buku yang menjadi simbol awal dari perjalanan mencapai sebuah impian yang masih panjang.

Projek yang semula diciptakan dengan kode “smiletown”, sebuah projek yang diciptakan untuk membuat seluruh anak-anak di Sydney dapat tidur dengan tenang dan dekat dengan orang tuanya, walaupun orang tuanya tidak ada disebelahnya karena kesibukannya.

Projek ini telah berubah bentuk, yang semula merupakan sebuah projek pengembangan adopsi struktur organisasi menjadi implementasi pengembangan sebuah platform terpadu. Diharapkan melalui buku ini, jiwa dari pengembangan dan kemandirian dapat tumbuh setelah membacanya. Yang terpenting dari penggeraan sebuah projek BlueOxygen ini adalah bukan kode atau hasilnya tetapi sebuah nilai perjuangan dan komitmen pengembangan serta interaksi dengan pihak lain. Bukti bahwa kejujuran dan kerja keras tidaklah cukup.

Buku ini sarat berhubungan dengan projek BlueOxygen, yang sebenarnya adalah sebuah novel perjalanan berpetualang dalam

mempelajari teknologi Java, *branding*, dan juga salah satu bentuk implementasi, bagaimana teori aktivitas dapat diimplementasikan didalamnya.

Buku ini juga sebenarnya adalah gambaran solusi, bagaimana seorang manusia berusaha bertahan didunia persaingan yang semakin sengit dari tahun ke tahun, dimana dunia IT adalah dunia yang sepersekian dari sebuah kebutuhan operasional, tetapi sebuah modal yang diberikan Tuhan yang paling murah bagi mereka yang tidak mampu untuk mencapai impianinya, sebuah sarana impian yang dapat dilakukan dengan hanya bermodal niat. Walaupun sampai hari ini penulis belum tahu, apakah projek BlueOxygen ini adalah media yang sebenarnya untuk mencapai impian itu.

Projek BlueOxygen juga salah satu bentuk solusi yang membuat penulis dapat bertahan didunia IT di Indonesia, salah satu sarana yang membantu publikasi penulis sehingga dapat bertemu dengan orang-orang yang umumnya hanya dapat dibaca namanya dimedia masa. Sarana yang memungkinkan penulis dapat menjadi *counterpart* yang satu *level* dengan para vendor, yang umumnya meminta sebuah kontribusi dengan nilai tertentu atau nilai investasi yang tidak sedikit, nilai yang mahal dan harus diberikan, hanya untuk memulai sebuah hubungan atau berekanan.

Buku ini dipersembahkan kepada semua pihak, dan diharapkan setelah membaca buku ini, walaupun isinya mungkin terlalu teknis untuk beberapa pihak. Diharapkan buku ini dapat menjadi sumber inspirasi, dan juga menjadi fondasi bagi siapapun mengenai pentingnya platform. Sumber inspirasi pentingnya arti sebuah sustainabilitas. Buku ini juga diharapkan menjadi sebuah contoh pentingnya swasembada teknologi untuk sebuah negara, baik jangka pendek maupun jangka panjang.

Terima kasih diucapkan kepada semua pihak, mulai dari Peter Manzo, Tony Haryanto, para investor Intercitra, rekan-rekan JUG

Indonesia, serta mereka yang telah dan masih menjadi anggota keluarga Intercitra, juga kepada mahasiswa magang di VDEC serta beberapa kampus yang sempat melakukan magang yang telah berkontribusi dalam bentuk kode atau testing level dalam semua projek BlueOxygen ini. Tanpa kalian semua, projek ini hanyalah sebuah projek personal.

Terima kasih kepada tim IGOS di Ristek maupun di Depkominfo, Richard Higgins, dan Geoff Zeis. CIDA yang akhirnya mau mensponsori percetakan buku ini. Kepada Sun Microsystem yang telah menciptakan Java, yang merupakan teknologi yang bukan hanya elegan tetapi juga telah merubah banyak hidup orang.

Ucapan khusus diberikan kepada papa dan mama di Sukabumi, banyak kata-kata tidak terucapkan yang merupakan misteri kehidupan yang mungkin sampai hari ini belum dipecahkan penulis, menjadi sumber inspirasi apa artinya sebuah hidup, dan mungkin buku ini adalah salah satu bentuknya, walaupun mungkin bukan media yang cocok untuk mengatakannya.

Juga kepada Endy Muhardin yang berkenan berkontribusi untuk bab Spring.

Jakarta, November 2006

Frans Thamura
frans@blueoxygen.org

Daftar Isi

Sekilas Mengenai Buku dan Penulis	iv
Kata Sambutan.....	v
Kata Pengantar	vii
Daftar Isi	x
Bab 1, Mekanisme Pengembangan dalam Java	
dari MVC sampai AJAX	1
- Politik Sekitar MVC	3
- Teknologi IoC	7
- MVC dalam Solusi Java untuk Web	10
- Dari MVC ke WARS, dilanjutkan dengan AJAX.....	12
- Jenis-Jenis MVC	13
- AJAX dan MVC	14
- Kesimpulan.....	18
Bab 2, Berkenalan dengan Teknologi Viewer dalam MVC	
Compile Time vs Run Time.....	19
JSP, Servlet dan Turunannya.....	23
Bekerja dengan Velocity	24
JasperReport dan iReport.....	30
JFreeChart dan Ceworlf	33
Mondrian dari Pentaho.....	33
Bab 3, Berkenalan dengan Hibernate untuk Solusi Model pada MVC	
37	

Berkenalan dengan Hibernate Tools	39
Membuat Object Mapping Pertama	43
Menjalankan Objek Hibernate	51
ORM dengan Relationship.....	53
Implementasi DBUnit sebagai ETL.....	60
Pengembangan Hibernate Lanjutan	63

Bab 4, Pemograman MVC dengan Controller

WebWork/Struts 2	65
-------------------------------	-----------

Sekilas Mengenai WebWork/Struts2.....	66
Bagaimana WebWork Bekerja	67
Membuat Aplikasi MVC dengan Result JasperReport	73
Membuat Aplikasi MVC dengan Result Chart menggunakan JFreeChart	74
Implementasi Validasi pada WebWork - Standard	75
WebWork Lanjutan - Interceptor	77

Bab 5, Pengembangan dengan IoC Menggunakan

Spring Framework.....	80
------------------------------	-----------

Mengapa IoC Penting dalam Aplikasi?.....	80
Sekilas Tentang SpringIDE	92
Menginstall SpringIDE	93
Bekerja dengan SpringIDE	93
Hubungan Hierarki dalam SpringIDE	100
Bekerja dengan XML Editor (Exadel)	103

Bab 6, Membuat Aplikasi Web CRUD

dengan Cimande.....	107
----------------------------	------------

Sekilas Tentang Projek Cimande.	108
Logika CRUD Mengacu pada Skema Cimande	110

Bab 7, Implementasi User Management dengan Filter dan Session	121
Memilih Session atau Cookies?	121
Berkenalan dengan Filter	122
Membuat Aplikasi dengan Filter dan Session.....	126
Kesimpulan	130
Bab 8, Manajemen Organisasi dengan BlueOxygen Cimande	131
User Management pada Cimande	134
Module Management Cimande.....	135
Mekanisme Workspace Explorer	137
Bekerja dengan Descriptor.....	138
Membuat Workspace Explorer Sendiri	138
Membuat Role.....	141
Membuat Site	141
Membuat Role-Site	141
Bab 9, BlueOxygen Postila, Solusi POS dengan Swing dan Hibernate	149
Sekilas Mengenai JavaPOS.....	149
Sekilas Tentang BlueOxygen Postila	152
Postila dan Netbeans	154
SessionFactory dalam Postila	159
Action pada Swing	160
Bab 10, Mengembangkan Aplikasi Berbasis SOA dan Memperkenalkan PASIR	166
Object Wrapper	173
Mengembangkan Web Services Sendiri.....	176

Mengembangkan SOAP dengan Eclipse Calisto	178
PASIR, SOA Versi Indonesia	189

Lampiran A, Berkenalan dengan Java SDK dan variannya 191

Sun Java SDK	191
Instalasi Sun Java SDK	192
Instalasi Sun Java SDK di Linux	193
Bea JRockIt	194
Instalasi Bea JRockIt di Linux	196
IBM Java SDK	196
Jikes	197

Lampiran B, Memulai Pemogramana Berbasis Java dengan Eclipse IDE 199

Sekilas Mengenai Eclipse	199
Rich Client Technology	202
Menginstall Eclipse IDE	204
Memulai Bekerja dengan Eclipse	207
Memulai Projek Java	209
Membuat Aplikasi Java Pertama	215
Mendebug Aplikasi Java	219
Memulai Projek WebWork (Metode 1) menggunakan Eclipse IDE Standar	224
Memulai Projek WebWork (Metode 2) menggunakan Eclipse Calisto	227

BAB I

Mekanisme Pengembangan dalam Java dari MVC sampai AJAX

Java memang tidak lepas dari kata Sun, sang penciptanya. Berkat Sun pula dan diikuti dengan gelombang Free Software paling hot abad ini yaitu Linux, telah membuat Java yang merupakan teknologi terbuka didunia pemrograman bertransformasi secara sangat cepat.

Java yang semula diciptakan untuk membuat aplikasi yang berjalan di browser, tetapi karena performancenya sangat jelek, telah berevolusi menjadi sebuah solusi untuk server. Dimana bundle semua teknologi berbasis Java ini lebih sering disebut J2EE, singkatan dari Java 2 Enterprise Edition. Resmi tahun 2005, Sun mengganti kata J2EE menjadi Java EE, karena tidak lah bagus Java terus menerus versi 2, setelah lebih dari satu dekade menyandang kata itu. Padahal sebenarnya saat ini Java telah memasuki versi 5.0, sehingga *bundle* Java untuk solusi yang lebih kompleks juga berganti nama menjadi Java EE 5.0

Implementasi Java EE ini ternyata sangat beragam, karena didalam spesifikasi Java EE terdapat banyak sekali teknologi yang siap pakai seperti servlet, JSP, JSF, EJB, JMS atau JCA. Java EE adalah merek dagang dari Sun Microsystems, sehingga untuk informasi lebih lanjut mengenai Java EE, dapat mengunjungi websitenya di <http://java.sun.com>.

Implementasi dilapangan Java EE memerlukan *container* atau middleware, sehingga setiap objek-objek yang dikembangkan dapat dijalankan didalam container tersebut, termasuk juga pengelolaan objek dilakukan oleh *container* tersebut. Beberapa

implementasi container Java EE adalah JBoss, Jonas, Weblogic, Websphere atau Glassfish. Umumnya Java EE yang disebutkan adalah *full stack* Java EE container.

Ternyata banyak sekali kebutuhan yang tidak memerlukan fitur-fitur Java EE yang sangat banyak tersebut, yang mana umumnya memerlukan hanya fitur yang berhubungan dengan Web, teknologi Java EE yang hanya berurusan dengan Web ini disebut servlet, singkatan dari server applet. Sebuah *container* Java EE yang hanya berisikan servlet ini disebut *servlet container*, implementasi dilapangannya adalah Tomcat dari Apache, Jetty, Resin dari Cauchy, atau yang commercial Websphere Express, JRun dari Adobe. Geronimo dari Apache atau JBoss AS menggunakan Tomcat sebagai *servlet container*nya. Sehingga, setiap aplikasi yang dikembangkan di atas Tomcat, umumnya dapat berjalan tanpa modifikasi bilamana dideploy di Geronimo atau JBoss AS.

Turunan teknologi servlet ini ada bermacam-macam, tetapi yang distandarisasikan adalah JSP dan JSF. Sedangkan yang dibahas dibuku akan membahas turunan dari teknologi servlet ini yang tidak distandarisasi di JCP, dimana penulis merasakan kehebatan teknologi non JCP ini untuk pengembangan.

Penulis merasa perlu menjelaskan teknologi turunan ini, karena teknologi tersebut terutama Velocity, telah bertransformasi dari sekedar teknologi untuk web menjadi teknologi *code generator*, yang artinya kode yang kita kembangkan akan menghasilkan kode baru baru lagi, atau dengan kata lain aplikasi yang menghasilkan aplikasi lain.

Walaupun mekanisme tersebut tidak dibahas dalam buku ini, mekanisme ini akan menjadi salah satu teknologi unggulan. Dimana mekanisme ini akan menjadi salah satu fitur berikutnya.

Politik Sekitar MVC

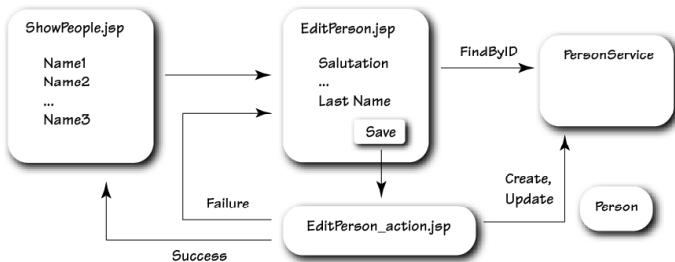
Dengan semakin kompleksnya dan naiknya popularitas servlet ini, dan juga merupakan salah satu teknologi yang menjadi jembatan antara JavaBean dengan Enterprise JavaBean atau teknologi lainnya didalam Java EE, seperti JTA, JCA, atau komponen didalam Java SE.

Hal ini juga berhubungan dengan teknologi Java yang mirip dengan Smalltalk yang dalam era tahun 80-an sangat populer, terutama teknologi MVCnya, membuat dunia Java juga dipenuhi banyak teknologi manajemen pengembangan yang mengimplementasikan servlet, artinya sebuah teknologi berbasis MVC untuk web.

Apalagi setelah tim Smalltalk yang sekarang semuanya bermarkas di IBM, membuat Eclipse, dengan teknologi SWT yang merupakan turunan dari Smalltalk, telah membuat Java dan Smalltalk menjadi satu. Tentu saja primadona Smalltalk masuk kedalam Java.

Sebenarnya sebelum MVC muncul, dan saat *web programming* meledak, yang dipelopori oleh Microsoft dengan ASPnya, kemudian disusul oleh PHP yang multiplatform dan sangat cepat, membuat tim Sun membuat sebuah teknologi JSP, yang mana bekerja mirip dengan teknologi *web scripting* ASP atau PHP, tetapi membawa sifat awal dari Java, yaitu OOP.

JSP yang merupakan turunan dari servlet, memungkinkan sebuah HTML diberi kode Java dan berinteraksi dengan object didalam container Java EE. Metode ini dikenal dengan pemograman model 1, yang mana dalam dunia nyatanya, karena JSP setiap kali dieksekusi harus dirubah menjadi sebuah class Java, membuat JSP dianggap *the dark side of Java* didunia *Web programming*.



Pengembangan Java Model 1 dengan JSP

Dengan sifat JSP yang dianggap buruk, dan membuat sampah didalam *container* Java EE, sedangkan implementasi pemogramaman berbasis Web dengan servlet adalah sangat rumit dan tidak semudah kompetitornya yaitu ASP dan PHP. Membuat programmer Java harus berpikir keras untuk membuat Java menjadi sebuah teknologi yang layak dipakai didunia Web.

Akhirnya tepatnya sekitar tahun 1996-an, muncul sebuah projek yang mengacu pada model 2 yaitu MVC, model yang getol dipopulerkan oleh tim Smalltalk, jauh sebelum Java lahir, tepatnya sekitar tahun 1988.

Tahun 1996-an, lebih tepat diakhir tahun 2000, Apache yang saat itu sedang mulai melakukan hosting projek Java, dengan bendera Java, tetapi harus mengganti menjadi Jakarta, karena kasus merek dagang. Memiliki sebuah subprojek baru yang bernama Struts.

Struts yang berupa *controller* memerlukan JSP sebagai *presentation layer*nya. Struts bekerja dengan EJB untuk modelnya, sehingga dalam format MVC, M diisi oleh EJB, C diisi oleh Struts, dan V diisi oleh JSP.

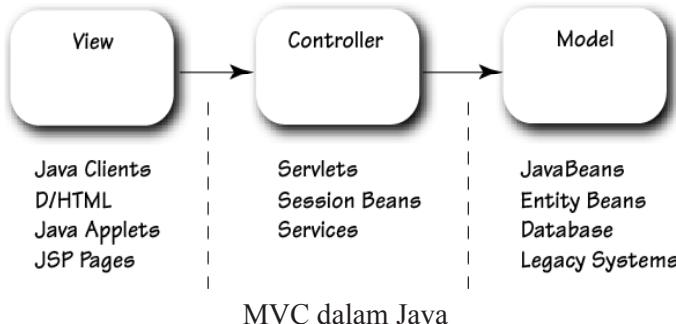
Walaupun setelah itu Apache mengeluarkan teknologi yang merupakan jawaban dari dunia hitam JSP dengan nama Velocity yang akan dibahas dibab berikutnya, serta beberapa teknologi

untuk mengganti EJB khusus untuk non *distributed computing*, seperti iBatis. Bersamaa dengan itu, searah dengan ledakan Java didunia. Lahirlah gerakan *framework* yang merupakan sebuah nilai tambah terhadap servlet atau lebih tepatnya Java EE.

Struts adalah projek MVC Java pertama yang secara langsung telah merubah peta pasar Java di dunia. Adopsi Struts hampir disemua solusi berbasis Java, mulai dari perusahaan kecil sampai bank-bank kelas dunia mengadopsinya.

Pendekatan Struts yang implementasinya mirip dengan EJB, memerlukan banyak class untuk sebuah eksekusi aplikasi Web, telah membuat seorang jenius didunia Java bernama Richard Oberg mengeluarkan WebWork, dengan konsep HMVC, Hierarchical MVC. Richard Oberg ini adalah salah satu dari orang yang mewarnai dunia Java dengan teknologi populer seperti JBoss dan Xdoclet. Yang terakhir, Xdoclet, merupakan *hacking tools* paling populer didunia Java. Yang memungkinkan teknologi Java dapat dibuat menjadi apa saja, dengan hanya memanage sebuah *comment* didalam class.

Sayangnya Struts yang diciptakan Craig, ditinggalkan sang empunya tanpa sempat mentransfer ide beriliannya, Craig akhirnya mengembangkan sebuah spesifikasi standard berbasis MVC juga dengan nama JavaServer Faces. Yang saat ini sedang getol dipromosikan oleh Oracle, Sun dan JBoss. Dimana mereka semua adalah leader dari spesifikasi ini,



Perang Struts vs WebWork ini adalah perang Java paling menarik, karena WebWork 2.3 diganti namanya menjadi Struts 2.0. Kok bisa? Padahal secara projek mereka tidak berhubungan. Bab berikutnya akan membahas apa itu WebWork.

Teknologi MVC ini sebenarnya tidak hanya ada didunia web atau Java EE saja, sebenarnya didalam Java standard yang lebih dikenal dengan Java SE, telah ada teknologi berbasis MVC, yaitu Swing. Teknologi ini konon diciptakan oleh tim IBM, yang notabene merupakan perusahan pelopor MVC, karena akusisi Smalltalk tadi.

Swing yang berjalan diteknologi AWT, merupakan MVC juga, hampir semua objeknya seperti JTable adalah MVC. Malah beberapa sumber mengatakan hal ini terjadi karena IBM sangat yang terdapat tim Smalltalk didalamnya, telah berkontribusi lebih dari 80% terhadap Swing. Ini yang menjadi alasan mengapa Swing sangat MVC.

Jadi bilamana ingin jadi programmer Java, kata MVC adalah hal biasa. Malahan didunia Java yang terkenal dengan solusi yang selalu memikirkan arsitektur dan kekuatan *container*, MVC telah memiliki banyak turunan, yang berbentuk pola yang lebih sering disebut dengan *pattern*.

Buku ini sebenarnya tidak mendalami Java EE lebih lengkap, tetapi lebih condong penekanan pada MVC dan turunannya, serta tools pendukung yang memungkinkan pengembangan aplikasi MVC lebih efektif.

Teknologi IOC

Struts, Webwork, yang kemudian diikuti oleh Tapestry, JSF, Stripes, akhirnya muncul SpringMVC. Semua ini adalah teknologi yang memungkinkan pemograman memisahkan antara data, business logic dan presentation layer dalam sebuah pola standar bernama MVC.

Malah SpringMVC yang berdiri diatas IOC (*Injection of Control*) atau DI (*Dependency Injection*), telah membuat framework-framework MVC ini dapat saling dipertukarkan. Sebagai contoh sebuah solusi menggunakan model EJB3, kemudian diganti dengan Hibernate saja, atau solusi lain dengan TopLink. Hal ini memungkinkan dengan IOC.

IOC merupakan sebuah mekanisme yang secara awam adalah sebuah mekanisme memanggil sebuah objek tetapi tanpa inisialisasi.

Revolusi IOC ini yang semula diciptakan karena EJB yang memang diciptakan untuk komputasi terdistribusi, dan tidak cocok untuk projek yang berbudget kecil, bertransformasi menjadi sebuah layer baru dalam pemograman, terutama sebagai perekat antara model, *viewer* dan *controler* dalam implementasi aplikasi berbasis MVC didunia Java khususnya.

Teknologi IOC sebenarnya telah lama ada di dunia Open Source Java, tetapi balik lagi, hanya beberapa yang bertahan, diantaranya Avalon dari Apache yang *discountinued*, terus ada JBoss Micro Kernel yang merupakan engine inti dari JBoss. Buku ini hanya membahas Spring.

IOC ini sebenarnya menarik diikuti, dan untuk beberapa kasus sangat bagus digunakan dalam solusi, terutama mengisi interkoneksi antara M, V dan C dalam pemograman berbasis MVC. Yang mana dalam buku ini IOC lebih ditekankan pada integrasi antara WebWork dan Hibernate yang akan dibahas di Bab 5 sampai akhir buku ini.

Definisi IOC, untuk lebih lengkap dapat mengunjungi <http://martinfowler.com/articles/injection.html>.

Buku ini memposisikan IOC sebagai sebuah solusi yang menarik, terutama bagi mereka yang menjalankan solusi berbasis Cimande pada JBoss. Artinya dapat dikatakan menggunakan dua teknologi IOC yaitu sebagai wrapper antara Controller dan Model dalam MVC, serta sebagai container engine pada JBoss.

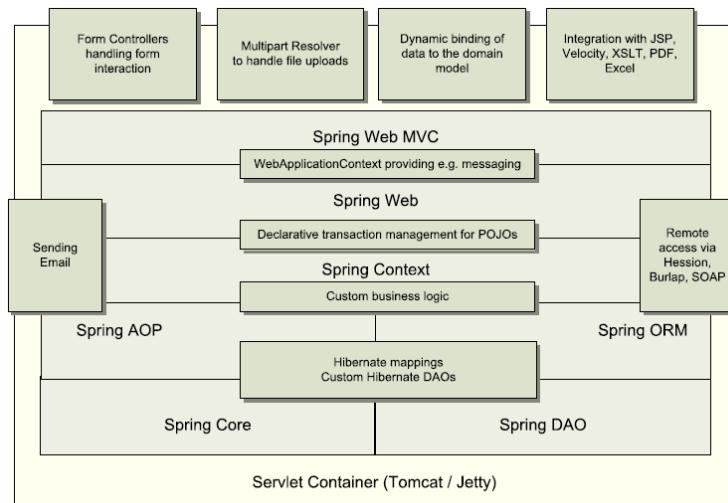
Reposisi ini, apalagi setelah JBoss sangat aktif untuk menjadi pemain utama dalam industri EJB3, yang mana EJB3, terkesan sangat Hibernate, walaupun sebenarnya untuk mengembangkan EJB3 dapat menggunakan saingan Hibernate seperti Toplink dari Oracle, atau Kodo dari Bea. Yang semuanya Open Source, kecuali TopLink comercial.

Dipasaran terjadi persaingan antara Spring melawan JBoss, sebab beberapa berita dari tim Spring, yang berhasil mengembangkan sebuah solusi yang lebih sering disebut dengan *light development* of Java, telah membuat Spring + Tomcat adalah solusi yang sangat ideal untuk production.

Sedangkan JBoss yang menggunakan JBoss Microkernel sebagai IOC engine, yang merupakan engine didalam JBoss AS dan juga wrapper solusi antara JSF dan Hibernate, dalam produk mereka bernama JBoss Seam.

Membuat posisi JBoss AS + JBoss Seam mirip dengan Spring stack. Dimana buku yang membahas Spring MVC dan JBoss Seam akan dikeluarkan setelah buku ini direlease.

Yang lebih menarik lagi, buku ini adalah akan menjelaskan versi yang berbeda dengan posisi dari perang JBoss dengan Spring.



Arsitektur Spring

Buku ini sebenarnya lebih menekankan integrasi *framework* MVC antara WebWork dengan Hibernate menggunakan Spring. Dimana Seam menggunakan JSF dan Hibernate. Sedangkan Alfresco menggunakan JSF, Hibernate dengan Spring.

Semua merupakan kombinasi yang berbeda, dan tinggal pembaca menentukan solusi mana yang lebih sesuai.

Sebenarnya ke-3 pilihan solusi ini dapat diintegrasikan, yaitu dengan merubahnya menjadi solusi berbasis SOA, yang mana akan menjadi topik bahasan pada bab 9.

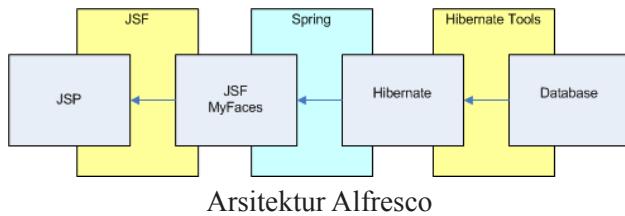
Reposisi Spring yang berawal dari sebuah IOC (Spring Bean) menjadi sebuah solusi yang lengkap membuat Tomcat + Spring memiliki teknologi yang alternatif dari JBoss AS. Padahal teknologi Spring ini berjalan diatas JBoss.

Bentuk persaingan ini mirip dengan persaingan Windows dengan Java, yang saat ini terjadi. Kita tahu Java berjalan diatas Windows.

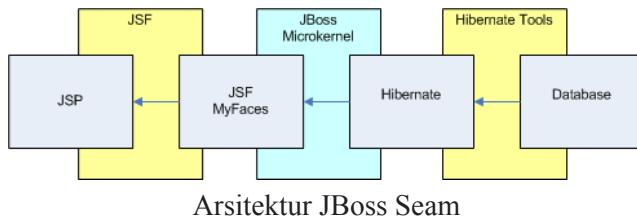
MVC dalam Solusi Java untuk Web

Dalam pengembangan MVC, sebenarnya kita dapat melakukan kombinasi berbagai teknologi untuk membuat solusi berbasis MVC. Jadi jangan berharap dengan pakai Struts saja kita dapat mengimplementasikan MVC.

Alfresco (<http://www.alfresco.org>), merupakan solusi digital library yang mengimplementasikan MyFaces, Spring, Hibernate, digabung dengan teknologi JCR untuk penyimpanan file, Lucense untuk search engine.

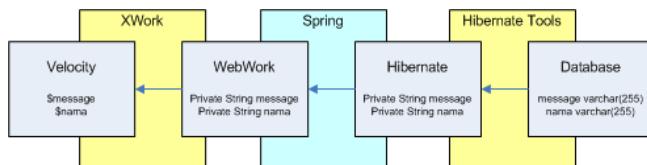


JBoss Seam dari JBoss, merupakan solusi fondasi Web 2.0, yang mengintegrasikan MyFaces, Hibernate, dan JBoss Micro Kernel sebagai IOCnya.



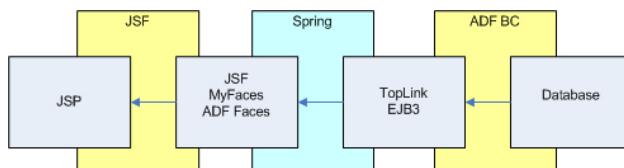
Arsitektur JBoss Seam

Sedangkan produk dalam negeri Indonesia, BlueOxygen Cimande, menggunakan WebWork, Spring dan Hibernate sebagai kombinasi MVCnya.



Arsitektur BlueOxygen Cimande

Sedangkan dari Oracle, setelah Mike Keith, sang pencipta TopLink bekerja sama dengan Spring, teknologi MVC Oracle juga mengadopsi *stack* serupa Alfresco.



Arsitektur Oracle ADF (Fussion Middleware)

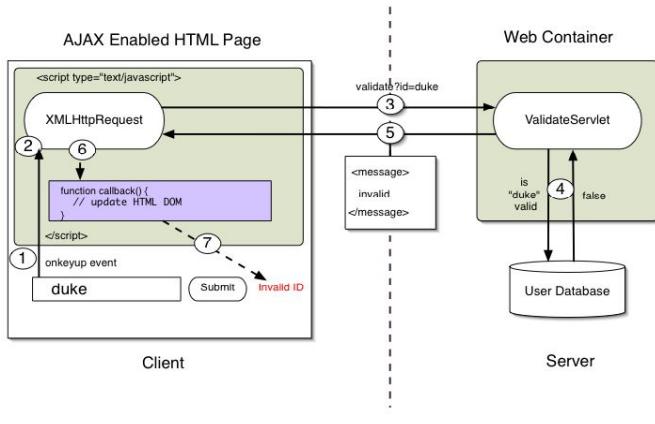
Seperti yang dijelaskan disebab sebelumnya, ke-4 teknologi ini adalah memiliki kerangka awal yang sama, tetapi memberikan kombinasi yang berbeda.

Malah reposisi Spring yang menyaingi JBoss Micro Kernel, memungkinkan Cimande suatu hari bersaing dengan JBoss AS.

Apakah itu memungkinkan?

Dari MVC ke WARS, dilanjutkan dengan AJAX

Tahun 2005, MVC telah berkembang dengan lahirnya produk Open Source untuk *workflow*, seperti Shark dari Enhyrda, JBPM dari JBoss, serta teknologi *rule management* seperti Mandarax, Drools (JBoss), serta diadopsinya teknologi *expert system* kedalam Java, yang memungkinkan setiap MVC bukan hanya mengakses model data saja, tetapi telah memerlukan sebuah interupsi tambahan seperti status pekerjaan dalam *workflow* repository, serta memungkinkannya setiap proses harus melalui peraturan yang disisipkan dalam kode kita, menggunakan *rule script*.



Arsitektur AJAX

Yang lebih hebatnya, *workflow* dan *rule script*nya ini dapat berubah dan bekerja terpisah secara dinamis diluar MVC.

Mekanisme ini pernah dicetuskan dengan sebutan WARS singkatan dari Workflow, Action, Result and State, yang sebenarnya sebuah nama baru dari MVC+W. Maklum orang Java paling suka nama baru, konon biar keren dan setiap yang keren-

keren ini memiliki daya jual lebih. Jadi kalau mendengar kata WARS, akan terasa berbeda, sebenarnya itu adalah MVC yang didalam memiliki teknologi *workflow*.

Mekanisme yang sama terjadi dalam dunia Javascript, yang mana setiap interaksi dengan server dengan XMLHttpRequest, disebut AJAX.

Jenis-Jenis MVC

Kembali ke tahun 2000-an, sebenarnya perang MVC terjadi antara Struts dengan WebWork, yang mana WebWork lebih mengutamakan kemudahan, dengan implementasi teknologi dispatcher, sedangkan Struts yang bernaung di Apache, yang mana Apache merupakan nama yang paling hot dalam dunia Open Source, telah membuat Struts menjadi *framework* untuk MVC paling populer saat itu. Secara marketing, terlihat WebWork yang mengatakan dirinya pull MVC, sedangkan Struts adalah push MVC, tampak terlihat berbeda. Merger WebWork dengan Struts ditahun 2005, telah membuat kombinasi yang menarik sekali.

Saat itu, orang beranggapan Struts dan WebWork adalah MVC, tetapi dengan berkembangnya waktu, dan semakin banyak rekan-rekan kita diseluruh dunia yang membuat *framework* yang mungkin saja berbasis pada *framework* yang sudah ada, tetapi dengan tambahan fitur, seperti setelah Struts muncul JSF, sedangkan setelah WebWork lahir Stripes. Stripes, walaupun belum sehandal WebWork, terlihat mengimplementasikan *annotation* didalamnya, yang konon tim WebWork sedang mencari cara bagaimana mengimplementasikan *annotation* yang mudah didalamnya.

Teknologi mirip MVC yang khusus untuk XML adalah Cocoon yang juga dari Apache, dilanjutkan dengan Turbine, lalu lahir lagi komponen sejenis seperti Tapestry.

Alhasil penulis mendapatkan untuk memudahkan pemilihan

MVC, diperlukan pemisahan antar MVC, baik MVC yang berbasis *component* atau MVC yang berbasis *action*.

MVC tipe lain adalah yang berbasis *component* adalah yang bekerja seperti halnya pemograman *event driven*, setiap tag memungkinkan dibuat komponennya, teknologi yang sangat serius dengan model ini adalah JSF. Yang mana implementasi JSF ini sebenarnya merupakan adopsi teknologi Swing kedalam teknologi berbasis Web.

Sedangkan teknologi MVC berbasis *action*, adalah adopsi implementasi yang mengacu pada *request* dan *responsenya* teknologi HTML, teknologi ini dilead oleh WebWork, yang mana setiap pengembangan mengadopsi page-page dari HTML.

Tahun-tahun kedepan sepertinya akan terjadi integrasi antara framework ini, seperti Struts 2.0 yang merupakan projek integrasi dari Struts dan WebWork, ternyata mulai memasukan unsur komponen kedalamnya, dengan memasukan JSF sebagai komponennya.

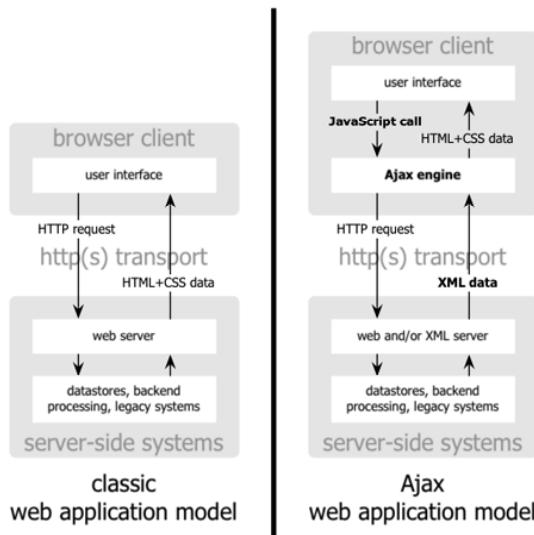
Jadi dapat dikatakan, tahun-tahun kedepan akan lahir MVC yang mengadopsi kedua teknologi *action* dan *component*. Ini tentu saja evolusi MVC baru lagi, yang selama lebih dari 10 tahun yang dilakukan komunitas Open Source.

Dari semua ini yang menarik adalah teknologi web yang tidak Open Source malahan tidak dapat bertahan, seperti WebObject dari Apple, merupakan teknologi Java berbasis komponen yang sangat bagus, tetapi sayang karena tidak Open Source dan berjalan diatas Mac saja, serta focus Apple yang bukan di Java, membuat produk ini tenggelam.

AJAX dan MVC

Searah dengan semakin cepatnya Internet, serta keterbatasan

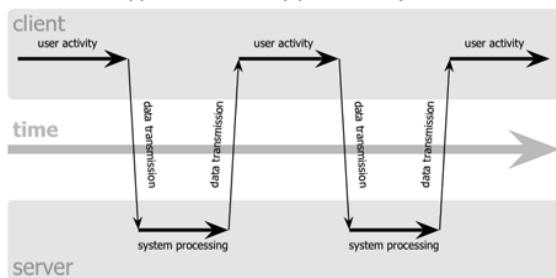
HTML untuk memungkinkan terjadi sebuah aplikasi berbasis Web yang interaktif. Lahirlah sebuah teknologi bernama AJAX, yang mana populer setelah Google meluncurkan Gmail, yang merupakan *free email* yang bekerja sangat cepat.



Pengembangan Web Classic vs AJAX

AJAX yang merupakan singkatan dari *Asynchronous Javascript and XML*, merupakan sebuah mekanisme yang memungkinkan proses Web dipisah antara server dan *client*, ini disebabkan PC client yang diasumsikan sudah terlalu cepat, dan mubajir bilamana kita menggunakan teknologi berbasis MVC sekalipun. Karena ternyata dengan menggunakan AJAX, sebuah penekanan beban server dari 30% sampai 60% dapat terjadi pada server yang sama.

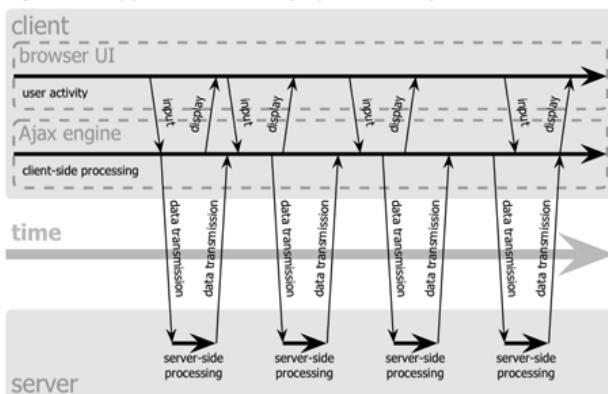
classic web application model (synchronous)



Metode Request dan Response aplikasi Classic / Non AJAX

Yang lebih hebat lagi, AJAX memungkinkan kita mengakses beberapa data resource tanpa perlu melakukan *refresh page* didalam *browser*, yang tentu saja ini merupakan teknologi interaktif yang bagus sekali, terutama untuk mereka yang memerlukan analisa dan proses yang cepat.

Ajax web application model (asynchronous)

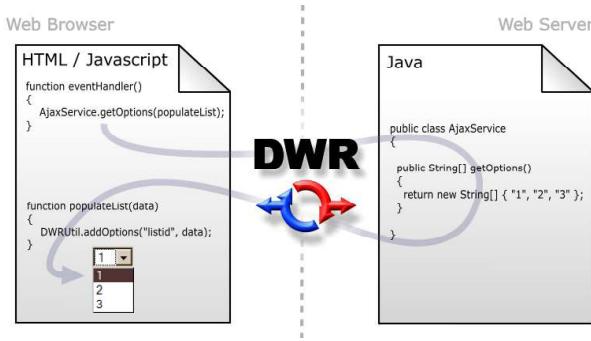


Metode Request dan Response AJAX

Kehebatan AJAX ini terus berkembang, yang malahan dipercaya merupakan teknologi yang akan memungkinkan kita dapat bekerja untuk melakukan pekerjaan harian perkantoran tanpa

perlu menginstall aplikasi Office Automation seperti OpenOffice atau Microsoft Office.

Untuk membuat aplikasi AJAX sebenarnya tidaklah sulit, karena komunitas AJAX telah banyak membuat sebuah *wrapper* atau *script library* yang siap pakai, seperti Yahoo dengan Yahoo UI, Google dengan GWT, DojoToolkit, atau JSON, ataupun DWR buatan Joe Walker yang memungkinkan object Java dapat berinteraksi dengan *object Javascript*.



Mekanisme Pengembangan AJAX dengan DWR

Alhasil dari semua evolusi internet ini, dalam waktu dekat akan muncul duo MVC, yaitu MVC diserver serta MVC untuk pemorsesan XML dengan AJAX.

Dengan merger antara MVC berbasis komponen dan action, serta integrasi MVC baru ini kedalam teknologi AJAX, membuat sebuah pemograman semakin kompleks, interaktif dan lebih mudah didebug.

Penulis sedang mencoba mengembangkan teknologi duo MVC ini dalam project dibawah Cimande dengan nama Cimande Thin, yang mengutilisasi DWR serta fitur WebWork untuk validasi berbasis AJAX. Diharapkan dalam waktu dekat akan muncul

result management dari WebWork yang berbentuk AJAX. Saat ini, fitur ini sedang dikembangkan oleh tim DWR untuk DWR 2.0.

Jadi apakah anda akan berpartisipasi sebelum ini terealisasi atau menunggu semua ini selesai?

Kesimpulan

Penulis melihat dengan munculnya teknologi yang bermacam-macam dengan nama yang aneh-aneh, sebenarnya merupakan sebuah bentuk evolusi dari mekanisme pengembangan *dilayer* setelah *middleware*, dimana untuk *middleware*, kita hanya perlu mengikuti standar Java EE, semua sudah selesai.

JSF merupakan barang baru dan sedang dalam pembuktian kehandalan, bilamana bila dikombinasikan EJB3, telah membuat kharisma Java EE yang semua diinjak-injak oleh para programmer underground pengikut aliran *light* dapat muncul kembali.

Apalagi persaingan EJB3 dengan teknologi IOC yang mulai terjadi membuat solusi stack Java EE menjadi agak memusingkan bagi orang awam untuk diikuti.

Masuknya AJAX, yang membuat teknologi Java yang dianggap rakus resource, menjadi dapat ditekan *performancenya*, menjadi sampai hanya tinggal 1/3 saja, dan hebatnya dari AJAX ini, langsung diterima oleh semua pihak yang berseteru.

Buku ini sebenarnya akan menjelaskan solusi Java untuk solusi *enterprise* yang lain, yang mana, diharapkan dapat memberikan masukan bagi para pembaca untuk mengkaji, serta memahami, mengapa banyak pihak yang protes karena JSF itu sulit, EJB dianggap sebagai teknologi bagi mereka yang memiliki uang lebih untuk dihamburkan, karena biaya pengembangannya terlalu tinggi dibandingkan solusi lainnya.

BAB II

Berkenalan dengan Teknologi Viewer dalam MVC

Compile Time vs Run Time

Didunia MVC, Viewer yang merupakan bagian khusus untuk menangani *layer presentation* memiliki beberapa teknologi seperti JSP yang standar dan Velocity dari Apache (<http://jakarta.apache.org>), serta turunan Velocity yang lebih lengkap dan rich yaitu Freemarker (<http://www.freemarker.org>). Semuanya adalah turunan dari servlet.

JSP yang terkenal dengan sebutan *the darkside of Java*, adalah sebuah teknologi yang bekerja dengan cara mengkompilasi setiap *script* JSP menjadi *class*. Metode ini disebut dengan *compile time technology*.

Teknologi ini kelebihannya adalah langsung jalan tanpa perlu dikompilasi, tetapi tentu saja kelemahannya langsung terlihat yaitu membuat sampah. Bilamana kita membuat sebuah file, misalnya index.jsp akan menghasilkan 2 file untuk dapat dijalankan yaitu index.java yang merupakan turunan dari servlet dan index.class, binary sebenarnya yang dijalankan. Dimana index.java adalah *source code* dari hasil *mapping* index.jsp kedalam standar servlet.

Agara sampah tidak terlalu besar dan membuat performance server menurun, ada cara untuk mengatasi kelemahan JSP tersebut, disarankan mengimplementasikan taglib, sehingga *tag-tag* didalam JSP telah terlebih dahulu *dicompile* menjadi *bytecode*, sehingga proses didalam JSP menjadi lebih cepat.

Sayangnya taglib ini membuatnya tidak semudah yang dibayangkan, dan merupakan teknologi yang kurang populer. Beberapa taglib yang standar seperti JSTL telah dikeluarkan untuk memanipulasi *expression* didalam JSP, sehingga pemrograman menjadi lebih berstruktur. Tetapi saja, metode ini tidak semua dan tidak *seflexible* melakukan pemrograman langsung di JSPnya.

Apalagi di Indonesia, yang projek terkadang diserahkan langsung kepada programmer *junior* atau *project manager* yang tidak mengerti arsitektur Java secara keseluruhan, membuat programmer yang sudah merasa enak dengan JSP, langsung saja buat didalam JSP tersebut. Alhasil terjadi sebuah pemrograman yang bukan saja kotor tetapi sangat sulit *trace*. Walaupun Eclipse WTP sejak versi 1.5 memungkinkan mendebug JSP, tetapi saja dalam implementasinya JSP harus *convert* menjadi servlet. Error akan muncul bilamana menggunakan *include* didalam JSP pada WTP.

Ada pendekatan terbaru dari JSP yang dikembangkan oleh Craig sang pembuat Struts, dengan teknologi JSF, yang memungkinkan implementasi menggunakan JSP lebih rapi.

JSF ini dianggap kartu as dari para programmer pro JCP, karena JSF ini yang dikembangkan oleh juga Oracle, telah menjadi barang *marketing* dunia Java yang konon lebih *flexible*. Tentu saja ini strategy untuk melawan komunitas Open Source yang telah lebih dahulu berkuasa, yaitu Struts.

Kelemahan JSP ini yang kemudian dibaca oleh tim Apache dalam project Jakarta, dengan mengeluarkan Velocity, yang mana project ini mengambil ide dari tim WebMacro, yang merupakan sebuah teknologi berbasis servlet, yang mengeluarkan *layout managementnya* menjadi sebuah file biasa. Proses ini ternyata membuat proses *layouting* menjadi lebih cepat jauh dari JSP. Yang secara mudahnya adalah karena tidak adanya proses *compile on*

the fly dari *file code*.

Velocity dapat dikatakan servlet yang *diextend*, sehingga yang biasanya untuk membuat *layout* harus membuat *script* satu persatu dalam baris *code* servlet, menjadi tidak perlu dilakukan lagi. Malah dengan adanya library untuk HTML Designer seperti Dreamwaver, membuat layout menjadi lebih sangat productif.

Dengan Velocity, implementasi *layouting* dan coding dapat dipisah, sehingga dalam pemrograman lebih kompleks dalam bentuk tim yang lebih besar, tim pengembang *template* Velocity dan servletnya dapat dipisahkan.

Sayangnya proses untuk membuat sebuah *object* dapat *diparsing* didalam Velocity memerlukan keahlian khusus, walaupun tidak sulit. Hal ini karena Velocity bekerja seperti halnya sebuah keranjang sampah, setiap object yang telah diproses hanya perlu di masukan kedalam *contextnya* Velocity, dan langsung dapat *diparsing* didalam *template*. Yang mana mekanisme kerjanya akan dibahas setelah ini.

Ternyata proses pengiriman *object* kedalam keranjang Context Velocity, telah dipahami sekali oleh para programmer Open Source berbasis Java, sehingga lahirnya project VelocityTools yang memisahkan object pemrosesan dan proses pemasukan object kedalam keranjang Context Velocity dengan melakukan integrasi dengan Struts.

Ternyata mekanisme ini diadopsi juga oleh WebWork, yang saat itu merupakan saingan Struts. *Management result* dari WebWork memungkinkan setiap *object* yang terexecute, akan secara langsung dimap kedalam Velocity. Malah integrasi WebWork dengan Hibernate via Spring (yang semuanya akan dibahas setelah bab ini), memungkinkan semua model dalam Hibernate dilempar ke Velocity secara otomatis. Alhasil, mekanisme ini

terus dikembangkan, dan saat ini bukan hanya Velocity yang dapat diproses menjadi *viewer layer* dalam pemograman MVC, tetapi telah ditambahkan teknologi lainnya, seperti JasperReport untuk reporting, XML, JFreeChart untuk *charting*. Yang lebih menarik lagi, ternyata JSP juga dapat digunakan menjadi alternatif *presentation layer*, yang bekerja sama dengan Velocity.

Alhasil dengan kembalinya JSP sebagai *presentation layer*, ternyata secara *team work*, telah terjadi pergeseran mekanisme pemograman, programmer Java yang menggunakan integrasi dengan model MVC, telah lebih solid bukan hanya secara teknologi tetapi secara *team work*, alhasil karena mekanisme integrasi dan improvement yang dilakukan disetiap layer, baik itu oleh tim M, V ataupun C. Serta lahirnya teknologi *caching* yang dapat diimplementasikan disetiap layer, membuat sebuah teknologi yang secara teori lebih kompleks dan lebih sulit dipelajari, ternyata menjadi selain lebih mudah dipelajari, tetapi juga lebih cepat untuk beberapa kasus.

Pengalaman penulis yang telah merasakan kehebatan sebuah mekanisme pemograman berbasis MVC, penulis yang semula menggunakan teknologi JSP memerlukan minimum 2 minggu, dengan MVC dapat dikerjakan dalam 2-5 hari.

Bab-bab setelah ini akan mendalami bagaimana interaksi antara M, V, dan C didalam model pengembangan MVC, menjadi sebuah kekuatan.

Yang pasti dengan adanya MVC, secara tidak langsung kita telah berubah dari sebuah pemograman yang langsung didalam HTML menjadi sebuah mekanisme pemograman yang berbasis *run time*, yang mana bagi para programmer *embeded* HTML, mekanisme ini terkadang menyebalkan, karena kita harus mengcompile dulu untuk menjalankannya.

Tetapi teknologi ini telah diatasi juga dengan lahirnya teknologi *autodeploy*, yang memungkinkan melakukan semua *binary object* kita dihapus dan ditambahkan tanpa perlu server direstart terlebih dahulu. Mekanisme ini malah telah berkembang menjadi sebuah pengembangan berbasis modular. Diantaranya adalah dalam pemograman berbasis Portlet, memungkinkan dua *archive Java* dideploy dalam satu container menjadi satu aplikasi terintegrasi.

Untuk lebih jelasnya, mekanisme JSP dan Velocity akan dibahas setelah ini, yang mana, penulis menggunakan Tomcat sebagai Java EE *container*nya.

JSP, Servlet atau Turunannya

Bilamana kita perhatikan aplikasi yang dikembangkan dengan JSP lebih mudah, karena tidak diperlukan membuat baris `out.println` untuk membuat baris-baris HTML, yang mana merepotkan sekali bilamana dikembangkan menggunakan Servlet.

Dengan JSP, setiap programmer dapat mengembangkan aplikasinya menggunakan HTML Designer, sehingga *layout* HTML yang kompleks dapat *diembed* dengan perintah JSP. Ini pekerjaan menyenangkan. Tetapi akan menjadi mimpi buruk dalam pengembangan aplikasi Servlet.

Sayangnya JSP itu adalah sebuah teknologi yang kotor, yang membuat sampah di folder `/work/Catalina/localhost/namacontext`. Ini yang diyakini membuat JSP lebih lambat dari Servlet.

Ada sebuah metode yaitu dengan menggunakan JSPC, untuk membuat JSP menjadi sebuah *bytecode* tanpa perlu membuat sampah-sampah didalam folder work.

Selain itu JSP sulit sekali didebug. Karena JSP yang dibuat dan hasil kompilasi JSP berbeda secara teknologi dan juga proses *debugging* hanya dapat dilakukan didalam Servlet yang dihasilkan

dari kompilasi JSP. Saat ini Eclipse dengan WTPnya telah memiliki fitur debugging diatas JSP. Tetapi saja kalau *error* didalam sebuah Java *container*, *error* baris berbeda dengan *error* pada baris di JSP. Mendebbug dengan Eclipse dibahas dilampiran bab ini.

Sehingga didapat bahwa diperlukan sebuah servlet yang dapat membaca file HTML, sehingga *programmer* dapat menggunakan HTML designer untuk menghasilkan HTML, dan *programmer* dapat mendebbug dan membuat *bytecode* yang efisien.

Metode pengabungan HTML dengan Servlet ini dapat dilakukan dengan *extend* Servlet menjadi sebuah teknologi yang mereplace perintah `out.println`. Didunia Open Source metode ini telah menghasilkan sebuah teknologi yang ampuh, jadi kita tidak perlu membuat lagi, karena teknologinya sudah bagus sekali, teknologi gabungan ini semua disebut dengan Velocity dan FreeMarker.

FreeMarker adalah sebenarnya Velocity yang dikembangkan lebih luas sehingga fitur-fiturnya lebih banyak dibandingkan Velocity, dimana Velocity sendiri sesuai dengan namanya "kecepatan", tidak mengutamakan fitur-fitur tetapi mengutamakan sebuah eksekusi yang lebih ramping sehingga cepat. Salah satu fitur FreeMarker adalah kemampuan mengeksekusi taglib, dimana pada Velocity itu tidak memungkinkan.

Velocity dapat *download* di website <http://jakarta.apache.org/velocity>, dimana Velocity adalah sub *project* Jakarta dari Apache, yang mana satu level dengan project Tomcat.

Sedangkan FreeMarker dapat *download* dari <http://www.freemarker.org>

Bekerja dengan Velocity

Untuk menjalankan Velocity, masukan velocity-dep-1.4.jar kedalam

WEB-INF/lib. Didalam distribusi Velocity memang terdapat 2 jar, yaitu velocity-dep-1.4.jar dan velocity.jar. Perbedaannya velocity-dep-1.4.jar lebih lengkap, karena memasukan beberapa projek Apache lainnya seperti ORO dan Log4J.

Isi dari VeloServlet selengkapnya adalah:

```
package org.blueoxygen.book1.velocity;

import java.io.IOException;
import java.io.FileNotFoundException;

import java.util.Properties;
import java.util.Vector;

import javax.servlet.ServletConfig;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import org.apache.velocity.Template;
import org.apache.velocity.context.Context;
import org.apache.velocity.servlet.VelocityServlet;
import org.apache.velocity.app.Velocity;

import org.apache.velocity.exception.
ResourceNotFoundException;

import org.apache.velocity.exception.
ParseException;

public class VeloServlet extends VelocityServlet
{

protected Properties loadConfiguration(ServletConfig
config )
    throws IOException, FileNotFoundException {
    Properties p = new Properties();

    String path = config.getServletContext()
        .getRealPath("/");
}
```

```
if (path == null)
{
    System.out.println("VeloServlet.loadConfiguration()
: unable to " + "get the current webapp root.
Using '/'. Please fix.");
}

path = "/";
}

p.setProperty(
Velocity.FILE_RESOURCE_LOADER_PATH, path );
p.setProperty( "runtime.log", path +
"velocity.log" );
return p;
}

public Template handleRequest(
HttpServletRequest request,
HttpServletResponse response, Context ctx ) {
String p1 = "Frans";
String p2 = "Harold";

Vector personList = new Vector();
personList.addElement( p1 );
personList.addElement( p2 );

ctx.put("theList", personList );

int i = 100;
i = i *2000;

Vector variableList = new Vector();
variableList.addElement(""+i);

i=i*250;
variableList.addElement(""+i);

ctx.put("variableList",variableList);

Template outty = null;
```

```
try {
    outty = getTemplate("sample.vm");
} catch( ParseErrorException pee ) {
    System.out.println("VeloServlet :
        parse error for template " + pee);
}
catch( ResourceNotFoundException rnce ) {
    System.out.println("VeloServlet :
        template not found " + rnce);
}
catch( Exception e )
{
    System.out.println("Error " + e);
}
return outty;
}
}
```

Sedangkan template sample.vm adalah

```
<html>
<head><title>Sample Velocity page</title></head>
<body bgcolor="#ffffff">
<center>

<h2>Hello from BlueOxygen!</h2>
<i>Here's the list of people</i>
<table cellspacing="0" cellpadding="5" width="100%">
    <tr>
        <td bgcolor="#eeeeee" align="center">
            Names
        </td>
    </tr>
    #foreach ($name in $theList)
    <tr>
        <td bgcolor="#eeeeee">$name</td>
    </tr>
    #end
    <tr>
        <td bgcolor="#eeeeee" align="center">
            Variables
        </td>
    </tr>
</table>
</center>
</body>
</html>
```

```
</td>
</tr>

#foreach ($variable in $variableList)
|  |
| --- |
| $variable |

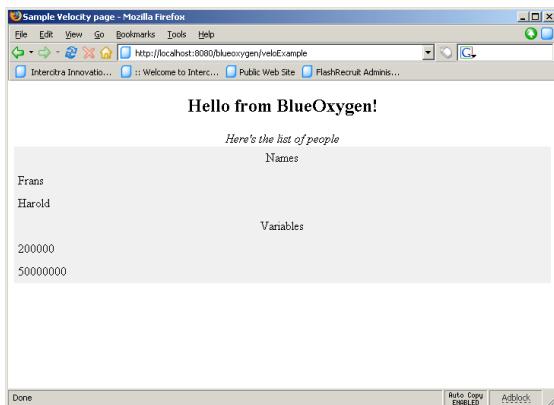
#end

</table>
</center>
</html>
```

Diatas adalah salah satu sample dari mekanisme yang memungkinkan mengembangkan aplikasi menggunakan HTML Designer sehingga menghasilkan HTML yang lebih enak dibaca, juga menggunakan Servlet sehingga ekskusi menjadi lebih cepat. Metode Velocity ini diyakinkan 4 kali lebih cepat dari pemograman berbasis JSP.

Perhatikan baris `public class VeloServlet extends VelocityServlet`, ini menyatakan bahwa applikasi SampleServlet bukan turunan dari Servlet yang telah dibahas sebelumnya tetapi VelocityServlet, dimana VelocityServlet ini adalah turunan dari Servlet.

Kemudian cari baris `outty = getTemplate("sample.vm") ;`, ini artinya memanggil file sample.vm, dimana sample.vm adalah sebuah HTML dengan embed velocity macro.



Output dari eksekusi dari VeloServlet

Adapun sebelum menjalankan VeloServlet, tambahkan baris berikut di web.xml

```

<servlet>
    <servlet-name>VeloServlet</servlet-name>
    <servlet-class>org.blueoxygen.book1.velocity.VeloServlet</servlet-class>

    <init-param>
        <param-name>properties</param-name>
        <param-value>/WEB-INF/velocity.properties</param-value>
    </init-param>

    <load-on-startup>1</load-on-startup>
</servlet>

<servlet-mapping>
    <servlet-name>VeloServlet</servlet-name>
    <url-pattern>/veloExample</url-pattern>
</servlet-mapping>
```

Setelah itu buatlah sebuah file velocity.properties didalam folder WEB-INF, isinya adalah:

```
resource.loader = class
class.resource.loader.class = org.apache.velocity.
runtime.resource.loader.ClasspathResourceLoader

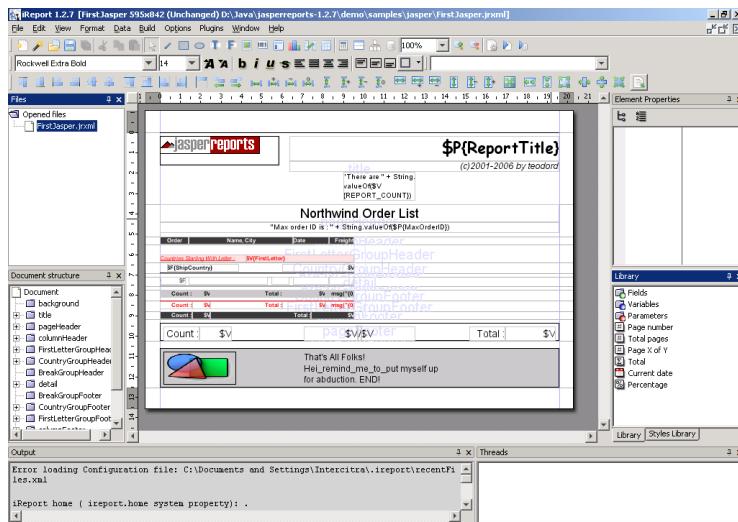
runtime.log = velocity.log
```

Setiap kali Tomcat dijalankan, akan terbentuk sebuah file velocity.log, file ini dapat dibaca, dan bilamana ada kesalahan dalam eksekusi, maka log ini akan melaporkan kesalahannya.

JasperReport dan iReport

Setelah kita mengetahui adanya teknologi presentasi yang dikembangkan dari Servlet, yaitu JSP, Velocity, ataupun Freemarker, yang mana teknologi Servlet tersebut adalah teknologi yang khusus untuk presentation layer. Ternyata ada teknologi teknologi lain yang dapat digunakan untuk menghasilkan *presentation layer* lainnya, seperti PDF, GIF, JPG, multimedia, ataupun teknologi *meta data* seperti XML, SOAP.

Untuk mengembangkan teknologi PDF atau outputnya berbentuk report seperti Excel, Word, ODF, ataupun CSV, dapat menggunakan JasperReport, iText atau POI. Sebenarnya ini cara membuat solusi untuk presentasi yang berbeda. Velocity, Servlet atau JSP sebenarnya menghasilkan output untuk dikonsumsi oleh browser, hal ini dikarenakan Servlet merupakan implementasi menghandle HTML.

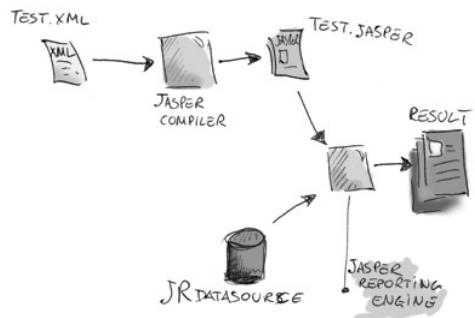


iReport sebagai IDE JasperReport

Secara garis besar, sebenarnya semua teknologi *presentation layer* adalah mentransformasikan sebuah sumber data menjadi data lain. JasperReport adalah salah satunya engine untuk membuat report baik itu yang outputnya sebagai PDF, Excel, ataupun CSV. Dimana untuk mengembangkannya telah datang sebuah IDE yang bekerja mirip seperti Eclipse, hanya bedanya IDE ini bekerja sebagai *designer* JasperReport. IDE ini bernama iReport. JasperReport dan iReport saat ini bernaung dibawah perusahaan JasperSoft, yang merupakan perusahaan Open Source yang menyediakan solusi *reporting* dan *business intelligence*.

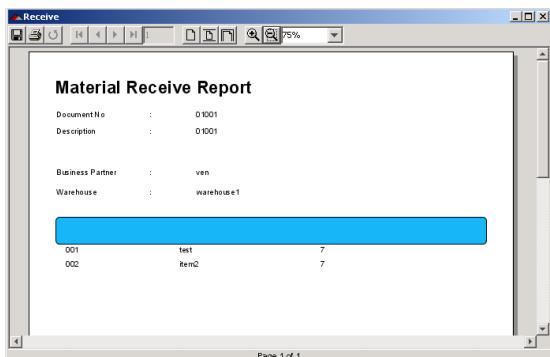
JasperReport memiliki sebuah format file yang berbentuk XML. XML ini adalah *metadata* yang spesifik bekerja di Java. *Metadata* ini bilamana *dicompile* menggunakan jasper *compiler*, sebuah *compiler* berbasis Java. *Compiler* ini akan merubah *metadata* ini menjadi report *bytecode*, yang secara teknis dapat dikatakan sebagai sebuah Java *class* yang digunakan untuk fungsi yang lain.

Bilamana bytecode tersebut dieksekusi menggunakan JasperReport, maka akan menghasilkan *output* yang diinginkan. Untuk lebih jelasnya dapat melihat ilustrasi dibawah ini:



Ilustrasi Transformasi dari JasperReport

Selain itu JasperReport juga telah datang dengan JasperReport Viewer, yaitu aplikasi Swing untuk melakukan *preview* terhadap *output* dari proses transformasi Jasper. Project Postila yang dibahas dibab 9, menggunakan implementasi Jasper sebagai *engine reportnya*.



JasperReport Viewer

JFreeChart dan Cewolf

Ada sebuah teknologi atau sebuah *engine* yang bekerja dengan menghasilkan *output* yang lebih ilustratif, yaitu berbentuk grafis. Engine tersebut disebut JFreeChart. JFreeChart adalah opensource *charting engine* berbasis Java. JFreeChart ini telah terintegrasi penuh dengan JasperReport, sehingga kita dapat menghasilkan report yang terdapat chart atau grafis seperti bar chart, scatter.



Output JFreeChart berbentuk Bar Chart

Untuk mempermudah implementasi JFreeChart dengan JSP, ada sebuah teknologi yang merubah JFreeChart menjadi taglib, projek opensource ini bernama cewolf.

Mondrian dari Pentaho

Servlet, JasperReport ataupun JFreeChart/Cewolf adalah solusi yang menghasilkan *output static*, artinya hasil akhirnya harus melakukan proses ulang atau *rerequest* untuk menghasilkan *report* yang lebih interaktif, projek ini adalah Mondrian. Mondrian sekarang merupakan salah satu tools *business intelligence* dari Pentaho, salah satu penyedia solusi BI berbasis Open Source terbesar didunia.

Mondrian sebenarnya implementasi dari MDX, sebuah *engine* OLAP yang disubmit ke ECMA oleh Microsoft, dimana MDX ini merupakan format standar dari SQL Server Repoting.

Mondrian datang dengan integrasi dengan JPivot, sebuah implementasi *pivot table*, yang mana kita dapat menemukannya di Microsoft Excel. Berikut adalah bentuk dari output JPivot, dimana source datanya berbentuk MDX menggunakan Mondrian.

		Measures		
Products	Region	▼ Measures[0]	▼ Measures[1]	▼ Measures[2]
- All Products[0]	- All Region[0]	902.31 ↗	1,069.87 ↗	1,195.14 ↘
	+Region[0]	848.53 ↘	1,031.69	884.32
	-Region[1]	975.27 ↗	805.13 ↗	948.12 ↘
	+City[0]	893.16 ↘	1,034.77	834.51
	+City[1]	1,012.79 ↗	884.11 ↗	1,101.19 ↘
	+City[2]	1,013.88 ↘	1,054.62	972.94
	+City[3]	881.08 ↗	1,175.80 ↗	967.78 ↘
	+City[4]	1,094.37 ↘	1,032.40	857.15
	+City[5]	1,002.53 ↗	997.01 ↗	1,073.30 ↘
	+City[6]	918.05 ↘	1,085.70	953.94
	+City[7]	1,099.73 ↗	944.40 ↗	1,190.58 ↘
+Category[0]	- All Region[0]	1,017.41 ↗	787.35 ↗	1,035.24 ↘
	+Region[0]	1,019.25 ↘	1,123.19	923.65
	+Region[1]	877.00 ↗	1,148.39 ↗	1,085.34 ↘
	+Region[2]	752.89 ↗	1,069.97	943.07
	+Region[3]	829.57 ↗	916.46 ↗	1,010.03 ↘
+Category[1]	+ All Region[0]	1,032.02 ↗	898.40 ↗	1,022.07 ↘
	+ All Region[0]	963.97 ↘	1,016.30	935.30
	+ All Region[0]			

Pivot berbentuk HTML dengan JPivot dan Mondrian

Adapun output diatas adalah data yang dikeluarkan merupakan *output* dari MDX untuk table C_INVOICELINE, yang merupakan MDX untuk melihat sales dari aplikasi Compiere.

Berikut adalah MDX skema dari pivot diatas.

```
<?xml version="1.0" encoding="UTF-8"?>
<Schema name="Compiere">
<Cube name="Sales">
<Table name="C_INVOICELINE"/>
```

```
<Dimension foreignKey="M_PRODUCT_ID" name="Product">
<Hierarchy allMemberName="All Product"
    hasAll="true"
    name="Product" primaryKey="M_PRODUCT_ID"
    primaryKeyTable="M_PRODUCT">

<Join leftKey="M_PRODUCT_CATEGORY_ID"
    rightKey="M_PRODUCT_CATEGORY_ID">
    <Table name="M_PRODUCT"/>
    <Table name="M_PRODUCT_CATEGORY"/>
</Join>

<Level column="NAME"
    name="Category" table="M_PRODUCT_CATEGORY"/>
<Level column="NAME"
    name="Product" table="M_PRODUCT"/>
</Hierarchy>
</Dimension>

<Dimension foreignKey="C_INVOICE_ID"
    name="Customer">
    <Hierarchy allMemberName="All Customer"
        hasAll="true" name="Customer"
        primaryKey="C_INVOICE_ID"
        primaryKeyTable="C_INVOICE">
        <Join leftKey="C_BP伙伴关系"
            rightAlias="C_BP伙伴关系"
            rightKey="C_BP伙伴关系">
            <Table name="C_INVOICE"/>
            <Join leftKey="C_BP_GROUP_ID"
                rightKey="C_BP_GROUP_ID">
                <Table name="C_BP伙伴关系"/>
                <Table name="C_BP_GROUP"/>
            </Join>
        </Join>
    </Hierarchy>
</Dimension>

<Dimension foreignKey="C_BP伙伴关系_ID"
    name="Customer Cat">
    <Hierarchy allMemberName="All Customer Cat"
        hasAll="true" name="Customer Cat"
        primaryKey="C_BP伙伴关系_ID"
        primaryKeyTable="C_BP伙伴关系">
        <Join leftKey="C_BP伙伴关系"
            rightAlias="C_BP伙伴关系"
            rightKey="C_BP伙伴关系">
            <Table name="C_BP伙伴关系"/>
            <Table name="C_BP_GROUP"/>
        </Join>
    </Hierarchy>
</Dimension>
```

```
</Dimension>

<Measure aggregator="sum"
    column="QTYINVOICED"
    formatString="#,###.0"
name="Qty Sales"/>

<Measure aggregator="sum"
    column="LINENETAMT"
    formatString="#,###.00" name="RM Sales"/>
</Cube>
</Schema>
```

BAB III

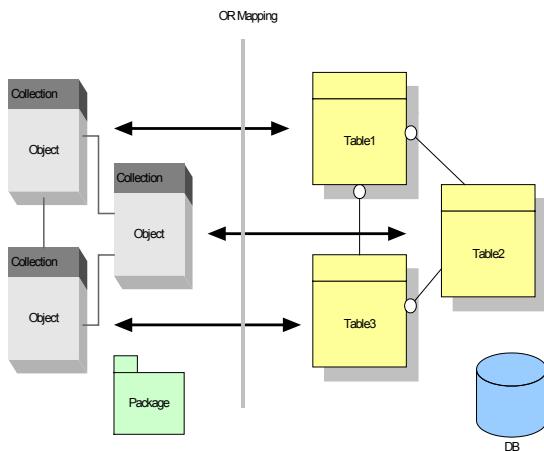
Berkenalan dengan Hibernate untuk solusi Model pada MVC

Bab ini akan membahas lebih dalam mengenai teknologi layer Model pada MVC. Model adalah sebuah *layer* yang lebih dekat ke sumber data, baik itu berupa database, webservices, atau file system.

Untuk membuat model ini dalam berinteraksi dengan Controller, dapat dilakukan dengan menggunakan mekanisme membuat *thread* baru dengan New, atau melakukan *injection*. Bab berikutnya akan membahas bagaimana *injection*, yang mana akan membuat layer *controller* dan *model* menjadi sebuah kesatuan.

Bab ini akan membahas lebih dalam mengenai mekanisme *model* yang berinteraksi dengan database, yang sering disebut teknologi ORM atau singkatan dari *Object Relational Mapping*.

Sebenarnya teknologi ORM ada beberapa macam, seperti Toplink dari Oracle, Kodo dari Bea Systems, dan tentu saja yang akan dibahas lebih dalam dibab ini adalah Hibernate. ORM paling populer didunia.



Mekanisme Object Relational Mapping Bekerja

ORM sebenarnya adalah sebuah teknologi yang menjadi satu lapis antara aplikasi dengan *database*, yang mana ORM ini bekerja seperti *database* juga, tetapi hanya berbentuk objek. Setiap objek didalam ORM, umumnya mewakili *table* dalam *database*. Akibat dari teknologi *mapping* ini membuat sebuah aplikasi yang dikembangkan dengan ORM, tidak terikat dengan *database* manapun. Sedangkan untuk melakukan *query database*, dengan ORM digunakan sebuah object relational language, yang bekerja mirip dengan SQL. Umunya setiap *table* dan objek POJO dimap dengan sebuah XML .

Setelah spesifikasi final EJB3 ini keluar, yaitu dengan datangnya standar *Java Persistence API* atau JPA sebagai salah satu dari spesifikasi inti dari EJB3. Dimana sebenarnya JPA ini adalah versi anonasi dari ORM metadata. ORM mengganti XMLnya menjadi perintah anonasi didalam Java. Akibat dari standar EJB3 yang bernomor JSR 220 ini, telah membuat teknologi Hibernate, Toplink dan Kodo, yang semuanya adalah teknologi ORM, menjadi inti dari EJB3. Untuk Hibernate, agar Hibernate dapat menjadi EJB3

harus melakukan integrasi antara Hibernate *core*, Hibernate annotation dan Hibernate entity manager. Dengan hadirnya JPA, membuat teknologi TopLink dan Hibernate secara kasat mata adalah sama.

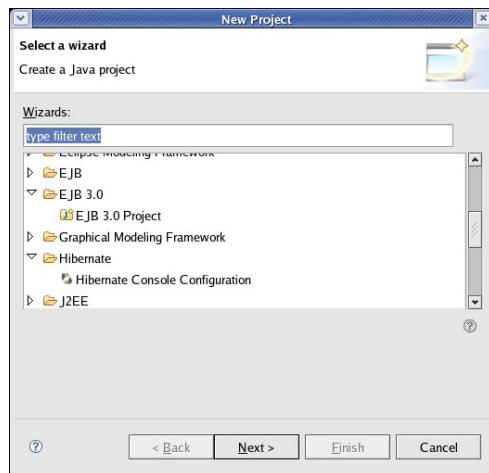
Hibernate yang datang dengan 2 versi yaitu versi core dan versi annotation, sebenarnya secara fungsi adalah sama, perbedaannya Hibernate versi *core* memerlukan XML sebagai *mappernya*. Yang mana dalam implementasinya sebenarnya hanya menganti *object Configuration* menjadi AnnotationConfiguration.

Berkenalan dengan Hibernate Tools

Untuk memulai sebuah projek berbasis Hibernate, yang termudah adalah dengan menggunakan Eclipse Calisto, dan menambahkan JBoss IDE kedalamnya.

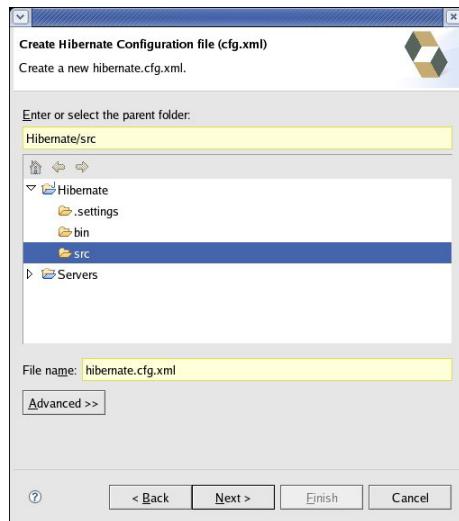
Hibernate Tools sebenarnya datang dengan 2 teknologi, sebagai *plugins* Eclipse dan sebagai *script* pada ant. Penulis merasakan bilamana menggunakan *plugins*, seorang *programmer* dapat mencoba HQLnya terlebih dahulu, sedangkan dengan Ant, memungkinkan diintegrasikan dengan *script* lainnya didalam Ant, seperti DBUnit, Junit, ataupun CVS update. DBUnit dan HibernateTool akan dibahas dalam bab ini.

Mekanisme mengidentifikasi Hibernate Tools telah berjalan dengan baik adalah melakukan New Project, coba *scroll*, pasti ada kata Hibernate didalamnya. Sedangkan cara menginstallnya adalah dengan mengextract JBoss IDE atau Hibernate Tools IDE kedalam *folder* Eclipse. Hal ini terjadi untuk semua tipe Eclipse tanpa perlu perubahan. Tentu saja plugins JBoss terbaru memerlukan Calisto.



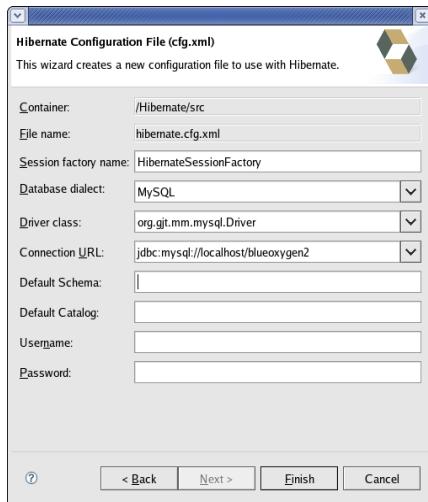
Membuat Project Baru

Buatlah sebuah Java Project, kemudian buat sebuah Hibernate Configuration file (hibernate.cfg.xml).



Membuat hibernate.cfg.xml baru

Sebaiknya hibernate.cfg.xml disimpan didalam *folder src*, yang artinya sama dengan dikenal saat terjadi *runtime*, dengan Eclipse, akan tercopy ke folder build.



Setting Configuration untuk hibernate.cfg.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE hibernate-configuration PUBLIC
"-//Hibernate/Hibernate Configuration DTD 3.0//EN"
"http://hibernate.sourceforge.net/hibernate-
configuration-3.0.dtd">

<hibernate-configuration>
<session-factory name="HibernateSessionFactory">
<property name="hibernate.connection.driver_class">org.
gjt.mm.mysql.Driver</property>

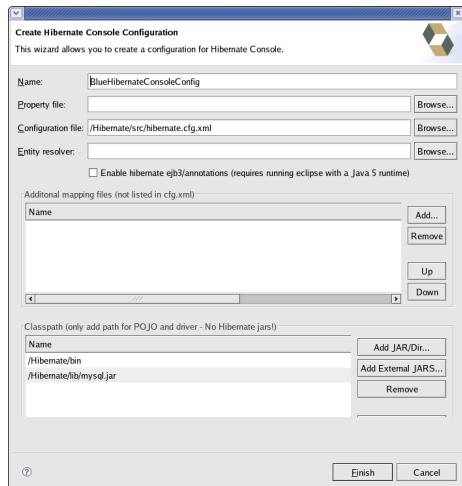
<property name="hibernate.connection.
password">tulalit</property>

<property name="hibernate.connection.url">jdbc:
mysql://localhost/blueoxygen2</property>
```

```
<property name="hibernate.connection.username">root</property>
<property name="hibernate.dialect">org.hibernate.dialect.MySQLDialect</property>

</session-factory>
</hibernate-configuration>
```

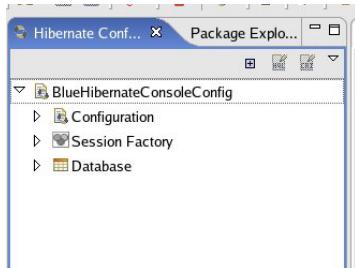
Step berikutnya adalah dengan membuat Hibernate Console Configuration, *configuration* ini diperlukan untuk membuat HQL (Hibernate Query Language) yaitu SQLnya Hibernate, atau mentest Criteria (*filter* terhadap sebuah *query*).



Jangan lupa memasukan output project kedalam classpath

Jangan lupa untuk memasukan folder binary kedalam *classpath* dari Hibernate Console Configuration ini. Hal ini diperlukan karena Hibernate Console ini tidak dapat mengenai objek ORM. Bilamana proses ini telah selesai dapat masuk ke Hibernate perspective, maka akan muncul sebuah *tree* dari Hibernate yang isinya adalah Configuration, SessionFactory dan database. Adapun

Database ini terdiri dari semua POJO yang memiliki anonasi JPA atau XML metadata.



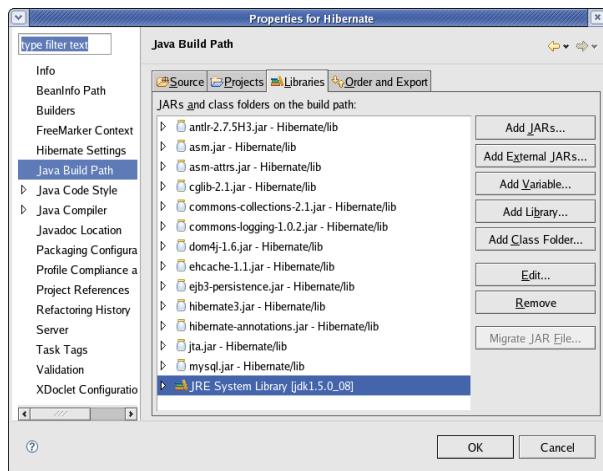
Hibernate Configuration console

Bilaman step ini telah berhasil, artinya Hibernate *plugins* pada Eclipse Calisto telah terinstall dengan baik.

Membuat Object Mapping Pertama

Langkah berikutnya adalah dengan membuat sebuah POJO yang memiliki anonasi, POJO ini akan terhubung dengan table “businesspartner”.

Adapun sebelum memulai langkah berikutnya, copylah sebuah .jar dari binary hibernate (dalam *folder lib*) yang didownload dari websitenya yaitu <http://www.hibernate.org>. Masukan semua jar tersebut dalam Build Pathnya Eclipse, caranya dengan mengklik kanan dari projek Javanya, kemudian tambahkan jarnya.



Library JAR yang dimap dalam projek

Bilamana telah selesai, buatlah sebuah class DefaultPersistence, objek ini adalah sebuah objek yang termapping untuk menghasilkan id yang unik dari setiap kegiatan persistensi. Objek ini sangat diperlukan dalam pengembangan kedepannya, karena dengan melakukan implementasi objek ini, akan terbentuk sebuah table dengan record id, serta akan terisi otomatis bilamana dilakukan save terhadap data baru.

Adapun DefaultPersistence ini adalah:

```
package org.blueoxygen.bbook2;

import java.io.Serializable;
import javax.persistence.Embedded;
import javax.persistence.GeneratedValue;
import javax.persistence.Id;
import javax.persistence.MappedSuperclass;
import org.hibernate.annotations.GenericGenerator;

@MappedSuperclass
public class DefaultPersistence implements Serializable
```

```
{  
private String id;  
  
@Id @GeneratedValue(generator="system-uuid")  
@GenericGenerator(name="system-uuid",  
strategy="uuid")  
public String getId() {  
    return id;  
}  
  
public void setId(String id) {  
    this.id = id;  
} }
```

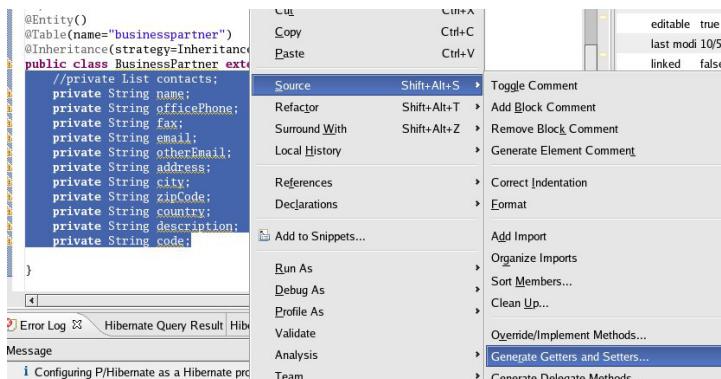
Adapun langkah yang harus dilakukan adalah dengan membuat sebuah Java class dengan nama BusinessPartner.java.

```
package org.blueoxygen.bbook2;  
  
import java.util.ArrayList;  
import java.util.List;  
import javax.persistence.Column;  
import javax.persistence.Entity;  
import javax.persistence.Inheritance;  
import javax.persistence.InheritanceType;  
import javax.persistence.ManyToOne;  
import javax.persistence.Table;  
import org.blueoxygen.bbook2.DefaultPersistence;  
  
import org.hibernate.Session;  
  
@Entity()  
@Table(name="businesspartner")  
@Inheritance(strategy=InheritanceType.SINGLE_TABLE)  
public class BusinessPartner extends DefaultPersistence  
{  
    //private List contacts;  
    private String name;  
    private String officePhone;  
    private String fax;  
    private String email;  
    private String otherEmail;
```

```
private String address;
private String city;
private String zipCode;
private String country;
private String description;
private String code;

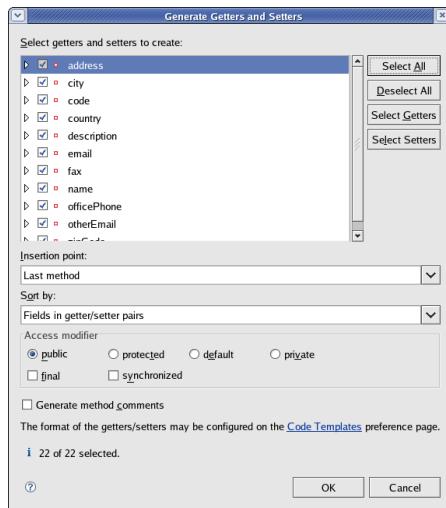
}
```

Setelah itu, pilih semua *private object* dari name sampai *code*, kemudian klik kanan, pilihlah



Membuat Get dan Set dari variable

Kemudian pilihlah semua variable yang ada, dan secara otomatis semua *variable* akan memiliki method *getXXX* dan *setXXX*.



Pemilihan untuk generate Get/Set

Tambahkan @Column() pada semua baris diatas getXXX, seperti contoh pada getAddress():

```
@Column()
public String getAddress() {
    return address;
}
```

Setelah itu jangan lupa untuk *register class* yang telah dibuat didalam hibernate.cfg.xml, dengan menambahkan baris berikut:

```
<mapping class="org.blueoxygen.bbook2.BusinessPartner"/>
```

Mengenerate Database dengan HibernateTools

Bilamana langkah ini telah dilakukan, maka proses berikutnya adalah dengan membuat *table* secara otomatis menggunakan HibernateTools.

Untuk mengenerate HibernateTools diperlukan ant *script* (build).

xml), sehingga table businesspartner dapat digenerate otomatis.

Adapun file ant atau build.xml adalah sebagai berikut:

```
<project name="bbook2hibernate"
    default="schemaexport">

<target name="init">

    <property name="bb2h.base"
        value="/home/frans/workspace/Hibernate"/>

    <property name="bb2h.lib"
        value="${bb2h.base}/lib"/>

    <property name="bb2h.classes"
        value="${bb2h.base}/bin"/>

    <property name="bb2h.source"
        value="${bb2h.base}/src"/>
</target>
<target name="schemaexport" depends="init">
<taskdef name="hibernatetool"
classname="org.hibernate.tool.ant.
HibernateToolTask">
    <classpath>
        <fileset dir="${bb2h.lib}">
            <include name="**/*.jar"/>
        </fileset>
        <path location="${bb2h.bin}"/>
    </classpath>
</taskdef>
<hibernatetool destdir="${bb2h.bin}">

<annotationconfiguration configurationfile="${bb2h.
source}/hibernate.cfg.xml"/>

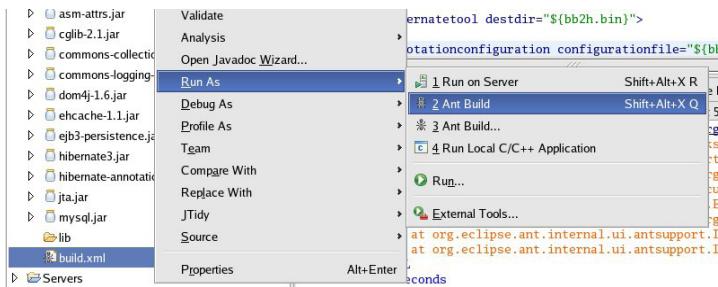
<hbm2ddl drop="true" console="false" create="true"
    outputfilename="bb2h-ddl.sql"
    delimiter=";"
    export="true"
/>
```

```

</hibernatetool>
</target>
</project>

```

Jalankan script build.xml tersebut dengan cara melakukan klik kanan terhadap file build.xml yang telah dibuat, kemudian pilih Run As, dilanjutkan dengan Ant script.



Mengeksekusi Ant dari Eclipse

Bilamana tidak terjadi masalah, maka *console* akan menghasilkan output seperti berikut ini:

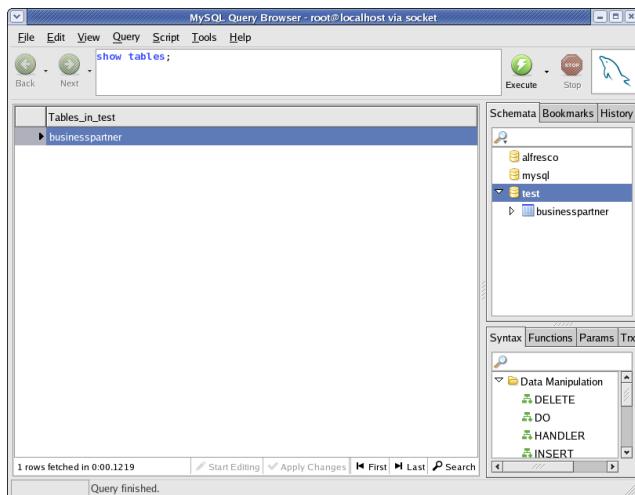
```

Buildfile: /home/frans/workspace/Hibernate/build.xml
init:
schemaexport:
[hibernatetool] Executing Hibernate Tool with a
Hibernate Annotation/EJB3 Configuration
[hibernatetool] 1. task: hbm2ddl (Generates database
schema)
[hibernatetool] Oct 5, 2006 11:18:49 PM org.hibernate.
cfg.Environment <clinit>
[hibernatetool] INFO: Hibernate 3.2 cr1
[hibernatetool] Oct 5, 2006 11:18:49 PM org.hibernate.
cfg.Environment <clinit>
[hibernatetool] INFO: hibernate.properties not found
[hibernatetool] Oct 5, 2006 11:18:49 PM org.hibernate.
cfg.Environment buildBytecodeProvider
[hibernatetool] INFO: Bytecode provider name : cglib
.....

```

```
....  
....  
[hibernatetool] Oct 5, 2006 11:18:50 PM org.hibernate.  
connection.DriverManagerConnectionProvider configure  
[hibernatetool] INFO: connection properties: {user=root,  
password=*****}  
[hibernatetool] Oct 5, 2006 11:18:50 PM org.hibernate.  
tool.hbm2ddl.SchemaExport execute  
[hibernatetool] INFO: schema export complete  
[hibernatetool] Oct 5, 2006 11:18:50 PM org.hibernate.  
connection.DriverManagerConnectionProvider close  
[hibernatetool] INFO: cleaning up connection pool:  
jdbc:mysql://localhost/test  
BUILD SUCCESSFUL  
Total time: 3 seconds
```

Bilamana tidak ditemukan *error*, coba jalankan MySQL Query Browser, klik *table* test, maka secara otomatis akan terbentuk *table* businesspartner. Jalankan perintah “show tables”, maka secara otomatis akan muncul *tables_in_test* bernama businesspartner. Ini artinya proses pembuatan *table* dari hibernate telah berhasil dengan sukses.



MySQL Query Builder

Menjalankan Objek Hibernate

Langkah berikutnya bilamana pembuatan database telah berhasil, adalah dengan mencoba menjalankan sebuah aplikasi Java yang memanggil objek Hibernate.

Secara konsep proses ini adalah melakukan inisiasi `HibernateSessionFactory`, kemudian baru melakukan eksekusi, dimana untuk mengeksekusi diperlukan script HQL (Hibernate Query Language). Adapun *script* untuk melakukan testing Hibernate adalah sebagai berikut:

```
package org.blueoxygen.bbook2;

import java.util.ArrayList;
import java.util.List;

import org.hibernate.Criteria;
import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.hibernate.cfg.AnnotationConfiguration;

public class HibernateQueryTest {

    public static void main(String[] args) {

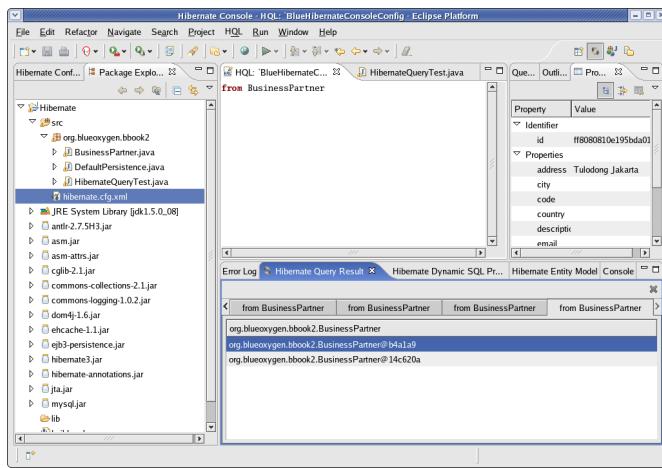
        SessionFactory sf;
        Session sess;
        AnnotationConfiguration config =
            new AnnotationConfiguration();
        config.configure("hibernate.cfg.xml");
        sf = config.buildSessionFactory();
        sess = sf.openSession();
        BusinessPartner bpt = new BusinessPartner();
        bpt.setName("Meruvian11");
        bpt.setAddress("Tulodong Jakarta");
        sess.saveOrUpdate(bpt);
        List<BusinessPartner> bp =
            new ArrayList<BusinessPartner>();
```

```
bp = sess.createQuery("FROM "
+BusinessPartner.class.getName()).list();
for(BusinessPartner business : bp){
    System.out.println("Biz Name:"
+business.getName());
}
sf.close();
}
}
```

Terdapat 2 kegiatan dalam *script* diatas, yaitu membuat *record* baru dengan perintah `saveOrUpdate` dan melakukan *query* ke *database*. Perintah *query* ke database (HQL) adalah “`FROM`”`+BusinessPartner.class.getName()`.

Melakukan Query dengan HSQL Editor

Adapun cara lain untuk mengetahu total *record* didalam database adalah dengan menjalankan HSQL Script Editor pada Eclipse. Cara memanggil HQL Editor adalah dengan menekan *icon* HSQL diatas Hibernate Configuration Config. Masukan perintah berikut “`FROM BusinessPartner`”, artinya *query* semua data dari objek `BusinessPartner`. Harap diperhatikan agar nama objek adalah *case sensitive*.



HQL Editor

Cobalah klik Hibernate Query Result. Bilamana query telah berhasil, akan terdapat hash objek didalam Query Result, cobalah klik salah satu dari objek hash tersebut. Secara otomatis Propeties akan memunculkan informasi dari isi dari objek tersebut. Bandingkan dengan melakukan perintah “SELECT * FROM businesspartner” pada database, hasilnya harus sama.

Bilamana semua langkah diatas telah berhasil dijalankan, artinya Anda telah berhasil bekerja dengan sebuah teknologi ORM bernama Hibernate.

ORM dengan Relationship

Setelah mencoba membuat projek kecil berbasis Hibernate dengan memapping satu *table* dan memasukan anotasi, berikut ini adalah mengembangkan dengan metode many to many, yaitu dengan kasus Event dan Person.

Untuk lebih mudahnya buat lah dua class dengan nama Event dan Person pada projek Java yang telah ada. Adapun kode untuk

Person.java adalah

```
package org.blueoxygen.bbook2;

import java.util.Set;

import javax.persistence.CascadeType;
import javax.persistence.Entity;
import javax.persistence.Inheritance;
import javax.persistence.InheritanceType;
import javax.persistence.ManyToMany;
import javax.persistence.Table;

import org.blueoxygen.bbook2.DefaultPersistence;

@Entity
@Table(name="person")
@Inheritance(strategy=InheritanceType.SINGLE_TABLE)
public class Person extends DefaultPersistence {
    private String age;
    private String firstname;
    private String lastname;
    private Set<Event> events;

    public String getAge() {
        return age;
    }
    public void setAge(String age) {
        this.age = age;
    }
    public String getFirstname() {
        return firstname;
    }
    public void setFirstname(String firstname) {
        this.firstname = firstname;
    }
    public String getLastname() {
        return lastname;
    }
    public void setLastname(String lastname) {
        this.lastname = lastname;
    }
}
```

```
@ManyToMany (mappedBy="persons")
public Set<Event> getEvents() {
    return events;
}
public void setEvents(Set<Event> event) {
    this.events = event;
} }
```

Sedangkan kode untuk Event.java adalah

```
package org.blueoxygen.bbook2;

import java.sql.Date;
import java.util.Set;

import javax.persistence.CascadeType;
import javax.persistence.Entity;
import javax.persistence.Inheritance;
import javax.persistence.InheritanceType;
import javax.persistence.ManyToMany;
import javax.persistence.Table;

import org.blueoxygen.bbook2.DefaultPersistence;

@Entity
@Table(name="event")
@Inheritance(strategy=InheritanceType.SINGLE_TABLE)
public class Event extends DefaultPersistence {
    private String title;
    private Date date;
    private Set<Person> persons;

    @ManyToMany(cascade={CascadeType.PERSIST,
    CascadeType.MERGE})
    public Set<Person> getPersons() {
        return persons;
    }
    public void setPersons(Set<Person> person) {
        this.persons = person;
    }
    public Date getDate() {
```

```
        return date;
    }
    public void setDate(Date date) {
        this.date = date;
    }
    public String getTitle() {
        return title;
    }
    public void setTitle(String title) {
        this.title = title;
    }
}
```

Perhatikan baris pada Event.java, dengan anotasi @ManyToMany

```
@ManyToMany(cascade={CascadeType.PERSIST,
    CascadeType.MERGE})
public Set<Person> getPersons() {
    return persons;
}
public void setPersons(Set<Person> person) {
    this.persons = person;
}
```

Baris diatas menerangkan bahwa isi dari *persons* adalah berbentuk Java *collection*, dimana pada kasus ini mengimplementasikan Set. Untuk kasus ini, sebenarnya dapat melakukan implementasi yang bermacam-macam seperti ArrayList, HashMap, ataupun menggunakan implementasi yang lebih *advanced* menggunakan Jakarta Commons Collections dari Apache.

Bilamana telah selesai, *register* kedua *class* tersebut kedalam hibernate.cfg.xml diantara session-factory.

```
<mapping class="org.blueoxygen.bbook2.
BusinessPartner"/>
<mapping class="org.blueoxygen.bbook2.Event"/>
```

Jalankan kembali build.xml, untuk membuat *table* Event dan Person.

Untuk mencoba hubungan antara Event dan Person, dapat dengan menggunakan mekanisme yang sama dengan contoh pertama. Berikut ini adalah sebuah contoh lengkap untuk menambah data kedalam objek Event.

```
package org.blueoxygen.bbook2;

import java.sql.Date;
import java.util.ArrayList;
import java.util.HashSet;
import java.util.List;
import java.util.Set;

import org.blueoxygen.bbook2.Event;
import org.blueoxygen.bbook2.Person;
import org.hibernate.Criteria;
import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.hibernate.cfg.AnnotationConfiguration;

public class HibernateOnetoManyTest {
    public static void main(String[] args) {
        SessionFactory sf;

        Session sess;
        Criteria crit;

        AnnotationConfiguration config = new
            AnnotationConfiguration();
        config.configure("hibernate.cfg.xml");
        sf = config.buildSessionFactory();
        sess = sf.openSession();
        sess.beginTransaction();

        Event event = new Event();
        List<Person> persons =
            new ArrayList<Person>();
        persons = (ArrayList<Person>)

        sess.createCriteria(Person.class).list();
```

```
event.setTitle("Third Event");
event.setDate(
    new Date(System.currentTimeMillis()));
event.setPersons(new HashSet<Person>());
for(Person p : persons){
    event.getPersons().add(p);
}
sess.save(event);
sess.getTransaction().commit();

Person person =
    (Person) sess.get(Person.class, "1");
for(Event e : person.getEvents()){
System.out.println(e.getTitle());
}
sf.close();
}
}
```

Coba jalankan, pasti *error* pada baris (Person) sess.get(Person.class, "1"), hal ini dikarenakan, method ini mereturn objek Person dengan id=1. Untuk mengatasinya, buatlah sebuah record pada table Person diisi dengan data terlebih dahulu, dengan id = 1. Atau menjalankan build.xml dengan Ant, dengan menjalankan target untuk export-data.

Perhatikan baris untuk mengisi objek Event.

```
sess.beginTransaction();
Event event = new Event();
List<Person> persons = new ArrayList<Person>();
persons = (ArrayList<Person>)
sess.createCriteria(Person.class).list();

event.setTitle("Third Event");event.setDate(new
Date(System.currentTimeMillis()));
event.setPersons(new HashSet<Person>());
for(Person p : persons){
    event.getPersons().add(p);
```

```
}

sess.save(event);
sess.getTransaction().commit();
```

Baris ini sebenarnya adalah implementasi JTA (Java Transaction API), yang membuat setiap proses transaksi harus *dicommit* untuk disetujui untuk direkam kedalam database, bilamana tidak, secara otomatis akan di *rollback*.

Baris diatas bilamana diartikan adalah sebagai berikut:

1. Transaksi dimulai dengan begin()
2. Masukan ArrayList dengan isi dari database yaitu person
3. Isi variable title dan date pada Event
4. Lakukan *looping* untuk semua isi dari persons (ArrayList)
5. Tambahkan setiap user kedalam object Event
6. Rekam semuanya dengan perintah save(event)
7. Commit semua kegiatan ini.

Jalankan beberapa kali HibernateManyToMany diatas, kemudian cobalah *script* dibawah ini. Coba buka *table* event, pilihlah salah satu id, kemudian ganti isian ff88xxx dengan id yang terdapat di *table* event. Jalankan.

```
Event e = (Event) sess.get(Event.class,
    "ff8080810e205e45010e205e47f00001");
for(Person p : event.getPersons()) {
    System.out.println(p.getFirstname());
}
```

Coba jalankan dengan beberapa id yang berbeda, hasilnya berbeda bukan. Kita tahu *table* Event tidak memiliki relationship *record* atau *foreign key* terhadap Person, dan begitu juga sebaliknya. Tetapi *resultnya terfilter* berdasarkan idnya bukan. Sebenarnya relationship *table* antara event dan person terdapat pada event_

person.

Yang hebat dari tipe ManyToMany dari JPA adalah bilamana ada objek bertipe ini, secara otomatis akan membuat table penghubungnya.

Coba ganti-ganti id didalam table baik event atau person, jalankan kembali. Apa yang terjadi? Error akan muncul.

```
No row with the given identifier exists: [org.blueoxygen.bbook2.Event#ff8080810e205e85010e205e878e0001]
```

Implementasi DBUnit sebagai ETL

Hibernate Tools khusus untuk menggenerate table, memiliki beberapa metode, dan secara default adalah menghapus yang ada, kemudian membuat yang baru. Dalam implementasinya, metode ini adalah bagus sekali bilamana kita tidak mau pusing dengan table yang degenerate, tetapi ternyata pada implementasinya sering terjadi ada table yang harus diisi terlebih dahulu. Malahan yang lebih tinggi lagi adalah data harus ditransfer dari satu database ke database lain, mengikuti independensi ORM yang digunakan, seperti yang pernah penulis alami, membuat sebuah implementasi berbasis Hibernate tentu saja, sistem dikembangkan dengan MySQL, client menggantinya menjadi SQL Server. Padahal MySQL sudah dijalankan untuk operasional. Yang lebih parah lagi, pada saat terjadi migrasi, ada beberapa struktur database yang berubah.

Sehingga yang terjadi adalah kita harus melakukan patch terhadap struktur table mengikuti objek Hibernate yang berubah, tetapi isi tetap ada.

Ada dua pendekatan yang dapat dilakukan tentu saja, yaitu menggunakan tools migrasi untuk memindahkan data dari MySQL ke SQL Server, kemudian objek Hibernate yang berubah dimap

dengan table-table di SQL Server, dan dirubah satu persatu.

Ternyata ada metode yang lebih baik, yaitu dengan mengimplementasikan ETL singkatan dari Extract Transform Language, sebuah mekanisme yang memungkinkan data di MySQL dipindahkan ke SQL Server atau database manapun bernama DBUnit. Sedangkan HibernateTools tetap diutilisasikan, sehingga Hibernate yang membuat struktur database, dan DBUnit yang mengisi table yang baru terbentuk, dan semua ini hanya dijalankan dengan satu perintah yaitu ant.

Adapun untuk melakukan integrasi migrasi ini dapat dilakukan dengan menggunakan build.xml yang telah kita kembangkan, kemudian ditambahkan dengan task dbunit.

Tentu saja ide ini dapat diimplementasikan bilamana kita mengembangkan produk dengan Hibernate, artinya pihak client dapat menentukan databasenya apa saja, dan Hibernate akan mengenerate table, dan isinya oleh DBUnit, untuk semua database yang didukung Hibernate dan DBUnit tentu saja hal ini terjadi.

Untuk menggunakan DBUnit dalam Ant sebenarnya sangat mudah. Pertama yang paling penting adalah skema untuk export dari database ke format XML.

Scriptnya adalah sebagai berikut

```
<target name="export-data"
      description="Export data to xml file">

<taskdef name="dbunit"
      classpathref="classpath.build"
      classname="org.dbunit.ant.DbUnitTask"/>

<dbunit driver="${db.driver}"
      url="${db.url}"
      userid="${db.username}">
```

```
        password="${db.password}">
    <export dest="${db.exportdata}"
           format="xml"/>
</dbunit>
<echo message="Data exported"/>
</target>
```

Perintah ini berarti mengcopy semua data dari database (\${db.url}) menjadi metadata berformat xml. Adapun *output* dari menjalankan ini dalam Ant adalah

```
Buildfile: /home/frans/workspace/Hibernate/build.xml
export-data:
[dbunit] Executing export:
[dbunit] in format: xml to datafile: /home/frans/
workspace/Hibernate/dbunt-export.xml
[echo] Data exported
BUILD SUCCESSFUL
Total time: 2 seconds
```

XML dari kegiatan *export* ini dapat dikirim ke database lainnya, dan tentu saja harus JDBC compliance. Cara paling mudah adalah dengan menambahkan *script* berikut pada build.xml

```
<target name="import-data"
       description="Add some data to the database">
    <taskdef name="dbunit"
             classpathref="classpath.build"
             classname="org.dbunit.ant.DbUnitTask"/>

    <dbunit driver="${db.driver}"
           url="${db.url}"
           userid="${db.username}"
           password="${db.password}">
        <operation type="INSERT"
                   src="${dbtestdata}"
                   format="xml"/>
    </dbunit>
    <echo message="Data imported"/>
</target>
```

Bilamana import-data dijalankan oleh Ant, akan muncul output seperti ini, diluar ini artinya error.

```
Buildfile: /home/frans/workspace/Hibernate/build.xml
import-data:
[dbunit] Executing operation: INSERT
[dbunit] on  file: /home/frans/workspace/Hibernate/
all-db-data.xml
[dbunit] with format: xml
[echo] Data imported
BUILD SUCCESSFUL
Total time: 2 seconds
```

Sedangkan untuk membuat db.driver, db.url, db.username serta db.password dapat dikenal, buatlah sebuah file build.properties yang diletakan dilokasi yang sama dengan build.xml berada.

DBUnit sebenarnya dapat melakukan kegiatan yang lebih spesifik, seperti membandingkan data dengan isi database tujuan.

Pengembangan Hibernate Lanjutan

Sebenarnya pengembangan apliksi yang menggunakan ORM akan memiliki keflexisibilitasan terhadap database alias databasenya independent.

Untuk kasus diatas, hanya mengganti jdbc driver dari MySQL ke Oracle, dan mengganti *dialect* dari MySQLDialect menjadi OracleDialect, secara otomatis aplikasi kita berjalan pada database Oracle.

Kalau diperhatikan lebih lanjut, semua mekansime koneksi, bilamana mapping telah berhasil, adalah melakukan inisialisasi session, dan tentu saja apa yang terjadi bilamana aplikasi lebih kompleks, inisialisasi diinisialisasi disemua objek atau dibuat satu tetapi *shared*. Ini merepotkan bukan.

Pada pemograman berikutnya akan dibuat bagaimana mekanisme inisialisasi *session* baik itu SessionFactory maupun AnnotationConfiguration tidak dilakukan secara manual tetapi otomatis, ini adalah mekanisme yang disebut dengan IOC singkatan dari *Injection of Control*.

Tetapi bilamana tidak menggunakan IOC, session harus dikirim dari satu objek ke objek lain, hal ini terjadi bilamana mengimplementasikan Hibernate pada aplikasi desktop berbasis Swing pada bahasan mengenai projek Postila.

Bab-bab berikutnya akan menggunakan Hibernate secara lebih mendalam, tetapi tentu dengan implementasi IOC pada cimande workspace for Web, dan manual pada projek Postila dengan Swing.

BAB IV

Pemograman MVC dengan Contoller WebWork/Struts2

Pada bab sebelumnya dibahas bagaimana mekanisme JSP, Velocity dan Hibernate bekerja. Dengan JSP dan servlet Velocity, dapat dikembangkan aplikasi berbasis Java, tetapi kita juga mengetahui bahwa solusi tersebut belum yang terbaik. Bab ini akan menjelaskan teknologi paling penting dalam dunia MVC, yaitu *controller*. *Controller* ini adalah sebuah *layer* yang bekerja untuk mengurus urusan “antar layer”, yang artinya bertanggung jawab terhadap eksekusi aplikasi.

Sebenarnya ada banyak *controller* yang dapat digunakan, seperti JSF, Tapestry dari Apache, atau WebWork yang sekarang lebih dikenal dengan Struts 2.0. Yang mana bab ini akan mendalami teknologi berdasarkan WebWork, yang terkenal merupakan solusi MVC paling mudah didunia Java.

Sebelumnya ada Struts 1.x, yang mana adalah *framework* paling populer didunia Java. Tetapi tim Apache Struts telah setuju untuk membuat WebWork 2.3 menjadi Struts 2.0. Hal ini dilakukan karena dianggap Struts 1.x setelah ditinggal sang pembuatnya ke JSF, teknologinya mandek, sedangkan WebWork telah masuk ke teknologi berikutnya yang lebih modern.

Gabungan kekuatan nama Struts yang telah mendunia dan teknologi yang mengacu pada WebWork yang sangat stabil dan mature, diharapkan dapat mencounter gerakan JSF. Tetapi saat ini yang terjadi, bukan JSF melawan WebWork/Struts yang terjadi, tetapi Don Brown, salah satu commiter WebWork/Struts2 telah berhasil membuat component development didalam WebWork,

sehingga Struts2 sebenarnya adalah projek integrasi WebWork dengan JSF.

Beberapa fitur dari WebWork selain kemudahannya, diantaranya adalah telah terintegrasinya aplikasi yang umum kita perlukan seperti pembuatan Chart dengan JFreeChart, Reporting dengan Jasper, membuat XML untuk transformasi XHTML. Malahan teknologi AJAX yang baru saja direlease telah diintegrasikan kedalam WebWork sejak 2.2. Yang semua ini hanya dengan mengganti kata *result* dari html ke chart, atau xml didalam sebuah xml-nya.

Semoga setelah mendalami bab ini, dapat merasakan kekuatan WebWork.

Sekilas mengenai WebWork

WebWork adalah sebuah projek Java yang mengadopsi model pengembangan MVC (*Model – Viewer – Controller*), yang mana posisinya lebih tepat di area C dari MVC, yaitu *Controller*. WebWork merupakan implementasi dari servlet *dispatcher*, yang bekerja menghandle semua *request* dan *response* dari setiap akses Web.

WebWork diciptakan oleh Richard Oberg, salah satu orang jenius didunia Java dimuka bumi. Beberapa karyanya yang mendunia adalah JBoss dan XDoclet. Yang mana kedua teknologi ini dibahas juga didalam buku ini.

Versi 2.0 dari WebWork dikembangkan oleh Patrick Lightbody dan Jason Carreira, yang akhirnya membuat WebWork yang semua sebuah *project* Open Source Java sendirian, masuk ke OpenSymphony.com, sebuah host Open Source Java yang tentu saja tidak sebesar Apache, tetapi telah berhasil menghasilkan banyak perusahaan bernilai jutaan dolar, dengan mengadopsinya, seperti JiveSoftware di New York, Amerika dan Atlassian di

Sydney, Australia.

Walaupun sekarang web resmi WebWork telah migrasi ke Apache tepatnya <http://struts.apache.org>, tetapi tetap saja semua diskusi dan tips pengembangan WebWork didiskusikan di OpenSymphony, tepatnya di <http://www.opensymphony.com/webwork>.

Dalam dunia Java, WebWork adalah sebuah *sleeping elephant*, artinya sebuah teknologi yang sangat stabil, tetapi timnya kurang giat berpromosi, malah produk yang dikembangkan diatas WebWork yang lebih sering dipromosikan. Diantaranya adalah JiveForum, forum Java paling populer yang dipakai oleh java.sun.com, GE, Sony. Jadi kalau kita posting forum di Java.sun.com, secara langsung kita menggunakan WebWork.

Bagaimana WebWork Bekerja.

Inti dari WebWork adalah XWork, sebuah teknologi *controller generic*, yang dapat digunakan dalam solusi Web dan non Web, tetapi tentu saja yang solusi Web adalah yang paling mature dan stabil. Bilamana ingin mendalami bagaimana Xwork dikembangkan menjadi solusi berbasis Swing, dapat mendalami pendulum (<http://pendulum.sf.net>).

WebWork dapat dikatakan web *wrapper*nya XWork, sehingga XWork dapat berjalan didalam lingkungan solusi Web atau servlet.

Untuk mengerti bagaimana WebWork bekerja sebenarnya, kita hanya perlu mengerti bagaimana sebuah POJO dari Java bekerja. POJO yang singkatan dari *Plain Old Java Object* adalah mekanisme Java “mau-mau kita”, yang dalam implementasinya adalah dengan mengimplementasikan get atau set. Yang mana ini adalah mekanisme standar yang harus diketahui dalam pemogramaan Java.

Contohnya adalah berikut

```
package org.blueoxygen.bbook;
import com.opensymphony.xwork.Action;

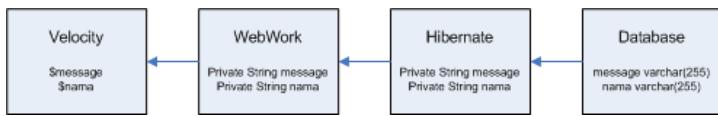
public class HelloWorld implements Action {
    private String message;
    public String execute() {
        message = "Hello, World!\n";
        message += "The time is:\n";
        message += System.currentTimeMillis();
    return SUCCESS;
}

public String getMessage() {
    return message;
}

public void setMessage(String message) {
    this.message = message;
}
}
```

Contoh diatas adalah satu *action* dari WebWork, perbedaannya dengan POJO adalah memiliki sebuah method *execute()*, dan mengimplementasikan Action dari xwork.

Setiap variable didalam sebuah POJO yang mengimplementasikan Action, secara otomatis akan dikenal pada *presentation layernya* MVC yang mengimplementasikan WebWork. Mekanisme ini berlaku saat kita memmapping hibernate kedalam MVC.



Mekanisme Interaksi MVC (Cimande)

Dengan konsep ini, secara otomatis, Anda telah mengimplementasikan MVC. Mudah bukan. Itulah WebWork.

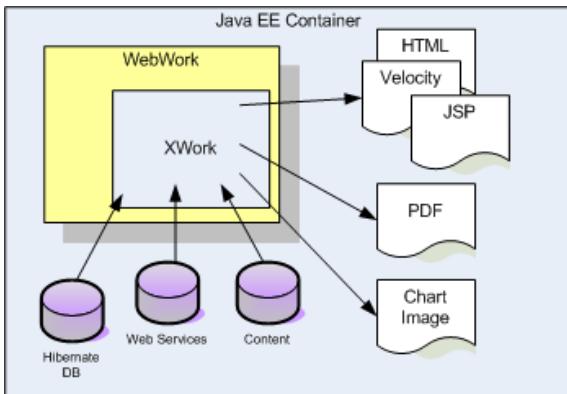
Setelah itu, kita hanya perlu mendaftarkan *object* yang telah kita buat, yang kita sebut WebWork Action ini kedalam file konfigurasinya yaitu xwork.xml.

Contohnya adalah seperti ini:

```
<!DOCTYPE xwork PUBLIC "-//OpenSymphony Group//XWork
1.0//EN"
"http://www.opensymphony.com/xwork/xwork-1.0.dtd">
<xwork>
<include file="webwork-default.xml"/>
<package name="action"
    extends="webwork-default"
    namespace="/action">
<action name="helloJSP"
    class="org.blueoxygen.bbook.HelloWorld">
    <result name="success">helloworld.jsp
    </result>
</action>
<action name="helloVelocity"
    class="org.blueoxygen.bbook.HelloWorld">
    <result name="success"
        type="velocity">helloworld.vm</result>
</action>
<action name="helloFreemarker"
    class="org.blueoxygen.bbook.HelloWorld">
    <result name="success"
        type="freemarker">helloworld.ftl</result>
</action>
<action name="hello"
    class="org.blueoxygen.bbook.Hello">
    <result name="success"
        type="velocity">hello.vm</result>
    <result name="input"
        type="velocity">helloworld.vm</result>
</action>
</package>
<package name="descriptor"
    extends="webwork-default">
```

```
    namespace="/descriptor">
<action name="create"
      class="org.blueoxygen.bbook.Form">
    <result name="success"
      type="velocity">add.vm</result>
</action>
<action name="add"
      class="org.blueoxygen.bbook.SaveForm">
    <result name="success"
      type="velocity">simpan.vm</result>
    <result name="error"
      type="velocity">gagal.vm</result>
</action>
</package>
</xwork>
```

Bilamana kita hendak mengganti *viewer layer* dari JSP ke Velocity, tambahkan type=velocity, atau bilamana hendak melakukan redirecting ke halaman berikutnya ganti dengan type=redirect, ini artinya setelah proses dilakukan web akan *forward* ke url lainnya. Gantilah dengan chart, jasper bilamana *outputnya* ingin PDF. Tentu saja didalam baris method execute(), kita harus menambahkan beberapa *script* yang memungkinkan *output* menjadi PDF.



Mekanisme Transformasi MVC (Cimande)

Adapun bentuk hasil akhir (*result*) default dari script diatas dengan JSP adalah

```
<%@ taglib prefix="ww" uri="webwork" %>
<html>
<head>
<title>Hello Page</title>
</head>
<body>
The message generated by my first action is:
<ww:property value="message"/>
</body>
</html>
```

Sedangkan bilamana menggunakan Velocity adalah

```
<html>
<head>
<title>Hello Page</title>
</head>
<body>
The message generated by my first action is:
$message
</body>
</html>
```

Terlihat velocity lebih simple. Bilamana kita menambahkan variable didalam Action, maka secara langsung akan dikenal dalam *presentation layernya*. Coba bandingkan dengan *script* Velocity dibab sebelumnya. Kita harus membuat sebuah *variable* implementasi dari Java Collection seperti hashmap, atau *object* Java lainnya, kemudian dimasukan kedalam Context, dan variable context yang diparsing didalam *presentation layer*. Dengan WebWork, semua tidak diperlukan lagi, semua telah diproses secara otomatis.

Sedangkan bilamana kita menambahkan *variable* transformasi mengembalikan *variable* sebuah String seperti “Dua ratus lima

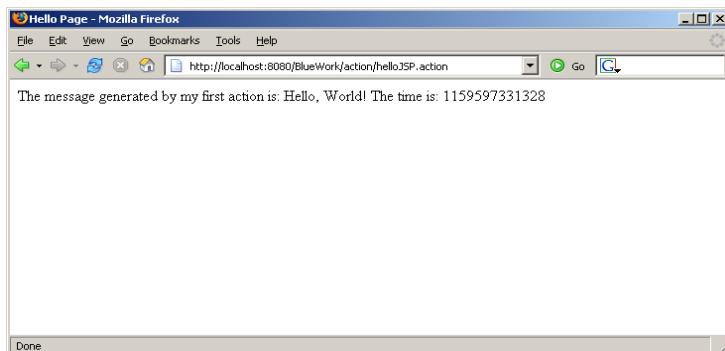
puluhan rupiah”, hanya perlu dibuat sebuah *variable* baru saja, dan diisi *variable* tersebut dengan value yang diperlukan.

Masalah sering terjadi bilamana yang direturn adalah null, karena secara default akan mereturn “null” di layer presentation. Ini tentu saja mengganggu, coba pakai perintah !\$message.

Sedangkan untuk memanggil code yang telah *termap* antara presentation dan xwork.xml, adalah dengan memanggil *code* mengikuti *namespacenya*.

Lihat baris berikut, yang terdapat pada file xwork.xml

```
<package name="action" extends="webwork-default"
namespace="/action">
<action name="helloJSP" class="org.blueoxygen.bbook.
HelloWorld">
    <result name="success">helloworld.jsp</
result>
</action>
```



Output eksekusi WebWork

Cara mengeksekusinya adalah dengan mengetik `http://localhost:8080/BlueWork/helloJSP.action`.

Setiap action didalam xwork, akan harus diberikan akhiran .action. Yang artinya menjalankan objek org.blueoxygen.bbook.HelloWorld. Bilamana eksekusi berhasil, akan melempar semua hasil eksekusi ke file helloworld.jsp.

Hal ini berlaku bagi semua tipe result dari WebWork.

Membuat Aplikasi MVC dengan Result JasperReport

WebWork telah terintegrasi baik sekali dengan JasperReport, dan ini artinya memungkinkan dikembangkan sebuah *output* sesuai dengan fitur dalam JasperReport.

Wrapper yang terintegrasi ini memungkinkan membuat aplikasi reporting tanpa harus pusing melakukan initialisasi pada jaspernya, tetapi WebWork melalui Xwork langsung menggabungkan semua menjadi satu hanya dengan menulis type="jasper" pada xwork.xml, dan intergasi terjadi

```
<result name="success" type="jasper">
    <param name="location">foo.jasper</param>
    <param name="dataSource">mySource</param>
    <param name="format">CSV</param>
</result>
```

Format standar dari result bertipe jasper adalah PDF, tetapi memungkinkan menghasilkan result berdasarkan JasperReportConstants yang terdapat pada JasperReport.

Bilamana hendak mengimplementasikan DataSource dapat menambahkan parameter

```
<result name="success" type="jasper">
    <param name="location">foo.jasper</param>
    <param name="dataSource">mySource</param>
</result>
```

Membuat Aplikasi MVC dengan Result Chart menggunakan JfreeChart

WebWork telah datang dengan integrasi dengan JfreeChart. JfreeChart adalah Open Source Charting yang mungkin terbaik di Internet.

Sayangnya sampai WebWork 2.x dikeluarkan, integrasi dengan JfreeChart belum seperti integrasi dengan teknologi lainnya, baik itu JSP, Velocity, Freemarker ataupun JasperReport. JfreeChart harus dimasukan kedalam code didalam *execute*, kemudian dilempar ke *result*.

Berikut ini adalah *setting* xwork pada salah satu projek didalam BlueOxygen Dashboard, yang merupakan Open Source untuk Balanced Scorecard, salah satu aplikasi Performance Management berbasis Java yang Open Source dari BlueOxygen.

```
<package name="chart"
    extends="default" namespace="/module/chart">
<action name="viewChart"
    class="org.blueoxygen.cimande.dashboard.chart.
ViewChart1">
    <result name="success" type="chart">
        <param name="width">800</param>
        <param name="height">300</param>
    </result>
</action>
<action name="create"
    class="org.blueoxygen.cimande.dashboard.chart.
ChartForm">
    <result name="success">view.vm</result>
</action>
<action name="filter"
    class="org.blueoxygen.cimande.dashboard.chart.
ChartForm">
    <result name="success">view.vm</result>
</action>
</package>
```

Sedangkan untuk menampilkannya hanya perlu menambahkan action di img pada html.

```

```

Adapun source code dari dashboard.chart.ViewChart1 adalah:

```
JFreeChart chart = ChartFactory.createXYBarChart(
    md.getName(), //Titel
    "Month", //Beschriftung X-Achse
    true, //Datumsachse
    "Performance Index (%)",
    dataset, //Datensatz
    PlotOrientation.VERTICAL, //Ausrichtung
        true, //Legende
        true, //Tooltips
        false //URLs??
);
XYPlot plot = chart.getXYPlot();

plot.setBackgroundPaint(Color.white);
plot.setDomainGridlinePaint(
    Color.lightGray);
plot.setRangeGridlinePaint(
    Color.lightGray);
DateAxis axis =
    (DateAxis) plot.getDomainAxis();
axis.setDateFormatOverride(
    new SimpleDateFormat("MMM"));
final XYBarRenderer renderer =
    (XYBarRenderer) plot.getRenderer();
renderer.setDrawBarOutline(false);

// set up gradient paints for series...
final GradientPaint gp0 = new GradientPaint(
    0.0f, 0.0f, Color.red,
    0.0f, 0.0f, Color.white
);
final GradientPaint gp1 = new GradientPaint(
    0.0f, 0.0f, Color.yellow,
    0.0f, 0.0f, Color.white
```

```
    );
    final GradientPaint gp2 = new GradientPaint(
        0.0f, 0.0f, Color.green,
        0.0f, 0.0f, Color.white
    );

    renderer.setSeriesPaint(0, gp0);
    renderer.setSeriesPaint(1, gp1);
    renderer.setSeriesPaint(2, gp2);
    renderer.setDrawBarOutline(true);

return chart;
```

Untuk lebih lengkapnya dapat melihat projek Balanced Scorecard didalam BlueOxygen.org.

Ada lagi satu teknologi yang mungkin sangat bagus didalam WebWork, yaitu validasi. Berita terbaru dari validasi ini, salah satu pembuat WebWork yaitu Jason Carreira, telah menjadi salah satu expert group dari spesifikasi validasi, jadi mungkin saja kedepannya di dalam JCP akan ada teknologi validasi yang bekerja mirip dengan validasi didalam WebWork. Bilamana realisasi terjadi, bukan tidak mungkin WebWork dan JSF memiliki validasi yang sama.

Untuk membuat sebuah validasi sebenarnya tidaklah sulit, kita hanya perlu membuat sebuah file xml dengan aturan yang sama dengan nama class ditambah kata validation.

Contoh validasi yang didapat pada projek Cimande Thin 1.0 terdapat AddModule yang merupakan file implementasi dari Action, dan sebuah file AddModule-validation.xml

Berikut adalah isi dari file AddModule-validation.xml

```
<!DOCTYPE validators PUBLIC "-//OpenSymphony Group//  
XWork Validator 1.0.2//EN" "http://www.opensymphony.
```

```
com/xwork/xwork-validator-1.0.2.dtd">

<validators>
    <field name="module.name">
        <field-validator type="requiredstring">
            <message>You must enter a name</message>
        </field-validator>
    </field>
    <field name="module.nameSpace">
        <field-validator type="requiredstring">
            <message>NameSpace Cannot be empty</
message>
        </field-validator>
    </field>
</validators>
```

Dimana type="requiredstring" merupakan proses validasi dari class `com.opensymphony.xwork.validator.validators.RequiredStringValidator`.

Type dari validasi ini merupakan referensi dari file validators.xml yang harus ditemukan didalam folder /WEB-INF/classes.

Bilamana hendak membuat sendiri validasinya, dapat melihat source code dari WebWork, Validator berada dalam `com.opensymphony.xwork.validator.validators`.

Jangan lupa memasukan baris `<interceptor-ref name="validation"/>`, pada package action, berikut adalah contohnya:

```
<action name="add" class="org.blueoxygen.
cimande.thin.module.action.AddModule">
    <interceptor-ref name="params"/>
    <interceptor-ref name="validation"/>
    <interceptor-ref name="component"/>
    <interceptor-ref name="workflow"/>
    <result name="success"
type="redirect">create.action</result>
    <result name="input"
```

```
type="redirect">>create.action</result>
</action>
```

WebWork Lanjutan – Interceptor

Telah dibahas sebelumnya, penggunaan *interceptor* pada setting xwork.xml. Ternyata sebenarnya *interceptor* itu ada banyak didalam xwork yang siap kita gunakan.

Interceptor adalah sebuah mekanisme yang memungkinkan sebuah objek dijalankan sebelum proses atau setelah proses Action. Interceptor bekerja seperti filter sebenarnya pada pemograman servlet, dimana kalau di JavaEE filter disetup di web.xml, sedangkan *interceptor* didalam WebWork diletakan diantara action-name.

Contohnya adalah

```
<package name="default" extends="webwork-default">
<result-types>
<result-type name="chart"
class="org.blueoxygen.cimande.dashboard.chart.
ChartResult"/>
</result-types>
<interceptors>
<interceptor-stack name="defaultComponentStack">
<interceptor-ref name="component"/>
<interceptor-ref name="model-driven"/>
<interceptor-ref name="validationWorkflowStack"/>
</interceptor-stack>

<interceptor-stack name="chainingComponentStack">
<interceptor-ref name="defaultComponentStack"/>
<interceptor-ref name="chain"/>
</interceptor-stack>
</interceptors>

<default-interceptor-ref
      name="defaultComponentStack"/>
</package>
```

Berikut adalah diagram interceptor yang disediakan dan siap digunakan:

Interceptor	Name	Description
Alias Interceptor	alias	Converts similar parameters that may be named differently between requests.
Chaining Interceptor	chain	Makes the previous action's properties available to the current action. Commonly used together with <result type="chain"> (in the previous action).
Component Interceptor	component	Enables and makes the components available to the Actions. Refer to components.xml
Conversion Error Interceptor	conversionError	adds conversion errors from the ActionContext to the Action's field errors
Create Session Interceptor	createSession	Creates an HttpSession automatically, useful with certain interceptor (e.g. TokenInterceptor) when an HttpSession is required in order to work properly
Execute and Wait Interceptor	executeAndWait	An interceptor that executes the action in the background and then sends the user off to an intermediate waiting page.
Exception Interceptor	exception	Maps exceptions to a result.
File Upload Interceptor	fileUpload	An interceptor that adds easy access to file upload support. See the javadoc for more info
I18n Interceptor	i18n	remembers the locale selected for a user's session
Logger Interceptor	logger	Outputs the name of the action
Model Driven Interceptor	model-driven	If the action implements ModelDriven, pushes the getModel() result onto the value stack.
Parameters Interceptor	params	Sets the request parameters onto the action.
Prepare Interceptor	prepare	If the action implements Preparable, calls its prepare() method.
Scope Interceptor	scope	simple mechanism for storing action state in the session or application scope
Servlet Config Interceptor	servlet-config	Give access to HttpServletRequest and HttpServletResponse (think twice before using this since this ties you to the Servlet API)
Static Parameters Interceptor	static-params	Sets the xwork.xml defined parameters onto the action. These are the <param> tags that are direct children of the <action> tag.
Timer Interceptor	timer	Outputs how long the action (including nested interceptors and view) takes to execute
Token Interceptor	token	Checks for valid token presence in action, prevents duplicate form submission
Token Session Interceptor	token-session	Same as above, but storing the submitted data in session when handed an invalid token
Validation Interceptor	validation	Performs validation using the validators defined in [Action]-validation.xml
Workflow Interceptor	workflow	Calls the validate method in your action class. If action errors created then it returns the INPUT view.
Parameter Filter Interceptor	N/A	Removes parameters from the list of those available to actions etc.

BAB V

Pengembangan dengan IoC Menggunakan Spring Framework

Bab sebelumnya kita telah membahas apa itu MVC dan komponen pendukungnya. Serta kombinasi dari semua komponen, untuk membentuk sebuah teknologi MVC yang terintegrasi. Ada 4 kombinasi yang mungkin dikembangkan yang dibahas pada bab pertama. Ini diluar dengan komponen lainnya yang belum dibahas. Malah dengan implementasi teknologi ini, membuat kombinasi semuanya menjadi satu, teknologi tersebut disebut Injection of Control (IoC), yang mana bab ini akan membahas Spring Framework, IoC paling populer didunia, yang mana setelah bab ini, IoC akan menjadi inti dari aplikasi MVC yang kita kembangkan.

Mengapa IoC Penting dalam Aplikasi?

Sebelum kita membahas lebih dalam, kita harus kembali ke awal dari semua masalah, yaitu sintak bahasa pemrograman. Bilamana kita telah menguasai semua sintaks bahasa pemrograman, apakah kita langsung bisa membuat aplikasi?

Belum tentu bukan! Ada pengetahuan lain yang dibutuhkan dalam pembuatan aplikasi. Di antaranya adalah bagaimana kita akan mengatur pengelompokan berbagai file di dalamnya. Sebagai contoh, kita harus memikirkan pengaturan hal-hal berikut :

*** Konfigurasi.**

Pengaturan koneksi database adalah salah satu hal dasar yang harus dapat dikonfigurasi. Dengan semakin fleksibelnya aplikasi, akan lebih banyak hal lain yang dapat dikonfigurasi

* **Bahasa atau terjemahan.**

Aplikasi yang baik dapat diterjemahkan dengan mudah. Cukup dengan mengedit satu file tertentu, semua tulisan di aplikasi, termasuk pesan *error*, dapat berubah sesuai pilihan bahasa.

* **Library pelengkap.**

Di jaman sekarang, sangat jarang aplikasi yang 100% dibuat sendiri. Pasti ada beberapa fungsi yang diambil dari internet melalui pustaka yang berbayar maupun gratis atau open source. Penempatan library ini harus sesuai agar bisa dikenali

* **Susunan source code.**

Kerapian susunan *source code* akan memudahkan kita apabila nantinya harus memperbaiki bug atau menambah fitur.

Masih banyak lagi hal-hal yang harus dipertimbangkan dalam membuat aplikasi. Ini membutuhkan pengalaman untuk dapat menentukan pengaturan yang baik.

Salah satu teknik untuk mengatur ketergantungan antar kode program adalah menggunakan teknik yang disebut *Inversion of Control* (IoC), atau juga dikenal dengan istilah *Dependency Injection* (DI). Konsep ini cukup abstrak, sehingga cara terbaik untuk memahaminya adalah langsung dengan kode program.

Misalnya kita membuat aplikasi toko *online*. Tentunya kita ingin menyimpan data produk, data pesanan, dan juga data pelanggan. Berikut adalah struktur data untuk Produk, memuat data id, nama produk, dan harga.

```
public class Product {  
    private Integer id;  
    private String name;  
    private BigDecimal price;  
  
    public Integer getId() {  
        return this.id; }  
    public String getName() {
```

```
        return this.name; }
public BigDecimal getPrice() {
    return this.price; }

public void setId(Integer id) {
this.id = id; }
public void setName(String name) {
this.name = name; }
public void setPrice(BigDecimal price) {
this.price = price; }
}
```

Sesuai dengan kesepakatan umum, masing-masing *property* diberikan akses private, kemudian disediakan *method* untuk membaca dan mengubah *property* tersebut (getter dan setter).

Tentunya kita ingin menyimpan data produk ke database. Praktek yang umum adalah mendefinisikan fungsi untuk mengelola data produk tersebut menggunakan database. Operasi yang biasa dilakukan adalah simpan data (*Create*), ambil data (*Read*), edit data (*Update*), dan hapus data (*Delete*). Operasi ini biasa disebut dengan istilah CRUD. Berikut adalah kode program yang mendefinisikan operasi CRUD.

```
public interface ProductDao {
    public void create(Product prod);
    public void update(Product prod);
    public void delete(Product prod);
    public Product getById(Integer id);
    public List<Product> getAll();
}
```

Untuk fleksibilitas aplikasi, kita buat *interface*, yang kemudian akan diimplementasikan di class lain. Manfaat dari pendekatan ini adalah untuk memudahkan kita bila nanti ingin mengganti implmentasi. Contoh nyatanya akan kita lihat nanti. Sementara ini, mari kita implementasikan *interface* di atas dengan JDBC biasa. Kita akan menggunakan PreparedStatement untuk

mengoptimasi *query* dan mencegah SQL Injection. Lebih lanjut tentang penggunaan PreparedStatement dapat dilihat di *tutorial* Java tentang JDBC. Ini adalah kode sebelum kita menggunakan teknik IoC.

```
public class ProductDaoJdbc implements ProductDao {  
    public void create(Product prod) {  
        String sql = "INSERT INTO tbl_product  
(name, price) VALUES (?,?)";  
  
        PreparedStatement stm = dbConn.  
prepareStatement(sql);  
        stm.setString(prod.getName());  
        stm.setBigDecimal(prod.getPrice());  
        stm.executeUpdate();  
    }  
}
```

Dalam kode di atas, ada satu variabel yang kita gunakan, yaitu dbConn. Variabel ini mencerminkan koneksi ke database dengan konfigurasi tertentu. Misalnya konfigurasinya adalah sebagai berikut:

- * Database Server : localhost
- * Jenis Database : mysql
- * Nama Database : latihan
- * Nama Driver : com.mysql.jdbc.Driver
- * Username : belajar
- * Password : rahasia

Kode untuk menginisialisasi dbConn adalah sebagai berikut :

```
DataSource dataSource = new MysqlDataSource();  
dataSource.setServerName("localhost");  
dataSource.setDatabaseName("latihan");  
dataSource.setUser("belajar");  
dataSource.setPassword("rahasia");  
Connection dbConn = dataSource.getConnection();
```

Pertanyaannya adalah, di mana kita harus meletakkan kode inisialisasi dbConn tersebut? Pendekatan yang paling sederhana adalah di dalam method create(Product p) sebelum *query* dieksekusi, seperti ini :

```
public void create (Product prod) {  
    DataSource dataSource = new MysqlDataSource();  
    dataSource.setServerName("localhost");  
    dataSource.setDatabaseName("latihan");  
    dataSource.setUser("belajar");  
    dataSource.setPassword("rahasia");  
    Connection dbConn =  
        dataSource.getConnection();  
    String sql = "INSERT INTO tbl_product  
        (name, price) VALUES (?,?)";  
    PreparedStatement stm =  
        dbConn.prepareStatement(sql);  
    stm.setString(prod.getName());  
    stm.setBigDecimal(prod.getPrice());  
    stm.executeUpdate();  
    dbConn.close();  
}
```

Ya, pendekatan ini dapat memecahkan masalah, sementara. Bagaimana kalau kita mengimplementasikan method selanjutnya? Misalnya method delete(Product p). Apakah kita akan menyalin semua kode di atas dan mengubah sedikit?

```
public void delete (Product prod) {  
    DataSource dataSource = new MysqlDataSource();  
    dataSource.setServerName("localhost");  
    dataSource.setDatabaseName("latihan");  
    dataSource.setUser("belajar");  
    dataSource.setPassword("rahasia");  
    Connection dbConn =  
        dataSource.getConnection();  
  
    String sql = "DELETE FROM tbl_product  
        WHERE product_id=?";  
    PreparedStatement stm =
```

```
        dbConn.prepareStatement(sql);
        stm.setInt(prod.getId());
        stm.executeUpdate();
        dbConn.close();
    }
```

Walaupun kode di atas bisa dijalankan, tapi masih jauh dari optimal. Pertama, ada duplikasi informasi di sana. Semua informasi tentang koneksi database diulang di dua tempat. Akan lebih banyak perulangan kalau kita mengimplementasikan *method* lainnya, yaitu update, getById, dan getAll. Apalagi kalau kita membuat kode untuk mengelola tabel pelanggan dan pemesanan.

Bayangkan kalau kita harus mengubah *username* dan *password* untuk database. Akan banyak sekali perubahan di berbagai tempat dalam kode program. Sangat mungkin kita melupakan satu diantaranya, sehingga aplikasi kita *error*.

Kedua, eksekusi program yang tidak efisien. Pada kode di atas, aliran program berjalan seperti ini:

1. Buat koneksi ke database
2. Lakukan query
3. Putuskan koneksi ke database

Koneksi ke database merupakan operasi yang ‘mahal’. Artinya, untuk dapat tersambung ke database banyak sekali kegiatan dan waktu yang dibutuhkan. Aplikasi harus mencari server database, setelah itu berbincang-bincang mengenai database yang akan digunakan, pemeriksaan username dan password, dan membuat kesepakatan tentang protokol yang digunakan. Semua perbincangan ini membutuhkan waktu yang tidak sebentar. Waktu setengah detik saja akan terasa lama untuk ukuran komputer. Bayangkan kalau semua kegiatan ini harus diulangi setiap kali *query* dijalankan. Sungguh suatu pemborosan yang tidak perlu.

Semua inefisiensi di atas dapat diatasi dengan menggunakan

teknik IoC. IoC, atau Inversion of Control, adalah teknik untuk memindahkan kontrol atas object ke tempat lain. Untuk contoh kita, object yang dimaksud adalah dbConn. Mengapa dbConn? Coba perhatikan kode di atas. Fungsi utama class ProductDaoJdbc adalah mengelola object Product dan tabel tbl_product. Dengan demikian, kode paling penting dari sudut pandang ProductDaoJdbc hanyalah empat baris kode berikut :

```
String sql = "DELETE FROM tbl_product  
        WHERE product_id=?";  
PreparedStatement stm = dbConn.  
prepareStatement(sql);  
stm.setInt(prod.getId());  
stm.executeUpdate();
```

Sedangkan baris kode lainnya, hanyalah pelengkap saja. Akan lebih baik kalau kode pelengkap ini kita pindahkan ke tempat lain. Sehingga di dalam *method delete* hanya tertinggal empat baris yang penting tersebut.

Bagaimana cara memindahkan kode pelengkap tersebut? Pertanyaan ini dijawab oleh nama lain teknik IoC, yaitu *Dependency Injection* (DI). Terjemahan bebas dari *Dependency Injection* adalah “suntikan ketergantungan”. Suntikan? Ketergantungan? Waduh, kelihatannya mengerikan.

Tenang, ini tidak ada kaitannya dengan narkoba. Coba kita empat baris di atas. Ada ketergantungan terhadap object dbConn. Object ini harus ada agar empat baris kode itu bisa dijalankan. Daripada dbConn kita inisialisasi sendiri, akan lebih baik kalau kita ‘suntikkan’ saja ke dalam class ProductDaoJdbc. Bagaimana caranya?

Ada beberapa teknik untuk menyuntik *object*, yaitu melalui *constructor*, *method setter*, dan langsung ke *property* yang bersangkutan. Caranya tidak sesulit yang kita bayangkan, cukup

berikan *object* dbConn sebagai *parameter*. *Constructor Injection* kodenya seperti ini :

```
public class ProductDaoJdbc {  
    private DataSource dataSource;  
    public ProductDaoJdbc(DataSource ds) {  
        this.dataSource = ds;  
    }  
    // method lainnya tidak ditulis  
}
```

Setter injection sebagai berikut:

```
public class CustomerDaoJdbc {  
    private DataSource dataSource;  
    public setDataSource(DataSource ds) {  
        this.dataSource = ds;  
    }  
    // method lainnya tidak ditulis  
}
```

Dengan menjadikan dbConn sebagai *instance variable*, kita dapat menggunakannya di mana saja di dalam class ProductDaoJdbc. Lalu muncul pertanyaan selanjutnya, kapan *object* dbConn disuntikkan? Jawabannya adalah, pada saat aplikasi baru dijalankan. Coba kita ambil contoh sederhana, menggunakan method main.

```
public class DemoIoC {  
    public static void main(String[] args) {  
        DataSource ds = initDataSource();  
        ProductDao prodDao =  
            new ProductDao(ds);  
        // constructor injection  
        // atau  
        CustomerDao custDao =  
            new CustomerDao();  
        custDao.setDataSource(ds);  
        // setter injection
```

```
    }

    private DataSource initDataSource() {
        DataSource dataSource =
            new MysqlDataSource();
        dataSource.setServerName("localhost");
        dataSource.setDatabaseName("latihan");
        dataSource.setUser("belajar");
        dataSource.setPassword("rahasia");
    }
}
```

Seperti kita lihat, sebenarnya teknik IoC atau DI ini sangat mudah diimplementasikan. Jangan tertipu dengan istilahnya yang terkesan rumit. Sebetulnya konsep dibelakangnya sangat sederhana. Walaupun demikian, dengan teknik sederhana ini, kode program kita bisa jadi lebih rapi dan mudah dibaca.

Selanjutnya, setelah kita memahami konsep IoC, kita akan melihat penggunaan *framework* IoC. Ada beberapa *framework* yang memudahkan kita dalam mengimplementasikan IoC, diantaranya adalah:

- * Spring Framework
- * HiveMind
- * Pico Container

Dari berbagai *framework* yang ada, Spring Framework adalah yang paling populer. Ini disebabkan karena selain IoC, Spring juga menyediakan berbagai pustaka untuk memudahkan kita membuat program. Konsep IoC sendiri diperkuat dengan penambahan fitur Aspect Oriented Programming sehingga aplikasi kita menjadi lebih modular. Pembahasan lebih lanjut tentang AOP akan diberikan pada lain kesempatan.

Mari kita lihat bagaimana Spring mengimplementasikan IoC. Kembali ke method main. Dengan menggunakan Spring, kodanya menjadi seperti ini:

```
public class DemoIoC {  
    public static void main(String[] args) {  
        ApplicationContext spring =  
            new FileSystemXmlApplicationContext(  
                "spring-config.xml");  
  
        ProductDao prodDao =  
            (ProductDao) spring.getBean("productDao");  
        CustomerDao custDao =  
            (CustomerDao) spring.getBean("customerDao");  
    }  
}
```

Inisialisasi dan injeksi dilakukan di dalam file `spring-config.xml` yang isinya sebagai berikut

```
<beans>  
    <!-- productDao menggunakan  
        constructor injection -->  
    <bean id="productDao"  
        class="tutorial.ioc.ProductDaoJdbc">  
        <constructor-arg ref="dataSource"/>  
    </bean>  
  
    <!-- customerDao menggunakan setter injection -->  
    <bean id="customerDao"  
        class="tutorial.ioc.CustomerDaoJdbc">  
        <property name="dataSource"  
            ref="dataSource"/>  
    </bean>  
  
    <!-- dataSource -->  
    <bean id="dataSource"  
        class="com.mysql.jdbc.jdbc2.optional.  
MysqlDataSource">  
        <property name="serverName"  
            value="localhost"/>  
        <property name="databaseName"  
            value="latihan"/>  
        <property name="user" value="belajar"/>  
        <property name="password"
```

```
        value="rahasia"/>
    </bean>
</beans>
```

Kalau hanya melihat fitur IoC-nya saja, kita akan segera berkesimpulan bahwa Spring hanyalah framework untuk menggantikan *constructor* belaka. Object yang tadinya diinisialisasi dalam kode Java secara manual, dipindahkan ke XML dan diinisialisasi oleh Spring. Sebetulnya tidak demikian. Dengan menggunakan Spring, kita bisa ‘menempelkan’ kode *lifecycle* pada *object* kita. Kita dapat menentukan bahwa ada kode lain yang dijalankan pada saat :

- * Object diinisialisasi
- * Property tertentu diisi nilainya.
- * Object dihancurkan

Spring juga cukup cerdas dalam menentukan tipe data. Sebagai contoh, coba lihat kode program untuk mendeskripsikan pelanggan di toko online kita.

```
public class Customer {
    private Integer id;
    private String name;
    private Date birthdate;
    private List<String> hobbies =
        new ArrayList<String>();
    // getter dan setter seperti biasa
}
```

Dalam kode Java biasa, kita akan menginstankannya seperti ini:

```
Customer cust = new Customer();
cust.setId(10);
cust.setName("Endy");
cust.setBirthdate(new SimpleDateFormat("dd-MM-
yyyy").parse("17-08-1945"));
cust.getHobbies().add("Membaca");
cust.getHobbies().add("Berenang");
```

Dengan Spring, kita akan menulis seperti ini:

```
<beans>
    <bean id="cust" class="tutorial.ioc.Customer">
        <property name="id" value="10"/>
        <property name="name" value="Endy"/>
        <property name="birthdate" value="17-08-1945"/>
        <property name="hobbies">
            <list>
                <value>Membaca</value>
                <value>Berenang</value>
            </list>
        </property>
    </bean>

    <bean id="customEditorConfigurer"
          class="org.springframework.beans.factory.
config.CustomEditorConfigurer">
        <property name="customEditors">
            <map>
                <entry key="java.util.Date"
                      value-ref="dateEditor" />
            </map>
        </property>
    </bean>

    <bean id="dateEditor"
          class="org.springframework.beans.
propertyeditors.CustomDateEditor">
        <constructor-arg>
            <bean class="java.text.SimpleDateFormat">
                <constructor-arg value="dd-MM-yyyy" />
            </bean>
        </constructor-arg>
        <constructor-arg value="true" />
    </bean>
</beans>
```

Demikianlah sekilas mengenai teknik *Inversion of Control* atau *Dependency Injection*. Seperti kita telah lihat, cara ini dapat

membuat aplikasi kita menjadi lebih rapi, karena keterkaitan antar object dapat dikumpulkan di satu tempat.

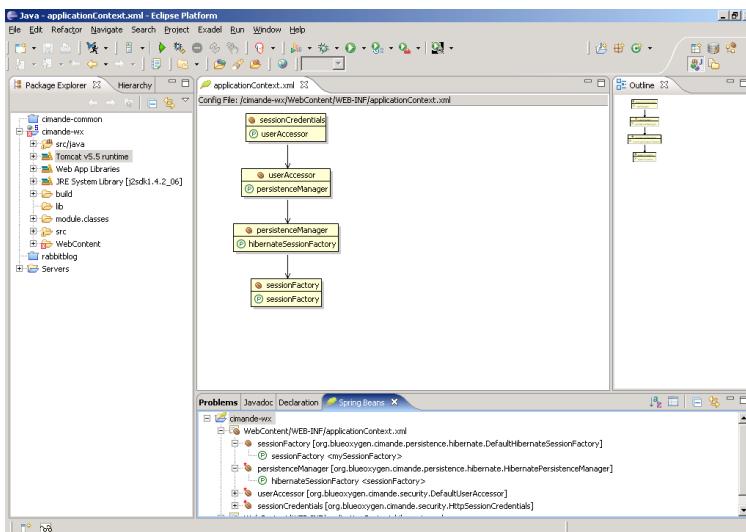
Spring Framework adalah salah satu *framework* IoC yang tersedia di pasaran. Selain sebagai *framework* IoC, Spring juga memiliki fitur lainnya, diantaranya:

- * AOP Framework
- * MVC Framework
- * Integrator untuk *framework* lain (Hibernate, TopLink)
- * Pustaka pembantu untuk memudahkan akses terhadap fitur J2EE seperti JMS, JMX, JNDI
- * Pustaka untuk menyediakan remoting service seperti RMI, Hessian, Burlap, dan SOAP

Sekilas Tentang SpringIDE

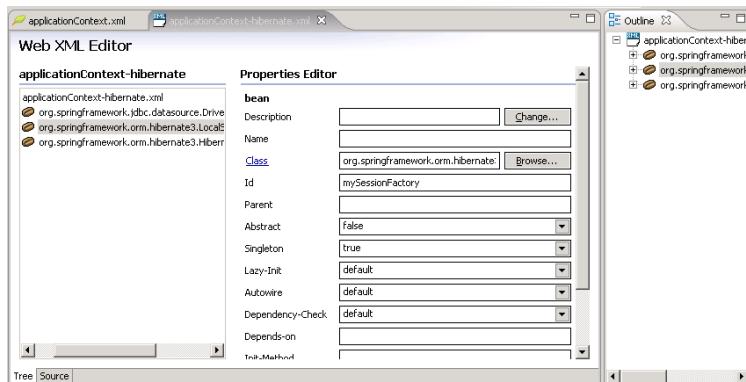
SpringIDE adalah sebuah plugins dari Eclipse. SpringIDE memungkinkan applicationContext.xml divisualisasikan dengan baik.

Fitur yang diperlukan dalam pengembangan IOC adalah adanya grafik visualisasi dari *application contextnya* spring yang dapat dilihat dibawah ini. Grafik ini membantu sekali untuk mengetahui hubungan antara bean didalam Spring Bean.



Eclipse Calisto yang telah terinstall SpringIDE

Penulis menambahkan Exadel 4.0 kedalam Eclipse Calisto, untuk menggunakan XML editor dari Exadel yang lebih baik daripada XML Editornya Calisto.



ApplicationContext Spring yang dibuka dengan XML Editor dari Exadel.

Menginstall SpringIDE

Buku ini telah mencoba menginstall SpringIDE, dan didapat sedikit trik untuk menginstallnya yaitu menginstall GEF 1.0 SDK secara manual. Setelahnya hanya memerlukan menambahkan lokasi dari SpringIDE untuk *download* secara kedalam Eclipse.

Bekerja dengan SpringIDE

Berikut adalah *application context* projek Cimande, yang mengimplementasikan IOC untuk Hibernate Session Factory dan SessionCredential untuk implementasi pada Security Filter. Security Filter akan dibahas pada bab 7 pada buku ini.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE beans PUBLIC "-//SPRING//DTD BEAN//EN"
"http://www.springframework.org/dtd/spring-beans.
dtd">

<beans >
    <bean id="sessionFactory"
        class="org.blueoxygen.cimande.persistence.
hibernate.DefaultSessionFactory"
        init-method="init"
        destroy-method="destroy"
        singleton="true">
        <property name="sessionFactory"
            ref="mySessionFactory"/>
    </bean>

    <bean id="persistenceManager"
        class="org.blueoxygen.cimande.persistence.
hibernate.HibernatePersistenceManager"
        init-method="init"
        destroy-method="dispose"
        singleton="false">
        <property name="hibernateSessionFactory"
            ref="sessionFactory"/>
    </bean>
```

```
<bean id="userAccessor"
      class="org.blueoxygen.cimande.security.
DefaultUserAccessor"
      singleton="false">
    <property name="persistenceManager">
      <ref bean="persistenceManager"/>
    </property>
  </bean>

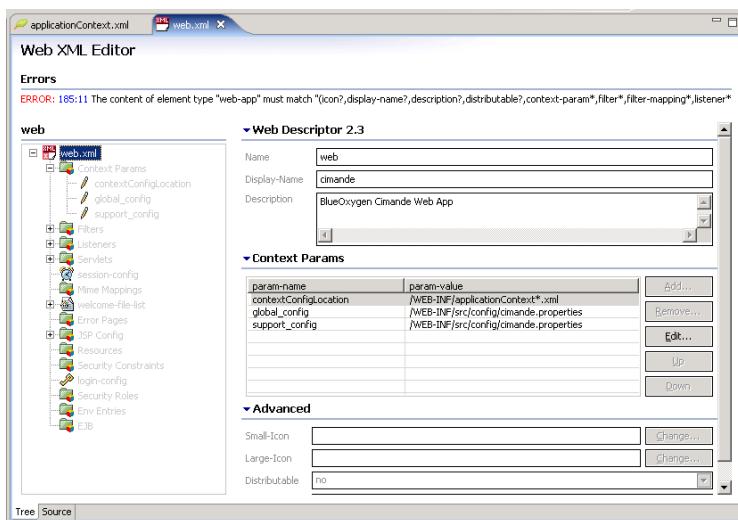
<bean id="sessionCredentials"
      class="org.blueoxygen.cimande.security.
 HttpSessionCredentials"
      singleton="false">
    <property name="userAccessor">
      <ref bean="userAccessor"/>
    </property>
  </bean>
</beans>
```

Lokasi dari applicationContext.xml diatas berada pada *folder* WEB-INF. ApplicationContext.xml ini yang akan menjadi kasus yang akan aktif secara otomatis saat Java EE *container* diaktifkan.

Dimana settingnya terdapat pada file web.xml pada folder WEB-INF juga. Inilah tag xmlnya untuk mengaktifkan applicationContext.xml didalam sebuah projek Java.

```
<context-param>
  <param-name>contextConfigLocation</param-name>
  <param-value>/WEB-INF/applicationContext*.xml</
param-value>
</context-param>
```

Atau bilamana melihat melalui *tree view*nya XML Editor adalah sebagai berikut:

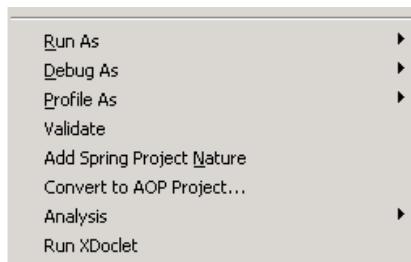


web.xml yang dibuka dengan XML editor

Jadi jangan kaget bilamana setelah bab ini, semua kegiatan persistence yang digunakan oleh buku ini yaitu Hibernate secara otomatis aktif, dan beberapa keajaiban muncul. Inilah kehebatan *Injection of Control*.

Untuk membuat sebuah projek Eclipse mendukung Spring adalah dengan membuat projek yang dibuka diakui sebagai projek berbasis Spring. Tanpa setting ini, *view* Spring Beans tidak dapat digunakan.

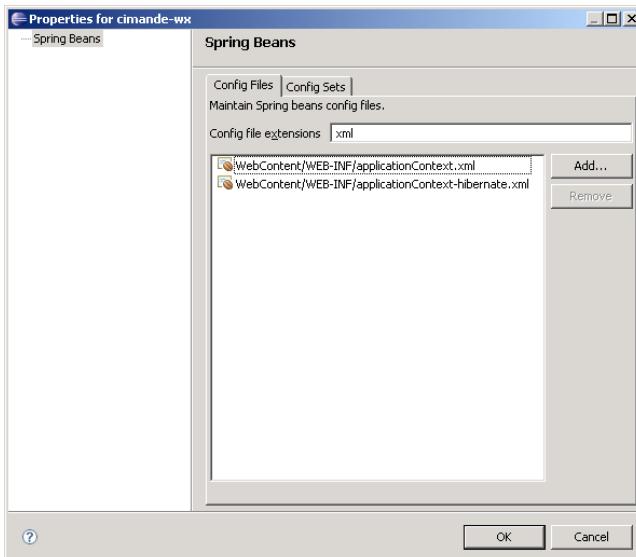
Caranya dengan meklik kanan nama projek yang akan *disetting*, lalu menambahkan Spring Project Nature kedalam projek tersebut.



Dialog untuk mengaktifkan projek mendukung Spring

Bilamana telah diset sebagai Spring Project Nature, maka projek Java tersebut otomatis akan dianggap projek Spring.

Untuk membuat view Spring Beans dapat digunakan, tambahkan applicationContext.xml kedalam *dialog boxnya*.

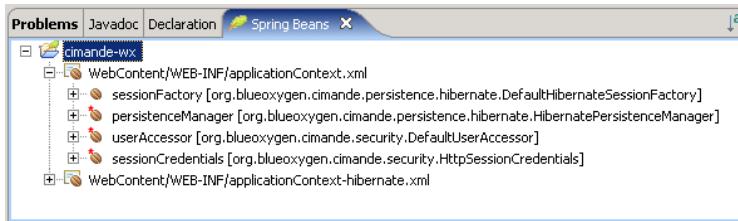


Dialog untuk memasukan application context.

XML ApplicationContext yang dimasukan kedalam Spring

Beans.

Spring Bean mendukung lebih dari satu XML. Bilamana telah selesai maka didalam *view* Spring Beans, semua bean akan tervisualisasi.



View Spring Beans dengan application context yang aktif

Untuk melihat hubungan antara *bean* didalam *application context*nya Spring dapat mengklik kanan terhadap *applicationContext* yang diinginkan.

Maka grafik hubungan *bean* akan muncul:

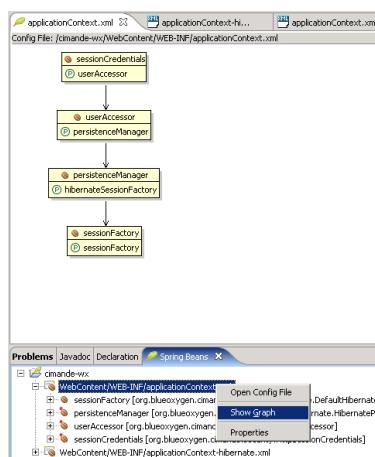


Diagram Hubungan untuk application context

Harap diperhatikan, bilamana melakukan klik 2x, maka yang muncul adalah XML Editor, sedangkan bilamana klik kanan, dapat memilih antara Show Graph untuk melihat diagram hubungan, dan membuka dengan XML Editor.

Berikut adalah diagram hubungan untuk applicationContext-hibernate.xml, yang digunakan projek Cimande untuk mengaktifkan persistence Hibernate.

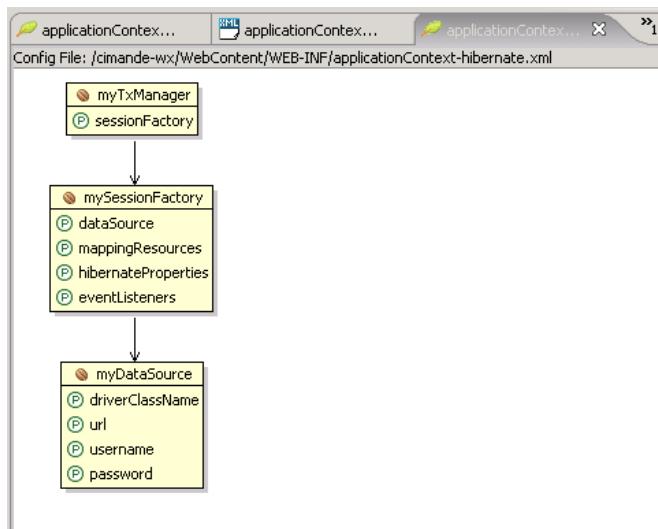
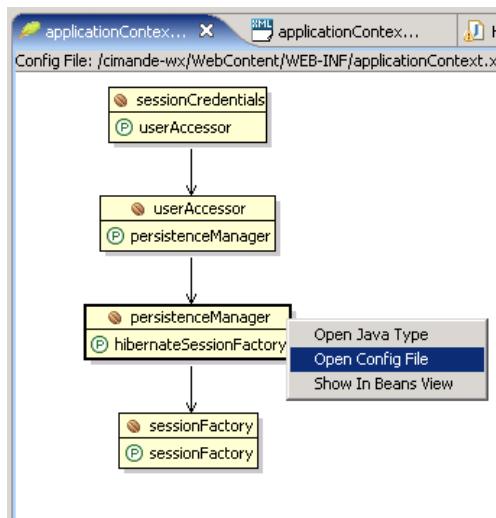


Diagram Hubungan applicationContext-hibernate.xml

Diagram hubungan pada SpringIDE, bilamana diklik 2x, maka akan mengaktifkan XML Editor, sedangkan untuk mengetahui dimana sebuah bean didalam applicationContext didaftarkan, harus mengklik kanan.



Dialog pada Diagram Hubungan

Bilamana klik kanan telah dilakukan ada 3 pilihan yaitu Open Java Type, Open Config File, dan Show In Beans View.

Open Java Type adalah kegiatan yang sama dengan mengaktifkan Java Editor, sedangkan Open Config File adalah membukanya dengan XML Editor, sedangkan Show In Beans View, adalah melink dengan view Spring Beans.

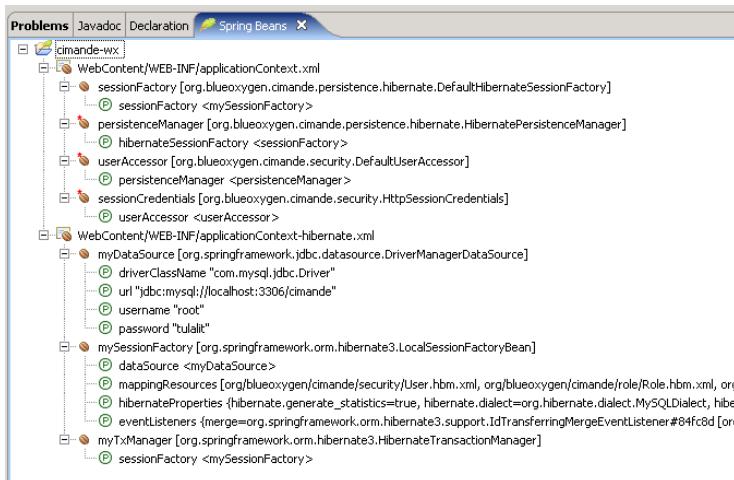
Hubungan Hierarki dalam Spring Beans.

Application context Spring adalah sebuah XML, sehingga dalam implementasinya memungkinkan dilakukan hubungan antara satu *bean* dengan *bean* lain didalamnya.

Sayang sekali, belum ada *editor* untuk *property* didalam applicationContext, sehingga untuk melihat lebih jelas isi dari sebuah beans ini harus menggunakan XML Editor.

Berikut adalah *drill down* semua applicationContext yang

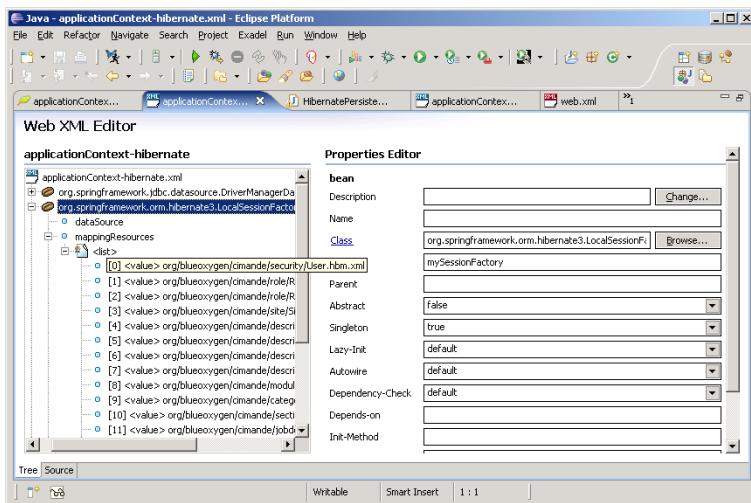
digunakan oleh projek Cimande.



View Spring Beans dengan detail application context

Sedangkan bilamana org.springframework.orm.hibernate3.LocalSessionFactoryBean dipilih maka didalam XML Editor kita dapat melihat apa saja yang ada didalamnya.

Cara Cepat Mengembangkan Aplikasi Java dengan Metode MVC



Tree XML editor yang termapping dengan DTD

Cobalah geser *mouse* kedalam <list> yang ada, maka kita dapat melihat isi dari propertiesnya.

<list> sebenarnya adalah kumpulan *parameter* didalam propertiesnya applicationContext.

Berikut adalah isi dari list didalam applicationContext-hibernate.xml

```
<list>
    <value>org/blueoxygen/cimande/security/User.hbm.xml</value>
    <value>org/blueoxygen/cimande/template/TemplateObjectDetail.hbm.xml</value>
    ...
    <value>org/blueoxygen/cimande/template/Template.hbm.xml</value>
</list>
```

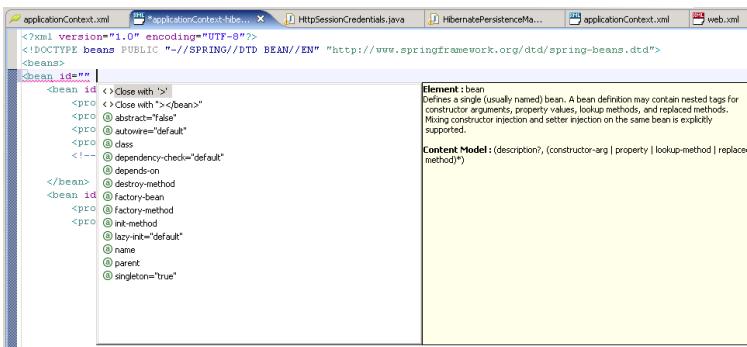
Terdapat 17 mapping Hibernate didalamnya. Mekanisme

applicationContext-hibernate sebenarnya adalah pilihan didalam implementasi pada Cimande, apakah hendak menggunakan hibernate.cfg.xml, atau menginject hibernate.cfg.xml kedalam applicationContext.

Bekerja dengan XML Editor (Exadel)

Spring IDE yang terintegrasi dengan baik dengan XML Editor didalam Eclipse, walaupun XML Editor yang dipakai penulis adalah XML Editor dari Exadel, ternyata tag-tag xml didalam SpringIDE telah terintegrasi.

Berikut adalah sebuah *dialog* yang muncul bilamana kita hendak membuat sebuah <bean> baru.



Tag pilihan pada saat editing xml.

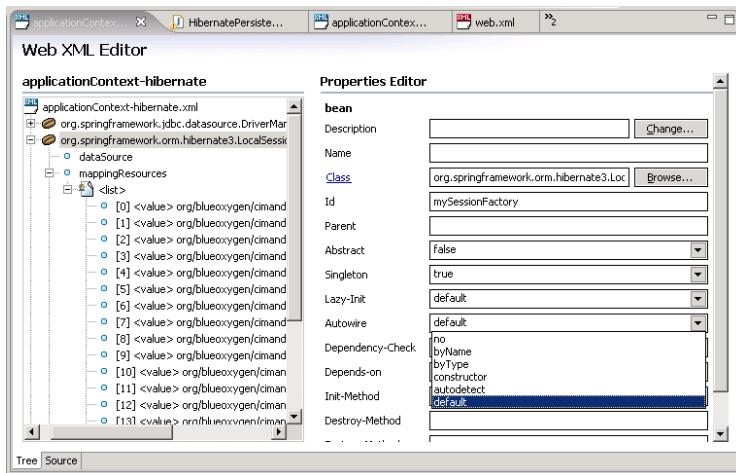
Sebenarnya rahasia semua ini terdapat pada satu tag yang tertulis diatas *application context*.

```

<!DOCTYPE beans PUBLIC "-//SPRING//DTD BEAN//EN"
"http://www.springframework.org/dtd/spring-beans.
dtd">
  
```

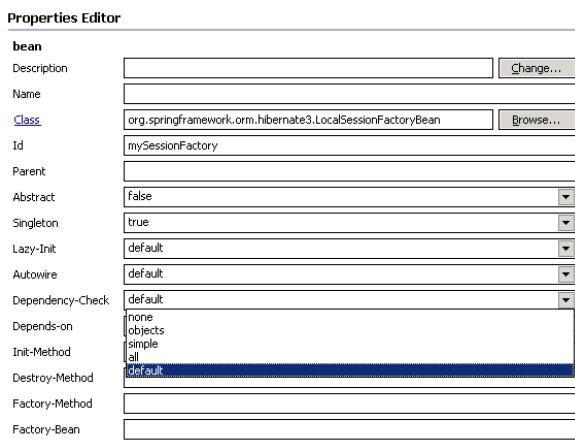
Bilamana telah membuat sebuah *bean* didalam *application context* yang terbuka, dapat pindah ke modus “tree”, dan secara otomatis semua isian dari tag-tag xml aktif. Kita dapat mengganti semua

isian dari *namespace* didalam XML yang ada.



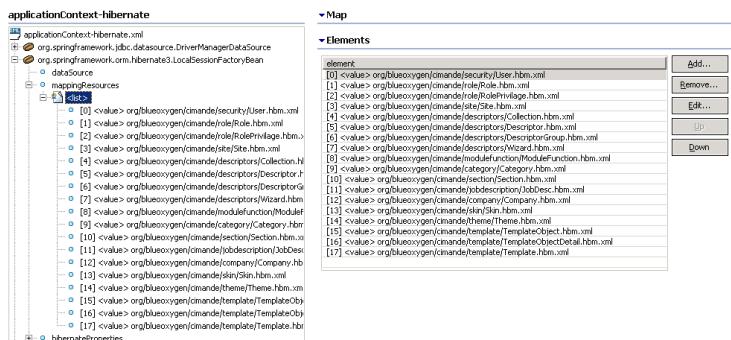
Pilihan autowire pada application context.

Berikut adalah *properties editor* yang ada didalam XML Editor.



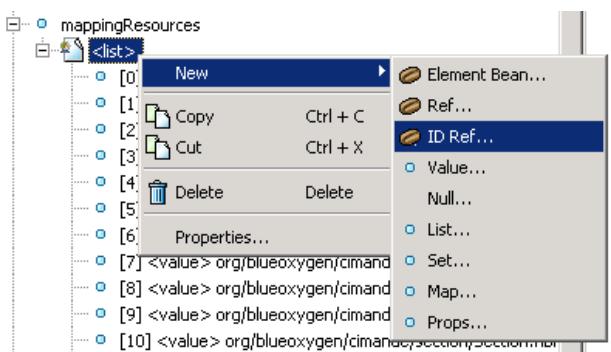
Isian untuk tipe Depedency Class

Bilamana implementasinya adalah berbentuk *list*, maka akan muncul sebuah *element grid* yang berisikan *value* didalam *list* tersebut.

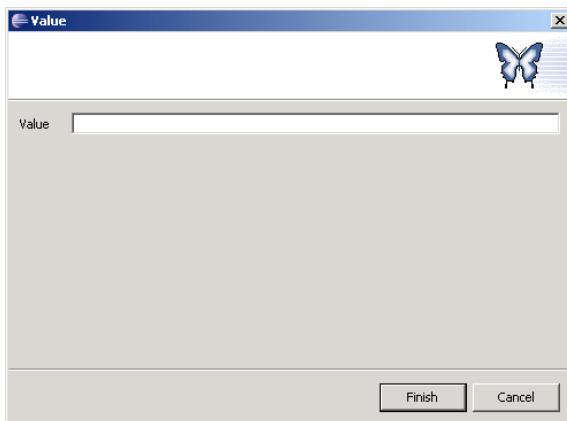


Element didalam list

Bilamana kita hendak menambahkan *value* didalam list tersebut tinggal klik kanan, maka akan muncul *dialog box*.

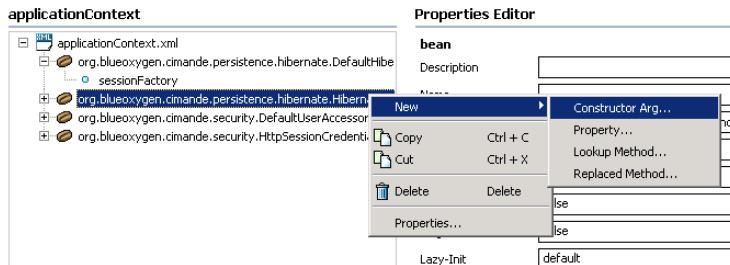


Penambahan *value* untuk *list*



Dialog untuk mengisi value

Sedangkan bilamana kita mengklik kanan langsung dari *beannya* akan muncul *dialog* untuk membuat Constructor, Property, Lookup Method atau Replaced Method.



Dialog untuk menambahkan isi dari sebuah beans

Sayang sekali semua implementasi dari New didalam Spring IDE ini hanya seperti isian *value*.

Untuk lebih jelas bagaimana cara kerja semua isian ini, dapat melihat pada DTD dari Spring.

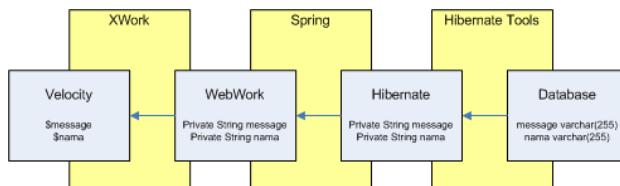
BAB VI

Membuat Aplikasi Web CRUD dengan Cimande

Bilamana bab sebelumnya kita telah membahas teknologi pendukung MVC, serta di bab pertama dijelaskan mengenai kemungkinan kombinasi untuk implementasi MVC.

Bab ini akan menjelaskan bagaimana membuat sebuah aplikasi dengan implementasi CRUD (*Create, Read, Update and Delete*) yang lebih umum dapat dikatakan membuat aplikasi untuk membuat content baru, melakukan *search/pencarian*, melakukan pengeditan data, serta menghapus data.

Adapun *mapping* standar setiap C, R, U dan D adalah dengan melakukan *data flow mapping* dari database ke Velocity sesuai dengan gambar berikut.



Arsitektur Projek Cimande dari BlueOxygen

Dimana metode ini adalah mengacu pada projek Cimande Thin dari BlueOxygen, project yang dikembangkan oleh tim diperusahaan penulis. Cimande Thin adalah contoh kecil dari fondasi solusi, dimana Cimande Thin adalah *framework* integrasi antara Velocity/ JSP, WebWork dengan Hibernate, hubungan WebWork dengan Hibernate menggunakan Spring. Pemetaan WebWork dengan

Velocity diurus oleh XWork. Untuk menghubungkan database dengan hibernate dapat menggunakan Hibernate Tools yang terintegrasi dengan Ant atau menggunakan Hibernate Tools yang merupakan bagian dari JbossIDE. Dimana, mekanisme Hibernate dengan database telah dibahas pada bab 3.

Jadi dapat dikatakan bab ini adalah merupakan skema integrasi dari teknologi MVC yang telah dibahas dibab sebelumnya, menjadi satu kesatuan, yang oleh penulis disebut Cimande.

Versi lengkap yang dikembangkan disebut Cimande WX, yang akan dibahas dibab setelah ini, yaitu bagaimana membuat CRUD yang dikembangkan dibab ini diintegrasikan dengan kombinasi filter dan session, sehingga CRUD hanya dapat dijalankan bilamana telah melewati proses login. Teknologi *filter* dan *session* akan dibahas setelah bab ini.

Sekilas Tentang Projek Cimande

Bagian ini akan menjelaskan bagaimana mengembangkan sebuah aplikasi berbasis MVC, yang menggunakan Velocity sebagai presentation layer, WebWork sebagai *controller*, serta Hibernate sebagai Modelnya. Sedangkan integrasi antara WebWork dengan Hibernate menggunakan Spring telah disediakan oleh Cimande, termasuk juga integrasi WebWork dengan Velocity telah diintegrasikan oleh Xwork, yang mana contoh pengembangan WebWork dengan Velocity telah dijelaskan pada bab 4.

Untuk membuat sebuah aplikasi MVC, sebenarnya sama dengan yang dijelaskan pada bab 4, tetapi yang membedakan adalah adanya implementasi PersistenceAware, yang merupakan sebuah mapping insialisasi objek yang lebih populer disebut dengan IoC, atau singkatan dari Injection of Control.

Mekanisme ini sebenarnya adalah sebuah cara sehingga kita tidak perlu melakukan inisialisasi dalam membuat session

untuk Hibernate. File yang mengurusi semua ini adalah file ApplicationContext.xml, yang merupakan mapper pada Spring.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE beans PUBLIC "-//SPRING//DTD BEAN//EN"
"http://www.springframework.org/dtd/spring-beans.
dtd">

<beans>

<bean id="sessionFactory"
      class="org.blueoxygen.cimande.persistence.
hibernate.DefaultSessionFactory"
      init-method="init"
      destroy-method="destroy"
      singleton="true"/>

<bean id="persistenceManager"
      class="org.blueoxygen.cimande.persistence.
hibernate.HibernatePersistenceManager"
      init-method="init"
      destroy-method="dispose"
      singleton="false">
    <property name="hibernateSessionFactory">
      <ref bean="sessionFactory"/>
    </property>
  </bean>
</beans>
```

XML diatas merupakan implementasi dari SpringBean, dimana pada implementasinya setiap objek yang hendak terhubung dengan SessionFactory dari Hibernate, harus mengimplementasikan PersistenceManager.

Bilamana dibab sebelumnya dibahas mengenai bagaimana mengimplementasikan Action pada WebWork, ini artinya secara otomatis WebWork akan terhubung dengan tipe result, baik itu Velocity, JSP, JasperReport ataupun JFreeChart. Sedangkan untuk menghubungkan antara WebWork dengan Hibernate diperlukan

implementasi PersistenceAware.

Berikut adalah contohnya:

```
public class DescriptorForm extends ActionSupport  
implements PersistenceAware
```

Kemudian dalam setiap implementasinya harus mengimplementasi PersistenceManager, seperti berikut

```
public void setPersistenceManager(PersistenceManager  
persistenceManager) {  
    this.manager = persistenceManager;  
}
```

Logika CRUD Mengacu pada Skema Cimande

Dalam implementasi dengan MVC, terutama menggunakan WebWork, penulis selalu melakukan proses CRUD *cycle*, dan tentu saja dalam implementasi dalam pengembangannya sebaiknya setiap modul yang mengimplementasikan CRUD dibuat matrixnya seperti berikut

Module	C	R	U	D	J
PageCollection	X	X	X	X	
Business Partner	X	X	X	X	
Contact	X	X	X	X	
Purchase Order	X	X	X		X

CRUD Matrix

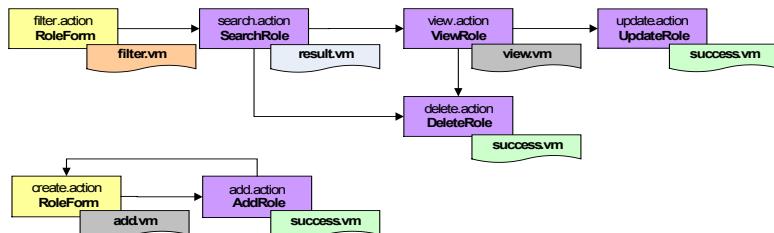
Matrix CRUD, umumnya penulis tambahkan J, artinya Jasper, ini adalah *report* untuk setiap transaksi CRUD.

Adapun beberapa mapping standard pengembangan CRUD yang dilakukan didalam projek penulis yaitu BlueOxygen adalah sebagai berikut:

	Naming	Action
C	New	new.action
R	Search	filter.action
U	Edit	edit.action
D	Delete	delete.action

CRUD - Action Mapping

Ada tambahan satu *action* yaitu view.action yang mana merupakan *result* dari filter.action. CRUD dan Action mapping bilamana diimplementasikan dengan *presentation layer* adalah sebagai berikut:



Alur CRUD dan template yang dipakai bersama.

Dalam pengembangan aplikasi CRUD menggunakan Cimande, sebenarnya template yang diperlukan tidak banyak, yaitu filter.vm, result.vm, view.vm, success.vm, dan add.vm. Kedepannya view.vm dan add.vm dapat digabung.

Berikut adalah diagram penghubung untuk kasus pengembangan role dengan MVC.

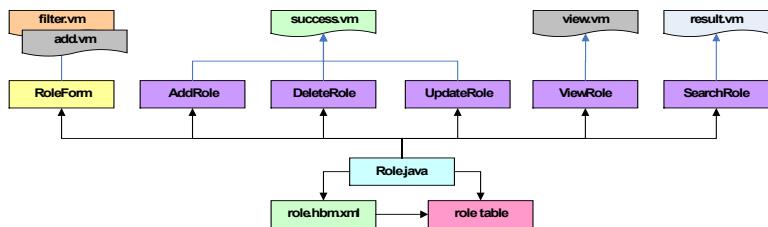


Diagram Implementasi Role antara Hibernate dengan Velocity menggunakan WebWork

Diagram diatas menjelaskan untuk berhubungan dengan table role, diperlukan sebuah file Role.java, dan bilamana mengimplementasikan Hibernate yang *core* bukan anonasi, diperlukan role.hbm.xml, bilamana menggunakan anonasi, hanya perlu mengimplementasikan JPA.

Sedangkan file RoleForm adalah file yang mengimplementasikan PersistenceAware, sedangkan file yang lain yaitu AddRole, DeleteRole, UpdateRole, ViewRole atau SearchRole semuanya merupakan turunan dari RoleForm.

Yang mana kita hanya perlu mereplace execute dengan kegiatan yang kita inginkan.

Sebagai contoh adalah untuk AddRole, isi dari execute adalah

```
public String execute() {
    Role role = new Role();

    if (getName().equalsIgnoreCase(""))
        addActionError("Name can't be empty.");
    if (getDescription().equalsIgnoreCase(""))
        addActionError(
            "Description can't be empty.");

    if (hasErrors())
        return INPUT;
```

```

} else {
    role.setName(getName());
    role.setDescription(getDescription());

    // logging information
    LogInformation log = new LogInformation();

    if (sessionCredentials.getCurrentUser()
    != null ) {
        log.setCreateBy(sessionCredentials
            .getCurrentUser()
            .getId());
        log.setLastUpdateBy(sessionCredentials
            .getCurrentUser()
            .getId());
    }

    log.setCreateDate(new
Timestamp(System.currentTimeMillis()));
    log.setLastUpdateDate(new
Timestamp(System.currentTimeMillis()));
    log.setActiveFlag(getActiveFlag());

    if (getActiveFlag() == -1) {
        log.setActiveFlag(
        LogInformation.INACTIVE);
    } else {
        log.setActiveFlag(getActiveFlag());
    }

    role.setLogInformation(log);
    pm.save(role);
    return SUCCESS;
} }

```

AddRole diatas kalau dilihat lebih dekat terdiri dari 3 area pengembangan, yang mana area pertama adalah validasi, area berikutnya adalah memasukan informasi user yang aktif ke dalam objek role, kemudian melakukan penyimpanan ke database.

Mekanisme pertama yaitu validasi, sebenarnya dapat diganti

dengan validasi yang ada di WebWork dengan melakukan implementasi interceptor validasi pada setiap *package action* yang hendak dilakukan validasi.

Sedangkan ViewRole adalah sebuah *action* untuk menampilkan isi dari sebuah *record*, untuk dilakukan proses *update*.

ViewRole dan SearchRole merupakan sebuah kegiatan yang melempar data dari Hibenate ke Velocity/JSP, sedangkan yang lainnya hanyalah sebuah kegiatan untuk berinteraksi dengan Hiberante.

Adapun code *execute* ViewRole adalah sebagai berikut:

```
public String execute() {
    role = (Role) pm.getById(Role.class, id);

    if (role == null) {
        addActionError(
            "Such role couldn't be found");
        return ERROR;
    } else {
        return SUCCESS;
    }
}
```

Sedangkan untuk SearchRole yang digunakan untuk filter.action adalah.

```
public String execute() {
try {

    sess = hsf.createSession();
    Criteria crit = sess.createCriteria(Role.class);
    if (!getName().equalsIgnoreCase("")) {
        crit.add(Expression.like("name", "%" +
            getName() + "%"));
    }
    if (!getDescription().equalsIgnoreCase("")) {
        crit.add(Expression.like("description", "%" +
            getDescription() + "%"));
    }
}
}
```

```

        getDescription()
        + "%"));
    }
    if (getActiveFlag() != -1) {
        crit.add(Expression.eq(
            "logInformation.activeFlag",
            new Integer(getActiveFlag())));
    }
    roles = crit.list();
    hsf.endSession(sess);

    return SUCCESS;
}

} catch (HibernateException e) {
    return ERROR;
} catch (SQLException e) {
    return ERROR;
} finally {
    try {
        hsf.closeSession(sess);
    } catch (HibernateException e1) {
        return ERROR;
    } catch (SQLException e1) {
        return ERROR;
    }
}
}
}

```

Adapun isi dari xwork.xml adalah sebagai berikut:

Untuk membuat form isian atau create.action

```

<action name="create"
       class="org.blueoxygen.cimande.role.actions.
RoleForm" >
    <result name="success"
           type="velocity">add.vm</result>
</action>

```

Untuk mengisi setiap hasil submit dari create.action

```

<action name="add"
       class="org.blueoxygen.cimande.role.actions.
AddRole">

```

```
<result name="success"
type="velocity">addSuccess.vm</result>
<result name="input" type="velocity">add.vm
</result>
</action>
```

Untuk menampilkan *form search* atau filter.action. Perhatikan perbedaannya dengan create.action, objeknya sama, tetapi *templatanya* berbeda.

```
<action name="filter"
       class="org.blueoxygen.cimande.role.actions.
RoleForm">
    <result name="success"
           type="velocity">filter.vm</result>
    <result name="error">/errors/errors.vm
    </result>
</action>
```

Untuk menampilkan hasil pencarian

```
<action name="search"
       class="org.blueoxygen.cimande.role.actions.
SearchRole">
    <result name="success"
           type="velocity">result.vm</result>
    <result name="error">/errors/errors.vm
    </result>
</action>
```

Untuk menampilkan sebuah *record* yang terpilih:

```
<action name="edit"
       class="org.blueoxygen.cimande.role.actions.
EditRoleLoad">
    <result name="success"
           type="velocity">edit.vm</result>
    <result name="error">/errors/errors.vm
    </result>
</action>
```

Untuk mengupdate perubahan terhadap data yang telah ditampilkan atau setelah submit dari edit.action

```
<action name="update"
       class="org.blueoxygen.cimande.role.actions.
EditRole">
    <result name="success"
           type="redirect">view.action?id=${id}</result>
    <result name="input"
           type="velocity">edit.vm</result>
    <result name="error">/errors/errors.vm
    </result>
</action>
```

Untuk menghapus *record* (dapat diakses langsung saat result dari pencarian)

```
<action name="confirmDelete"
       class="org.blueoxygen.cimande.role.actions.
ViewRole">
    <result name="success"
           type="velocity">confirmDelete.vm</result>
    <result name="error">/errors/notfound.vm
    </result>
</action>

<action name="delete"
       class="org.blueoxygen.cimande.role.actions.
DeleteRole">
    <result name="success"
           type="velocity">deleteSuccess.vm</result>
    <result name="error">/errors/notfound.vm
    </result>
</action>
```

Dari semua ini sebenarnya ada satu *link flow* yang hilang, yaitu bagaimana sebuah create.action dapat mengeksekusi add.action, dan filter.action dapat menghasilkan search.action, demikian juga dengan edit.action ke update.action, atau confirmDelete.action ke

delete.action.

Flow ini sebenarnya terdapat pada *form template*, yang mana pada kasus diatas menggunakan velocity semuanya.

Coba buka file add.vm, ini adalah *file* yang digunakan saat melakukan create.action, ada dua *area* yang dapat diperhatikan yaitu menampilkan error validasi dan *form* isian.

Berikut ini adalah cara menampilkan validasi.

```
#if (!$actionErrors.isEmpty())
<div class="errorMessage">Errors</div>
<ul class="errorMessage">
#foreach( $error in $actionErrors )
<li>$error</li>
#end
</ul>
#end
```

Sedangkan berikut ini adalah *form* untuk melakukan *action* berikutnya,

```
<form method="post" action="add.action">
<table border="0" cellspacing="5" cellpadding="5">
<tr id="tableHeader1">
    <td colspan="2">Role</td>
</tr>
<tr>
    <td>Name</td>
    <td><input type="text" name="name" value="$!name">
    </td>
</tr>
...
</table>
</form>
```

Perhatikan baris `<form method="post" action="add.action">`,

ini adalah *link flow* yang hilang, *form* ini artinya bilamana tombol submit diakses dilanjutkan dengan add.action.

Sedangkan *form* untuk edit.vm akan berisikan update.action, sedangkan filter.vm yang digunakan oleh filter.action akan memiliki instruksi `<form method="post" action="search.action">`.

Ini sesuai dengan diagram alur *flow* yang telah dijelaskan diawal bab ini bukan. Berikut adalah bentuk hasil dari new.action

Create Workflow's Role	
* Name:	<input type="text"/>
* Description:	<input type="text"/> <div style="border: 1px solid #ccc; padding: 5px; height: 200px;"> <p>Arial 1 (8 pt) Heading 1 B <i>I</i> <u>U</u> S </p> </div>
Path: body > html	
Parent ID:	<input type="text"/>
Active:	<input checked="" type="radio"/> Yes <input type="radio"/> No
<input type="button" value="Save"/> <input type="button" value="Reset"/>	

Output dari create.action

Berikut adalah bentuk hasil dari implementasi search.action.

Search Workflow Role	
Name:	<input type="text"/>
Description:	<input type="text"/>
Parent ID:	<input type="text"/>
Active:	<input type="radio"/> Yes <input type="radio"/> No
<input type="button" value="Search"/> <input type="button" value="Reset"/>	

Output dari filter.action

Sedangkan bilamana tombol Search ditekan, akan masuk ke result.action yang merupakan sebuah hasil dari pencarian.

Search Workflow Role		Found () role
Name	Description	Action
Journalist	Create the document	Edit Del
Editor	Edit the document	Edit Del
Gods	Gods	Edit Del
Approver	Approve the document	Edit Del
Publisher	Publish the document	Edit Del
Content Editor	Content Editor	Edit Del
Element Editor	Element Editor	Edit Del
Lotion	Lotion	Edit Del

result dari filter.action

Setiap hasil dari pencarian ada dua link yang dapat dilakukan mengikuti flow CRUDnya Cimande yaitu Edit untuk ke view.action dan Del untuk ke delete.action.

Bilamana Edit ditekan akan keluar form seperti create.action, tetapi tentu saja akan menampilkan isi dari *link* yang dipilih, sedangkan bilamana Del dipilih, akan muncul page konfirmasi yang akan memberikan konfirmasi untuk penghapusan. Konfirmasi ini dapat disebut sebagai delete.action.

Delete Workflow Role Confirmation:
<p>Are You sure Want to Destroy "Element Editor" Workflow Role?</p> <p style="color: red;">This Action cannot be Undone!!</p>
<input type="button" value="Delete"/> <input type="button" value="Cancel"/>

Konfirmasi penghapusan field

Proses CRUD diatas adalah fondasi yang selalu dilakukan oleh tim penulis untuk mengerjakan sebuah projek berbasis Web menggunakan Cimande.

BAB VII

Implementasi User Management dengan Filter dan Session

Memilih Session atau Cookies?

Setelah mendalami mengenai bagaimana MVC bekerja, serta melakukan implementasi pengembangan aplikasi mengacu pada CRUD. Langkah sebenarnya adalah dengan mengimplementasikan security terhadap solusi tersebut.

Bilamana pembaca adalah dari lingkungan pemograman embed HTML, yang artinya melakukan pemograman langsung di HTMLnya seperti ASP, PHP atau mungkin JSP, sering terjadi level keamanan diparsing.

Umumnya implementasi melakukan pelancakan terhadap *cookies* atau *session*. *Cookies* adalah mekanisme menyimpan sebuah format informasi, yang mana informasinya dapat disharing pada seluruh page dalam domain yang bersangkutan. Sedangkan *session* diciptakan bilamana informasi tersebut disimpan didalam server, dan umumnya masa aktifnya adalah 20 menit terhadap klikan terakhir.

Contoh implementasi dengan *session* yang sering dilakukan adalah

```
<%
if (session.getAttribute("GA_USER") !=null) {
%>
... code session active....  

<% } else {
```

```
%>  
... code session null  
<% } %>
```

Mekanisme diatas adalah pengecheckan terhadap *session* “GA_USER”, bilamana terdeteksi, baris diantara if dan else akan dieksekusi, sedangkan bilamana *session* adalah null, maka baris else akan dieksekusi.

Mekanisme ini untuk teknologi bersifat *parsing technology* akan terus diproses.

Sedangkan untuk menyimpan nilai didalam GA_USER, menggunakan perintah *session.setAttribute(“GA_USER”, object)*.

Session didalam Java ini sangat *powerfull* sebenarnya, ini yang membedakan dengan teknologi lain, *session* yang tersimpan bukan hanya satu kata tetapi sebenarnya adalah sebuah *container* dari objek Java. Yang mana artinya kita dapat memasukan objek dengan tipe *collection* seperti *HashMap*, *ArrayList* kedalam session.

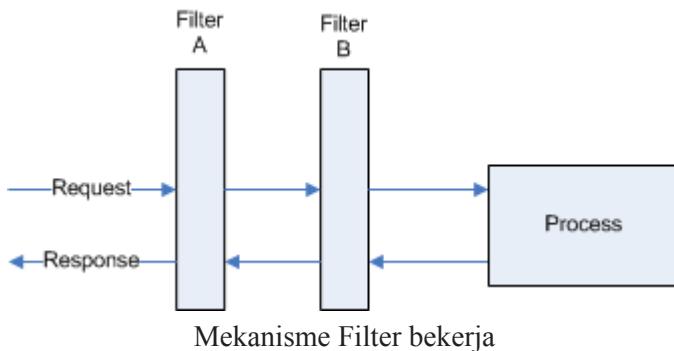
Contoh implementasi *session* dengan implementasi *security parsing* yang sangat kompleks dapat melihat pada folder /module/pagecollection dalam projek cimande.

Berkenalan dengan Filter

Proses pengecheckan *session* atau *cookies* didalam JSP, dengan metode if else, terkadang sangat merepotkan, terutama saat kita hendak mengganti variable *session security* kita, misalnya dari GA_USER menjadi GX atau isi dari *session* itu dari sebuah string username menjadi sebuah *token security user* yang mengimplementasikan X.509 yang sekarang sedang jadi *trend*.

Sebenarnya ada sebuah teknologi yang memungkinkan proses pengecheckan ini tidak perlu dilakukan, yaitu dengan mengimplementasikan javax.servlet.Filter.

Filter dapat dipasang pada web.xml, sehingga url yang dimap didalam web.xml, akan terlebih dahulu lewat filter tersebut.



Filter sebenarnya adalah sebuah pintu yang menahan semua proses berdasarkan URL, dan didalamnya sebenarnya ada sebuah proses if then yang simple. Kedepannya sebenarnya filter ini sangat bagus diintegrate dengan rule engine, sehingga menjadi sebuah filter yang sangat smart.

Adapun standar pengembangan Filter adalah

```
package org.blueoxygen.cimande.security;

import javax.servlet.*;
import javax.servlet.http.HttpServletRequest;
import java.io.IOException;

public class SecurityFilter implements Filter {
    public void doFilter(
        ServletRequest request,
        ServletResponse response,
```

```
FilterChain chain)
throws IOException, ServletException {
HttpServletRequest req =
(HttpServletRequest) request;

if (!(req instanceof
        SecurityHttpRequestWrapper)) {
req = new SecurityHttpRequestWrapper(req);
}

chain.doFilter(req, response);
}

public void init(FilterConfig filterConfig)
throws ServletException {
}

public void destroy() {
}
}
```

Baris chain.doFilter adalah implementasi dari FilterChain yang mana adalah sebuah perintah untuk menteruskan pemrosesan. Tanpa perintah ini, artinya tidak lolos pemfilteran.

Bilamana Filter telah berhasil dibuat, Filter tersebut harus diregister di web.xml, adapun mekanisme penulisannya adalah

```
<filter>
<filter-name>security</filter-name>
<filter-class>org.blueoxygen.cimande.security.
LoginFilter</filter-class>
</filter>
```

Sedangkan bilamana telah *register*, diperlukan melakukan mapping URL, yaitu dengan format sebagai berikut

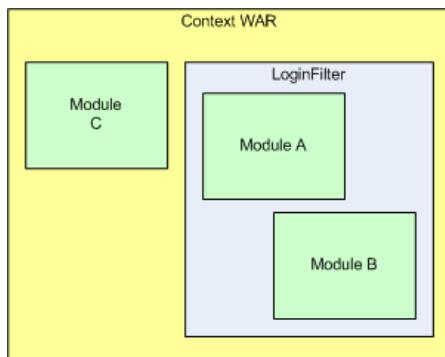
```
<filter-mapping>
    <filter-name>security</filter-name>
    <url-pattern>/module/*</url-pattern>
```

```
</filter-mapping>
```

Arti dari perintah diatas adalah melakukan registrasi Filter bernama LoginFilter dengan nama package org.blueoxygen.cimande.security. Sedangkan filter ini aktif untuk URI yang memiliki kata /module, contohnya <http://www.blueoxygen.org/module/index.jsp> atau <http://www.blueoxygen.org/module/module/index.jsp>, pokoknya semua akses terhadap /module/*.

Kelebihan mengimplementasikan Filter ini adalah kita dapat mengblok sebuah *folder* baik itu secara fisik yaitu *folder* didalam Context WAR, atau secara *virtual*.

Sebenarnya LoginFilter merupakan implementasi Filter yang melakukan pengecheckan *Session*. Ini adalah *filter* keamanan akses terhadap sistem yang lebih *modern* daripada mekanisme pengecheckan satu-satu terhadap nilai *session* seperti yang telah dibahas diatas. Implementasi Filter seperti ini yaitu dengan memisahkan antara aplikasi yang dikembangkan dengan keamanan sistem. Membuat aplikasi lebih sulit dihack, kecuali memang ada yang dapat melakukan *hacking* terhadap *memory* didalam Filter tersebut, yang umumnya Filter diurus oleh Java EE Container.



Integrasi Session dengan Aplikasi

Dengan mekanisme LoginFilter ini, kita dapat memisahkan *module-module* pengembangan dari satu area misalnya /guest/* ke /module/*, ini artinya secara otomatis *module* tersebut berarti tidak dapat diakses tanpa login.

Filter sebenarnya diciptakan bukan untuk keamanan saja, tetapi juga untuk melakukan banyak hal, contoh sebuah Action WebWork yang dapat melempar informasi ke *layer viewer* atau presentation layer, adalah sebuah contoh implementasi Filter. Mekanisme ini disebut DispatcherFilter.

Filter sebenarnya bekerja mirip seperti Interceptor pada WebWork juga, hanya Filter bekerja didalam Java EE secara *default*, sedangkan Interceptor berjalan didalam WebWork.

Kelebihan implementasi Filter dengan Session *handling* didalamnya adalah memungkinkan kita tidak perlu pusing lagi mengenai pengecheckan session didalam *memory*. Mekanisme ini tentu saja akan mempercepat pemograman, karena *programmer* hanya perlu berpikir terhadap aplikasi yang dibentuk.

Sedangkan bilamana diperlukan informasi user yang aktif, gunakan saja pemanggilan session, sehingga informasi dapat dijalankan.

Membuat Aplikasi dengan Filter dan Session

Penulis telah menggunakan *filter* untuk projek cimande, dan ini adalah cuplikan web.xml-nya

```
<filter>
<filter-name>login</filter-name>
<filter-class>org.blueoxygen.cimande.security.
LoginFilter</filter-class>

<init-param>
```

```
<param-name>loginPage</param-name>
<param-value>/backend/user/login.jsp</param-
value>
</init-param>
<init-param>
<param-name>siteSelection</param-name>
<param-value>/backend/user/site.jsp</param-
value>
</init-param>

<init-param>
<param-name>loginAction</param-name>
<param-value>/security/login.action</param-
value>
</init-param>
<init-param>
<param-name>ignoreExtensions</param-name>
<param-value>jpeg, gif, css</param-value>
</init-param>
</filter>
```

Perintah diatas adalah meregister LoginFilter, tetapi ada init-param yang merupakan sebuah variable yang disuntikan kedalam LoginFilter. Dimana untuk *register filter* telah dijelaskan sebelumnya. Init-param yang dimasukan terdapat loginPage dan siteSelection, variable ini secara otomatis dikenal dalam LoginFilter, tentu saja saat dia diinisialisasi. Implementasinya adalah sebagai berikut.

Sebelum memasuki implementasi Filter yang lebih dalam, harap diketahui bahwa Filter datang dengan 3 implementasi method, yaitu init(), destroy() dan doFilter().

Coba lihat kode init LoginFilter dibawah ini, yang merupakan init dari Filter, yang akan dieksekusi setiap kali Filter ini diakses, untuk kasus disini adalah akses url /module/*.

```
public void init(FilterConfig filterConfig) throws
ServletException {
```

```
loginPage =
filterConfig.getInitParameter("loginPage");
loginAction =
filterConfig.getInitParameter("loginAction");

extensions = new HashSet();

String ignoreExtensions =
filterConfig.getInitParameter(
"ignoreExtensions");
StringTokenizer st =
new StringTokenizer(ignoreExtensions, ", ");

while (st.hasMoreTokens()) {
    extensions.add(
        st.nextToken().toLowerCase());
}
}
```

Init diatas menjelaskan bahwa nilai dari loginPage dan loginAction diambil dari filterConfig, ini artinya nilai didapat dari inisialisasi pada web.xml.

Coba lihat kode LoginFilter dibawah ini:

```
public void doFilter(
    ServletRequest request,
    ServletResponse response,
    FilterChain chain)
    throws IOException,
        ServletException {
    HttpServletRequest req = (
        HttpServletRequest) request;
    HttpServletResponse res = (
        HttpServletResponse) response;

    String servletPath = req.getServletPath();

    String extension =
        servletPath.substring(
    servletPath.lastIndexOf('.') + 1).toLowerCase();
```

```
String uri = req.getRequestURI();
String contextPath = req.getContextPath();

if (servletPath.equals(loginAction) ||
    servletPath.equals(loginPage) ||
    extensions.contains(extension)) {
    chain.doFilter(req, res);

    System.out.println("loginAction");
} else if (
    req.getSession(true).getAttribute(
    LOGIN_CIMANDE_USER) == null) {
    System.out.println("redirecting");
    res.sendRedirect(contextPath +
    loginPage + "?redirectUri=" + uri);

    return;
} else {
    chain.doFilter(req, res);
}
```

Script diatas adalah sebuah doFilter yang melakukan pengcheckan terhadap session GA_USER, dan bilamana session tidak ada, langsung return, artinya proses ditolak, sedangkan bilamana ditemukan dilakukan chain.doFilter, artinya proses telah lolos dan siap masuk ke proses pemfilteran berikutnya.

Implementasi Filter diatas menjelaskan sebuah mekanisme untuk melakukan pengambilan path dari servlet, terus melakukan pengecheckan session yaitu LOGIN_CIMANDE_USER, kemudian dilanjutkan dengan chain.doFilter bila session tidak null, sedangkan bilamana null akan masuk ke loginPage.

Implementasi diatas sebenarnya berarti bilamana user belum login, maka session adalah null, dan akan redirect ke loginPage untuk login, tetapi ada redirectUri, yang mana ini adalah rekaman url sebelumnya. Sehingga bilamana telah login, page akan

kembali ke url sebelumnya.

Implementasi ini sangat baik bilamana kita memerlukan sebuah pengisian *survey*, misalnya terhadap pegawai, dan status *browser* di pegawai yang menerima belum login.

Kesimpulan

Kalau dilihat dari struktur standar pembuatan Filter, ini terlihat seberapa mudah mengembangkan aplikasi berbasis *security* pada pengembangan Web.

Cimande Thin adalah versi MVC yang tidak ber*security*, sedangkan Cimande Workspace for Web, merupakan contoh aplikasi yang mengimplementasikan Filter dan Session. Cobalah bermain-main dengan mengakses module yang ada didalamnya untuk mengerti bagaimana filter ini bekerja terhadap keseluruhan aplikasi didalam cimande.

Dapat dilihat setelah kita melakukan implementasi keamanan sistem dengan gabungan *session* dan *filter* untuk *user* yang login, maka akan terlihat seberapa cepat pengembangan aplikasi setelah ada filter ini.

Pengalaman penulis untuk mengembangkan sebuah aplikasi one-to-many dengan *mekanisme security* yang tidak diparsing adalah jauh-jauh lebih lambat dibandingkan menggunakan *security*. Pengalaman penulis mengatakan untuk mekanisme tradisional memerlukan lebih dari 2 minggu, sedangkan dengan metode *security* ini adalah hanya perlu max 2 minggu bagi mereka yang benar-benar berlum tahu Java, dan umumnya 3-5 hari untuk yang sudah mengenal *framework* cimande. Efisien bukan.

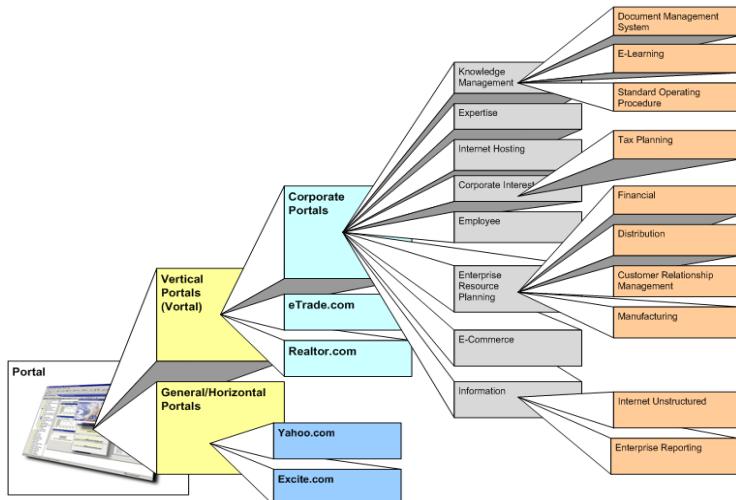
BAB VIII

Manajemen Organisasi dengan BlueOxygen Cimande

Sekilas mengenai Projek Cimande Workspace

Cimande merupakan sebuah projek yang mengacu pada *framework* kombinasi MVC yaitu Velocity/JSP, Hibernate dan WebWork.

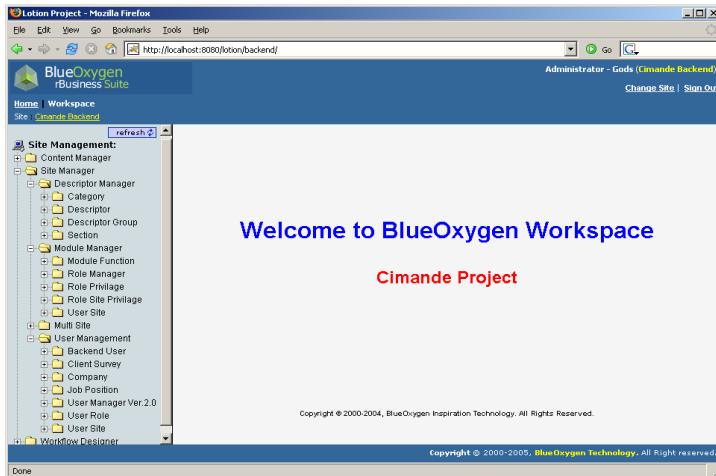
Adapun sebenarnya didalam Cimande memiliki lebih dari sekedar MVC plus *user management*, tetapi merupakan sebuah projek yang memungkinkan dikembangkan alias *platform pengembangan aplikasi*, atau lebih keren lagi disebut Web 2.0 Platform.



Portal Scope dari Hellen (Corporate Portal)

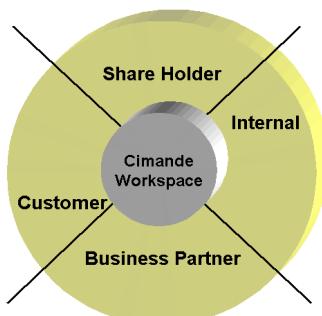
Diagram Pengembangan Portal dengan Cimande mengacu pada konsep Hellen

Secara konseptual, Cimande merupakan implementasi dari teori yang dituliskan Helen pada bukunya Corporate Portal.



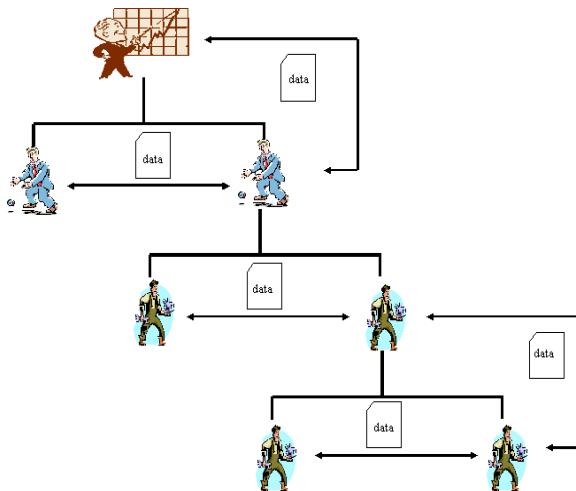
BlueOxygen Cimande Workspace edisi Web

Sedangkan *tree* yang berada disebelah kiri, disebut Workspace Explorer merupakan implementasi dari *360 degree organization mapping*. Diharapkan kedepannya Cimande dapat dikembangkan untuk menjadi media komunikasi antar *stakeholder*.



Cimande adalah Stakeholder Connector Platform

Konsep 360 adalah sebuah implementasi dari manajemen personalia yang memungkinkan terjadi *peering review* antar departemen, baik itu atas dengan bawahan, bawahan dengan atasan, sesama rekan.

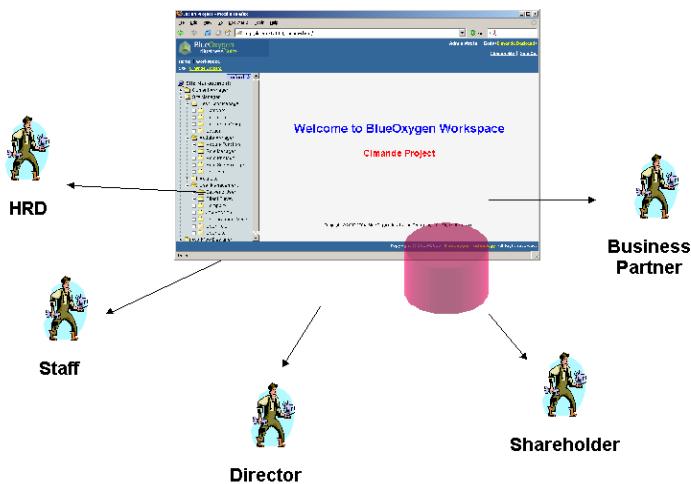


360 degree organization structure

Penulis berkeinginan versi kedepannya mendukung *Management By Objective* (MBO) dan organisasi dengan model

Matrix.

Selain itu model *user management* didalam Cimande mendukung multiple *hierarchical role*, yang artinya memungkinkan implementasi role untuk perusahaan-perusahaan atau organisasi yang memiliki posisi pegawai yang menduduki posisi rangkap.



Single Workspace untuk multiple role

User Management pada Cimande

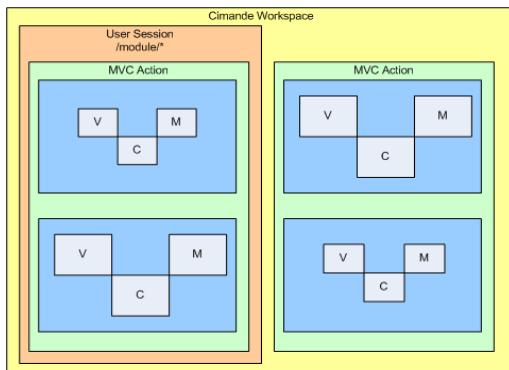
Bab ini adalah final dari mekanisme pembuatan sebuah workspace dengan arsitektur MVC, menggunakan WebWork, Velocity dan Hibernate. Sebenarnya projek pengembangan Cimande tidak semuanya MVC, hal ini dikarenakan pada awal-awal projek cimande dikembangkan tepatnya 2001, saat itu belum berhasil menemukan mekanisme integrasi MVC seperti sekarang, dan pada saat itu Spring yang merupakan teknologi injection juga belum lahir. Tetapi bentuk arsitektur awal yaitu *model driven development* masih tetap dikembangkan, dimana pada buku ini dibuat, komponennya masih dalam

pengembangan.

Bab ini akan membahas bagaimana MVC diwrap kedalam Cimande, sehingga module module CRUD yang telah dikembangkan didalam bab 6, hanya dapat diakses oleh user yang aktif. Hal ini juga dikarenakan didalam cimande telah diimplementasikan *filter* dan *session* untuk user managementnya.

Rahasianya sebenarnya terletak pada folder/module, yang mana *folder* ini adalah yang selalu dimonitoring oleh LoginFilter.

Bilamana kita melihat kode pada Cimande 1.0, akan didapat package org.blueoxygen.security. *Package* ini yang mengurusi semua *filter management* terhadap module tersebut, classnya adalah LoginFilter.

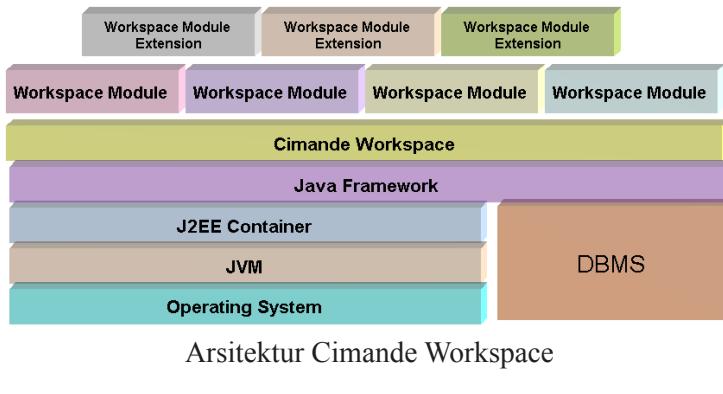


Module MVC dan Security Handling dalam Cimande

Security LoginFilter ini yang terintegrasi dengan module login, yang mana dalam versi kedepannya akan mendukung Single SignOn (SSO), yang juga sedang dalam penggerjaan.

Module Management Cimande

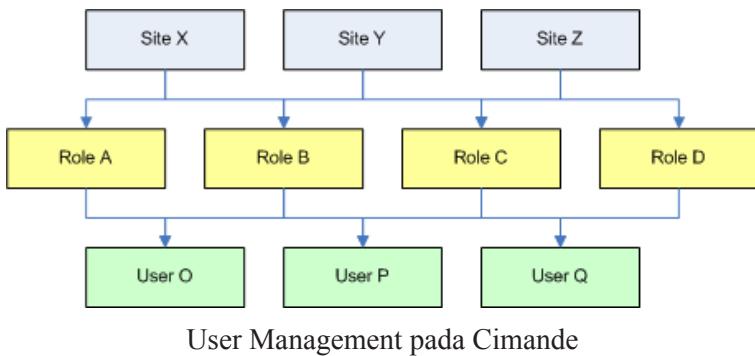
Sebenarnya ada fitur lainnya yang memungkinkan Cimande digunakan menjadi *module manager* dari setiap aplikasi MVC, karena memiliki fitur untuk mengatur module-module berdasarkan level yang dapat diatur.



Role adalah sebuah jabatan yang merupakan sebuah representatif dari posisi didalam perusahaan. Setiap role dapat terdiri dari lebih dari satu user, dan setiap role --saat ini-- memiliki fitur *parental*, artinya setiap role memungkinkan selain memiliki banyak user, juga memiliki *child* banyak *role*. Selain itu setiap role memungkinkan memiliki satu *parent*.

Dimana dalam struktur perusahaan ini berarti ada atasan dan ada bawahan.

Sebenarnya *module management* pada Cimande mengacu pada konsep Role-Site-User, yang artinya setiap modul dapat diakses bilamana *user* memiliki hak akses terhadap kombinasi Role-Site-User dengan modulnya.



User Management pada Cimande

Yang mana kombinasi Role-Site-User ini akan direpresentasikan melalui sebuah *tree* yang disebut Workspace Explorer

Workspace Explorer berbentuk pohon yang mana setiap node terakhirnya adalah sebuah *descriptor*. Descriptor ini adalah nama dari module MVC yang mekanisme pengembangannya telah dibahas dibab sebelum ini. Descriptor selalu memiliki 2 node New dan Search, yang merupakan perwakilan dari CRUD, sedangkan Module Function selalu berbentuk parent node yang mana childnya dapat berbentuk Module Function yang lain atau kumpulan Descriptor.

Module Function adalah struktur *representative* satuan terkecil dari tree pada Workspace Explorer. Setiap Workspace Explorer dapat diberikan lebih dari module function, dan module function yang dimasukan adalah yang memiliki parent=0, alias module function tertinggi.

Setiap node yang tidak memiliki *child* lagi, maka dalam Workspace Explorer akan muncul informasi mengacu pada *descriptor*.

Mekanisme Workspace Explorer

Untuk membuat sebuah *tree* atau workspace explorer

diperlukan kombinasi antara *role* dan *site*. Dimana setiap *role* dapat terdiri dari lebih dari satu *user*. User ini yang login dan mengaksesnya.

Sedangkan untuk membuat pohnnya, kita dapat memasukan lebih dari satu *module function*. Dimana didalam *module function* terdapat lebih dari satu *descriptor*. *Descriptor* ini merupakan aplikasi CRUD yang kita kembangkan.

Bilamana ada role yang berbeda, dan diinginkan untuk mengakses module yang sama, ini sama artinya mengassign dua kali module yang sama pada kombinasi role site.

Sebenarnya ada satu fitur yaitu *role privilege*, yaitu kombinasi *module function* hanya dengan *role*, tidak dengan *site*. Ini diperlukan untuk mengembangkan aplikasi yang lebih simple dengan Cimande. Sedangkan *global module* masih dalam tahap pengembangan pada saat buku ini dibuat.

Bekerja dengan Descriptor

Descriptor adalah satuan terkecil dari entitas yang digunakan untuk melakukan *mapping* struktur organisasi. *Descriptor* dapat juga dikatakan sebagai sebuah aplikasi kecil yang siap diakses oleh pihak yang diberi hak untuk mengakses.

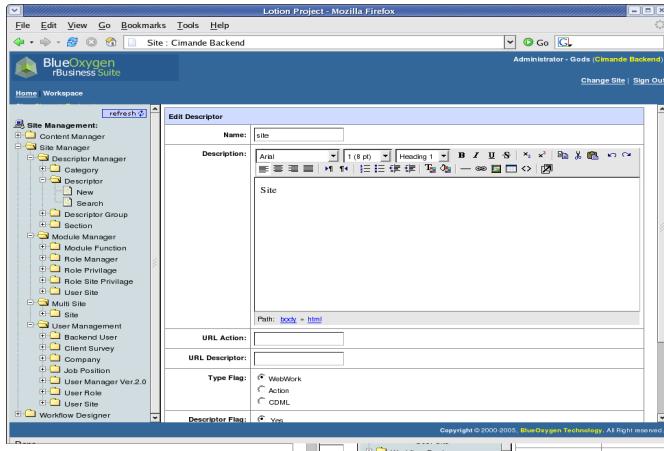
Nama dari *descriptor* harus huruf kecil, dan umumnya disamakan dengan nama *table* aplikasi tersebut dibuat.

Bilamana nama *table* adalah “*purchase_order*”, maka nama *descriptornya* adalah “*purchase_order*” juga. Untuk memudahkan dipahami dapat memasukan dalam *description* pada *descriptor* tersebut.

Membuat Workspace Explorer Sendiri

Bilamana kita telah berhasil membuat sebuah aplikasi CRUD,

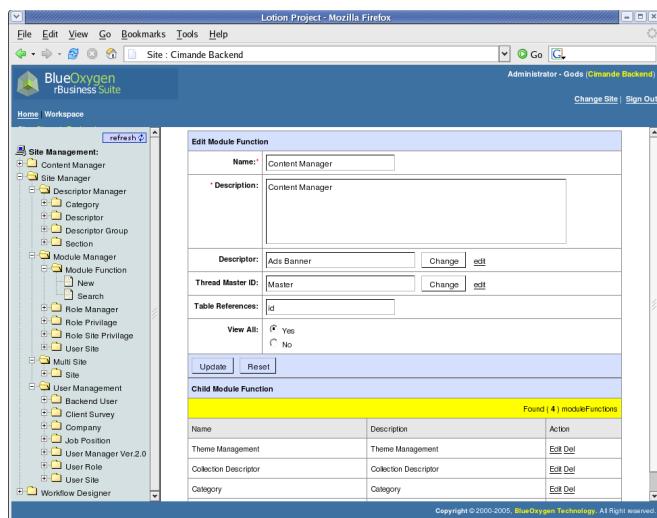
terutama memiliki `create.action` dan `filter.action`, maka aplikasi tersebut siap *register* didalam *descriptor*.



Form Isian Descriptor

Bilamana *descriptor* telah dibuat, proses berikutnya adalah membuat *module function*.

Module function memiliki 2 tipe, yaitu yang akan menjadi *tree*, dan yang menjadi *node*. Yang *node* ini artinya tidak memiliki *child*, untuk mengetahuinya adalah dengan mengakses *node Module Function* pada Workspace Explorer Administrator. Bilamana list dibawah *module function* tersebut adalah kosong, artinya ini *node*, bilamana ada artinya *tree*.



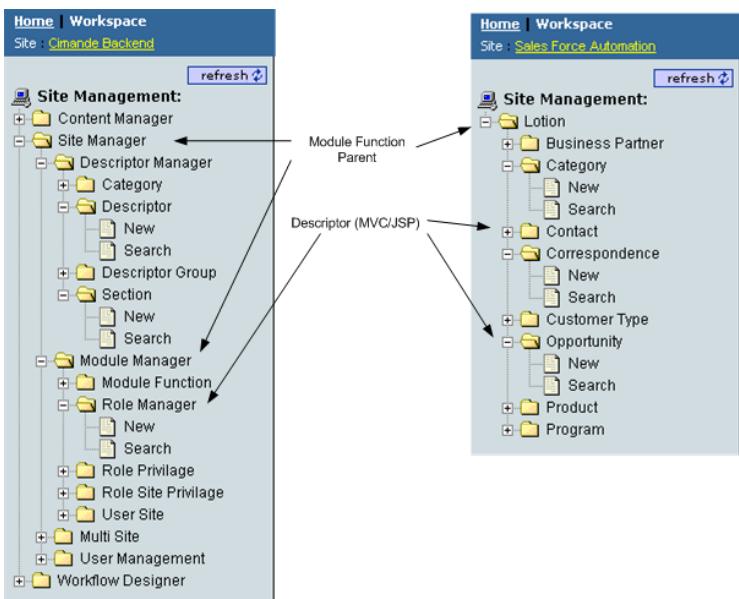
Module Function dengan Child

Bilamana pengisian telah selesai, proses berikutnya adalah :

- * Membuat Role
- * Membuat Site
- * Membuat Kombinasi Role-Site
- * Menggassign Module Function kedalam Role-Site
- * Mengasssign Module ke Role Privilage bilamana tanpa Site
- * Membuat User

Semua kegiatan ini dapat diakses pada Workspace Explorer dengan login admin/admin aka Cimande Backend.

Berikut adalah bentuk Workspace Explorer untuk Administrator, dan hasil pembuatan Workspace Explorer untuk Role-Site Lotion.



Module Function dan Descriptor untuk 2 Workspace Explorer
(Administrator dan Lotion)

Membuat Role

Membuat role baru dapat dilakukan dengan mengakses New pada Workflow Role atau User Manager. Form isiannya hanya meminta nama role dan descriptionnya.

Membuat Site

Site pada awalnya diciptakan untuk nama domain, sehingga memungkinkan terjadi sharing content dalam descriptor yang sama. Oleh karena ini, masa masukan pada site menjadi lebih banyak, tetapi sebenarnya yang diperlukan hanya name dan descriptionnya saja.

Membuat Role-Site

Role site adalah kombinasi, yang artinya bilamana user mengakses

workspace, dan rolenya adalah role A, dia akan diberikan pilihan mau masuk ke site mana. Bilamana site dipilih, maka muncul Workspace Explorer yang khusus untuk kombinasi role dan site tersebut.

Untuk membuat role-site, dapat memilih Role Manager pada Workspace Explorernya Administrator.

Bilamana node New dipilih, maka akan muncul semua role yang ada didalam Workspace.

The screenshot shows a Mozilla Firefox browser window with the title "Lotion Project - Mozilla Firefox". The address bar says "Site : Cimande Backend". The page content is from the "Administrator - Gods (Cimande Backend)" section of the BlueOxygen rBusiness Suite. On the left, there's a sidebar titled "Site Management" with various categories like Content Manager, Site Manager, Descriptor Manager, Module Manager, User Management, etc. Under "Role Manager", there are "New" and "Search" options. The main area is titled "Workflow Role List" and contains a table:

Name	Description	Role Site
Approver	Approve the document	Delete Add
Content Editor	Content Editor	Delete Add
Editor	Edit the document	Delete Add
Element Editor	Element Editor	Delete Add
Gods	Gods	Delete Add
Journalist	Create the document	Delete Add
Lotion	Lotion	Delete Add
Publisher	Publish the document	Delete Add

At the bottom right of the table, it says "Copyright © 2003-2005, BlueOxygen Technology. All Right reserved."

Role Site

Tekanlah Add untuk mengassign site kedalam role tersebut, bilamana tombol delete ditekan, artinya menghapus semua kombinasi.

The screenshot shows the BlueOxygen Business Suite interface. On the left, there's a navigation sidebar titled 'Site Management' with several categories: Content Manager, Site Manager, Descriptor Manager, Module Manager, Multi Site, User Management, and others. The main content area is titled 'Role Site'. It contains a table with one row where 'Role ID' is 3, 'Name' is 'Journalist', and 'Description' is 'Create the document'. Below this table is a list of sites assigned to this role: null, Makin Portal, Makin Intranet, Document Archive, Activity Management, and null again. At the bottom of the main panel, there's a 'Transaction' section with fields for 'Site ID' and 'Site Name', both currently empty.

Assign Site kedalam Role

Setiap role dapat diassign lebih dari satu site. Kombinasi ini nanti harus ditambahkan module function baru, sehingga bila user mengakses, workspace explorernya memiliki isi.

Untuk menambahkan module function kedalam kombinasi role site ini, klik New pada Role Site Privilage.

Ada 3 step untuk meassign module funciton ke role-site, yaitu:

1. Memilih Role

Akan muncul semua role yang memiliki site.

The screenshot shows a Mozilla Firefox browser window titled "Lotion Project - Mozilla Firefox". The address bar says "Site : Cimande Backend". The page title is "Administrator - Gods (Cimande Backend)". On the left, there's a sidebar with a tree view under "Site Management" containing categories like Content Manager, Site Manager, Descriptor Manager, Module Manager, etc. The main content area is titled "Role Site List" and contains a table:

Name	Description	Status
Journalist	Create the document	View Role Site
Gods	Gods	View Role Site
Lotion	Lotion	View Role Site

At the bottom right of the page, it says "Copyright © 2000-2005, BlueOxygen Technology. All Right Reserved."

List Role pada Role Site Privilage

2. Memilih Site

Akan muncul semua site mengacu pada pemilihan role sebelumnya. Berikut adalah site untuk role “lotion”

The screenshot shows a Mozilla Firefox browser window titled "Lotion Project - Mozilla Firefox". The address bar says "Site : Cimande Backend". The page title is "Administrator - Gods (Cimande Backend)". On the left, there's a sidebar with a tree view under "Site Management" containing categories like Content Manager, Site Manager, Descriptor Manager, Module Manager, etc. The main content area is titled "Site List" and contains a table:

Name	Description	Status
Material Movement	Material Movement	Add Privilage

At the bottom right of the page, it says "Copyright © 2000-2005, BlueOxygen Technology. All Right Reserved."

List Site

3. Memasukan Module Function

Setelah site dipilih, akan muncul menu untuk memasukan module function. Module function ini yang akan muncul pada Workspace Explorer.

Module Function ID	Module Function Name	Action
ff8080810b835c2b010b837162c6	Material Movement	Edit Del

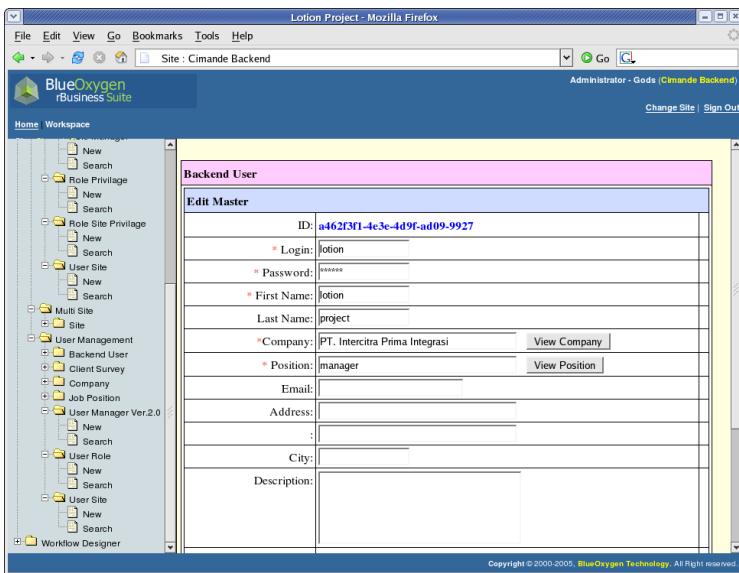
Module pada kombinasi role Lotion dan site Material

Membuat User, Mengassign Role dan Site

Proses berikutnya adalah membuat user. Setiap user harus diberi role, dan site.

Sebenarnya ada kebutuhan untuk membuat aplikasi diatas cimande, dan hanya memerlukan role saja, tanpa ada sitenya. Sebenarnya secara aplikasi, cimande sudah siap, tetapi user manager yang dikembangkan saat ini belum dapat mengacu pada kebutuhan ini.

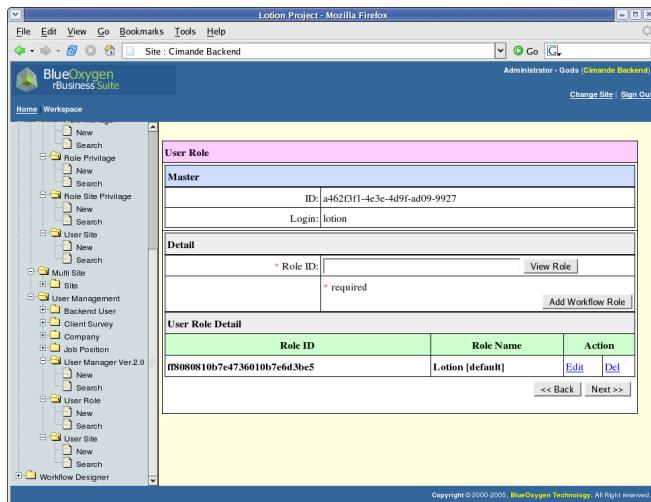
Bilamana dipelajari lebih dalam, Administrator adalah bentuk dari implemntasi role dan user saja, tanpa ada site.



Membuat User “Lotion”

Pembuatan user memerlukan 2 isian selain username, password, first name, dan last name, yaitu company dan position. Sebenarnya position ini sama dengan role, tetapi ada beberapa kasus yang didapatkan penulis, ada perusahaan yang memiliki role didalam komputer adalah manager, tetapi dia ternyata memiliki kekuasaan seperti direktur, maka dibuatlah tambahan ini.

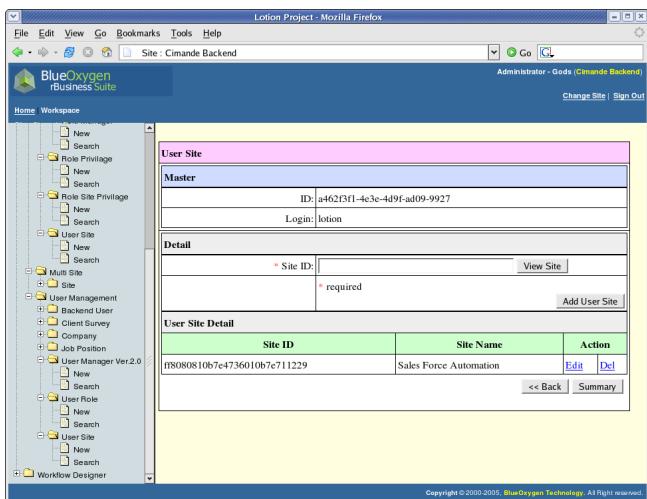
Bilamana isian telah selesai, tekanlah tombol Next, maka akan masuk ke modus pemilihan role.



Pemilihan role

Tekanlah tombol view role, maka popup yang menampilkan semua role yang ada, dan berstatus aktif, akan muncul, pilihlah. Dapat diassign lebih dari satu role untuk setiap user.

Bilamana telah selesai, tekanlah tombol Next, untuk masuk ke pemilihan Site.



Pemilihan Site

Dapat dipilih lebih dari satu site untuk setiap user. Tekan summary untuk melihat hasil akhirnya.

Harap diperhatikan, bilamana terassign lebih dari satu role dan lebih dari satu site, maka secara otomatis pada saat user login, maka user harus memilih role yang akan dipakainya, dan site yang akan diaksesnya.

Tentu saja, hasil pemilihan role dan site ini akan membuat Workspace Explorer yang berbeda tergantung dari privilagennya.

Cobalah login dengan user yang telah dibuat, dalam kasus ini adalah lotion dengan password lotion.

Sebenarnya step yang dilakukan untuk membuat user lotion, role lotion dan site lotion, adalah implementasi real dari projek CRM yang dikembangkan di BlueOxygen. Cobalah buat aplikasi lainnya sendiri.

BAB IX

BlueOxygen Postila, Solusi POS dengan Swing dan Hibernate

Bab ini akan mengerangkan cara lain mengimplementasikan Hibernate sebagai layer model untuk aplikasi berbasis Java Desktop. Java Desktop artinya aplikasi yang berjalan secara standalone.

Projek yang dikembangkan oleh tim penulis diberi nama Postila, yaitu projek untuk mengembangkan Point of Services, sampai tulisan ini dibuat hanya versi supermarket yang telah direlease, tetapi kedepannya akan muncul banyak versi dari Postila.

Postila lahir karena diperlukannya interaksi antara aplikasi yang dikembangkan dengan devices seperti RFID, SmartCard, Bar Code Reader, atau CashDrawer, yang semuanya merupakan standar dari ARTS , asosiasi ritel dunia. ARTS ini telah mengembangkan standar JavaPOS (www.javapos.com) yang merupakan standarisasi API yang memungkinkan mengakses semua devices tersebut menggunakan Java dengan mekanisme yang sama, walaupun vendor yang mengeluarkan devices adalah berbeda-beda.

Sekilas mengenai JavaPOS

ARTS singkatan Association for Retail Technology Standards adalah sebuah lembaga internasional yang berdedikasi untuk mencari sebuah mekanisme menekan biaya terhadap teknologi dengan mengeluarkan standar.

ARTS mengeluarkan empat standar yaitu:

- * The Standard Relational Data Model,

- * UnifiedPOS
- * IXRetail
- * the Standard RFPs.

Dari semua standar diatas, hanya Unified POS yang dibahas dibab ini.

UnifiedPOS adalah sebuah inisiatif retail-driven yang mengkombinasikan antara standar interface terhadap perangkat keras yang memungkinkan kebebasan dalam memilih perangkat keras untuk solusi Point of Servicesnya atau POSnya.

The UnifiedPOS specification will formalize and document the underlying retail device architecture, currently shared by both the JavaPOS and OPOS standards, in an operating system independent and language neutral manner.

Spesifikasi UnifiedPOS hadir dengan dokumentasi dan arsitektur yang dipakai oleh JavaPOS dan OPOS. Dimana OPOS adalah sebuah teknologi yang independen terhadap sistem operasi Windows dan merupakan implementasi dari COM, sedangkan JavaPOS adalah spesifikasi yang netral terhadap segala sistem operasi dan menggunakan Java sebagai teknologinya. Postila berdiri sebagai implementasi dari JavaPOS.

UnifiedPOS telah berkomitmen tidak akan mendukung teknologi diluar COM dan Java.

Beberapa faktor yang membuat UnifiedPOS diluncurkan adalah

- * Kesamaan Kebutuhan
- * Menggunakan Kembali Perangkat dengan Model sama
- * Menekan biaya pembelajaran perangkat keras baru.

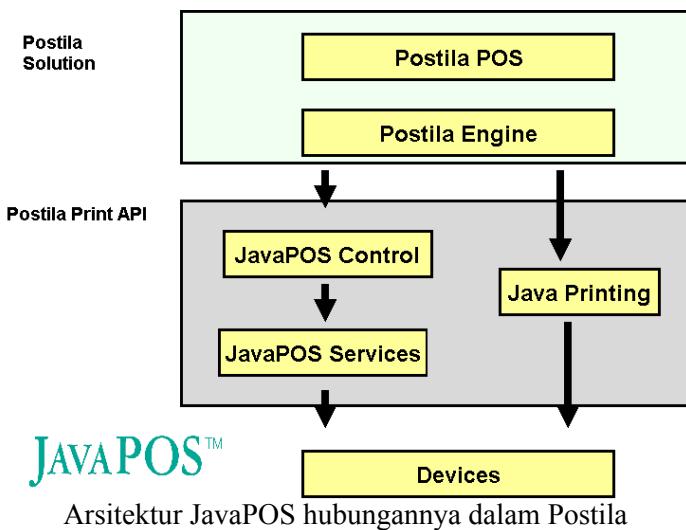
UnifiedPOS yang didukung secara internasional memungkinkan terjadi pertukaran informasi antara vendor, platform dan

format retail. Yang akhirnya mengakibatkan persaingan pada level implementasi bukan pada level perangkat yang telah distandarisasi.

Adapun perangkat keras yang telah distandarisasi adalah:

- * Bump Bar
- * Cash Changer
- * Cash Drawer
- * Credit Authorization Terminal
- * Coin Dispenser
- * Fiscal Printer
- * Hard Totals
- * Keylock
- * Line Display
- * Magnetic Ink Character Recognition Reader
- * Magnetic Stripe Reader
- * PIN Pad
- * Point Card
- * POS Keyboard
- * POS Printer
- * Remote Order Display
- * Scale
- * Scanner (Bar Code Reader)
- * Signature Capture
- * Tone Indicator
- * Forecourt (pending)
- * Smart Card (pending)

Adapun arsitektur dari JavaPOS adalah



UnifiedPOS Programmatic Names	OPOS Programmatic Ids	JavaPOS Class Names
BumpBar	OPOS.BumpBar	ipos.BumpBar
CashChanger	OPOS.CashChanger	ipos.CashChanger
CashDrawer	OPOS.CashDrawer	ipos.CashDrawer
CAT	OPOS.CAT	ipos.CAT
CheckScanner	OPOS.CheckScanner	ipos.CheckScanner
CoinDispenser	OPOS.CoinDispenser	ipos.CoinDispenser
FiscalPrinter	OPOS.FiscalPrinter	ipos.FiscalPrinter
HardTotals	OPOS.HardTotals	ipos.HardTotals
Keylock	OPOS.Keylock	ipos.Keylock
LineDisplay	OPOS.LineDisplay	ipos.LineDisplay
MICR	OPOS.MICR	ipos.MICR
MotionSensor	OPOS.MotionSensor	ipos.MotionSensor
MSR	OPOS.MSR	ipos.MSR
PINPad	OPOS.PINPad	ipos.PINPad
PointCardRW	OPOS.PointCardRW	ipos.PointCardRW
POSKeyboard	OPOS.POSKeyboard	ipos.POSKeyboard
POSPower	OPOS.POSPower	ipos.POSPower
POSPrinter	OPOS.POSPrinter	ipos.POSPrinter
RemoteOrderDisplay	OPOS.RemoteOrderDisplay	ipos.RemoteOrderDisplay
Scale	OPOS.Scale	ipos.Scale
Scanner	OPOS.Scanner	ipos.Scanner
SignatureCapture	OPOS.SignatureCapture	ipos.SignatureCapture
ToneIndicator	OPOS.ToneIndicator	ipos.ToneIndicator

Perbedaan antara OPOS dengan JavaPOS

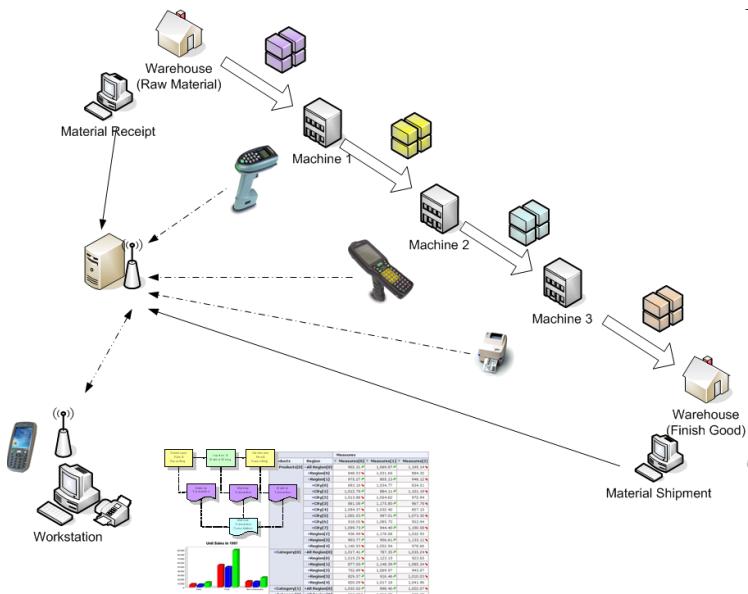
Sekilas Tentang BlueOxygen Postila

Projek Postila yang dibahas dibab ini sebenarnya diposisikan dengan dua model pendekatan yaitu secara teknologi dan secara fungsional.

Fitur secara fungsional untuk projek Postila adalah sebagai sebuah aplikasi yang bekerja meliputi:

- Barang Masuk
- Barang Keluar
- Transfer Barang antar Gudang
- Point of Sales

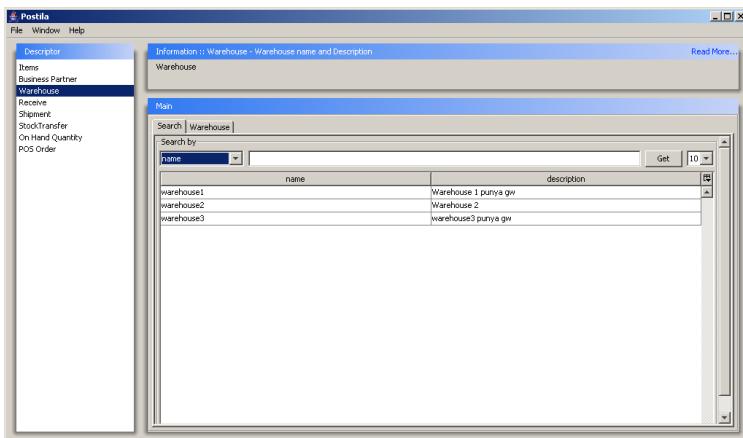
Sebenarnya ada beberapa fitur tambahan yang mungkin setelah buku ini diterbitkan baru selesai, yaitu fitur untuk mendukung POS untuk supermarket, restoran dengan meja, tempat makan siap saji, absensi.



Postila untuk Solusi Pergudangan

Sedangkan secara teknis, Postila sebenarnya sebuah implementasi dari teknologi yang akan dikeluarkan oleh Sun Microsystems yaitu extension dari SwingX, dengan teknologi persistensi Hibernate.

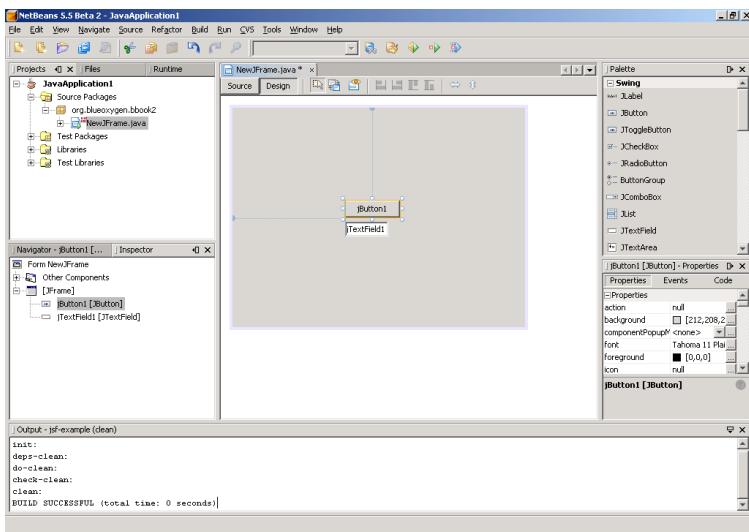
Sampai buku ini ditulis, sebenarnya Postila akan dikembangkan menggunakan sharing ORM dengan mengimplementasikan Spring sebagai IOCnya, tetapi sayang sekali sampai tulisan ini ditulis belum selesai dikerjakan, sehingga session factory dari Hibernate ditransfer dari setiap panel ke panel lain. Sedangkan untuk reporting dikembangkan menggunakan JasperReport.



Bentuk dari Aplikasi Postila

Postila dan Netbeans

Aplikasi Postila dikembangkan menggunakan Swing. Tidak seperti bab-bab sebelumnya yang sangat berbau Eclipse, Postila dikembangkan menggunakan Netbeans dari Sun Microsystems. Netbeans dipilih karena diawali dengan fitur Matisse, yang dapat bekerja dengan drag-and-drop, sehingga memungkinkan mengembangkan aplikasi berbasis Swing jauh lebih mudah dibandingkan dengan layout yang lain.



Netbeans dengan Teknologi Matisse

Selain itu juga dikarenakan kerja sama antara tim Netbeans dengan tim JBoss, terutama teknologi persistence baik itu TopLink maupun Hibernate.

Sebenarnya ada teknologi SWT/JFace dari Eclipse yang bekerja sangat cepat terutama di Windows, karena lebih native dari Swing, tetapi karena pasar dari Swing yang telah mencapai diatas 45% menurut Java Developer Journal, dan lebih mature, membuat penulis memilihnya. Selain itu juga dikarenakan adanya inisiatif untuk mengintegrasikan Postila kedalam Compiere, yang merupakan ERP Open Source paling handal didunia.

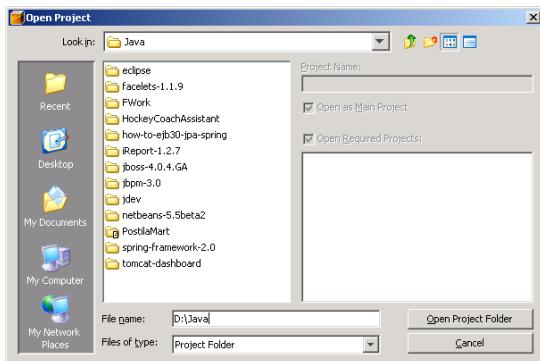
Sebenarnya pada saat Postila pertama dikembangkan, menggunakan Netbeans 5.0, tetapi beruntung telah keluar Netbeans 5.5b2, yang digunakan oleh tim penulis, yang bekerja sangat baik.

Adapun projek Postila adalah bersifat Open Source, yang dihost

di sf.net dengan alamat <http://www.sf.net/projects/postila>.

Bekerja dengan Postila

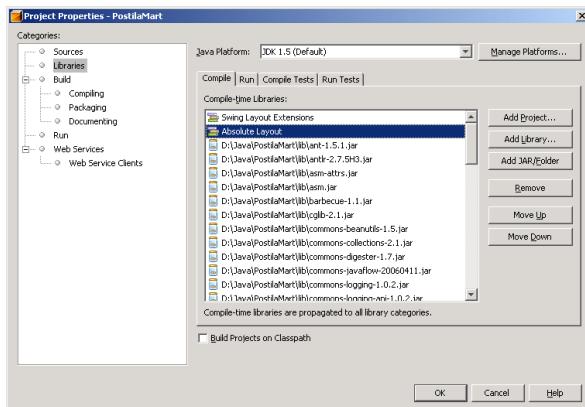
Bilamana telah mendapatkan source code dari Postila, dapat langsung dipanggil dengan Netbeans.



Dialog dengan PostilaMart yang memiliki icon projek Netbeans

Projek Postila memiliki folder yang ditambahkan icon kecil didalamnya, ini mengatakan bahwa projek PostilaMart ini adalah projek Netbeans. Bukalah projek ini, dan secara langsung didalam Projects akan keluar struktur dari Postila.

Seringkali saat membukanya muncul error, ini artinya folder tempat PostilaMart berada tidak diextract dengan benar, untuk itu diperlukan melakukan mounting jar kedalam project, caranya dengan memilih properties dan memilih Libraries, kemudian gantilah broken jar terhadap semua jar yang diperlukan Postila. Seluruh Jar diletakan difolder lib didalam folder PostilaMart.

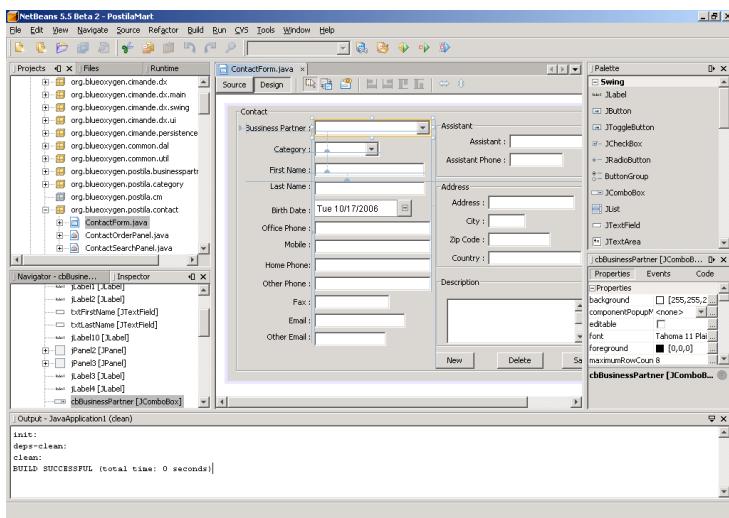


Jar yang tidak broken yang disiap digunakan

Bilamana semua proses telah selesai, jangan lupa untuk memload file postila.sql kedalam mysql, dan seperti biasa login dan password mysql adalah root dan tulalit, bilamana hendak menggantinya, gantilah file cimande.properties dan hibernate.cfg.xml.

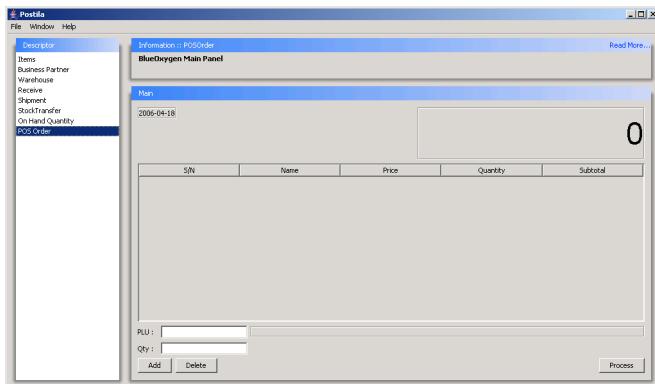
Cobalah buka file org.blueoxygen.postila.cm, maka akan muncul sebuah Frame yang menggunakan layout Matisse.

Cara Cepat Mengembangkan Aplikasi Java dengan Metode MVC



Postila didalam Netbeans dengan ContactFrame yang berlayout Matisse

Untuk menjalankannya tinggal menclik icon run project, maka secara otomatis Postila akan berjalan. Adapun login manajemen dari Postila adalah sama dengan projek Cimande.



Postila dengan panel POS (Point of Sales)

SessionFactory dalam Postila

Dikarenakan belum terintegrasinya Spring didalam Postila, sehingga setiap session factory dari Hibernate dicopykan kesetiap panel dan memasukannya kedalam constructor. Berikut ini adalah contoh session pada constructor ContactFrame.

```
public ContactForm(Session sess) {  
    this.sess = sess;  
    cat = sess.createQuery("FROM  
        +Category.class.getName()).list();  
    bp = sess.createQuery("FROM  
        +BusinessPartner.class.getName()).  
    list();  
    initComponents();  
  
    for(BusinessPartner business : bp){  
        cbBusinessPartner.addItem(  
            business.getName());  
    }  
    for(Category categories : cat){  
        cbCategory.addItem(  
            categories.getName());  
    }  
}
```

Adapun untuk melakukan loading session factory dari Hibernate, sebenarnya diload pada module org.blueoxygen.cimande.dx.main.Postila. Berikut adalah isi dari initHibernate() pada class Postila.

```
public void initHibernate() {  
    properties = new Properties();  
    try {  
  
        properties.load(  
        DbBean.getResourceAsStream(  
            "postila.properties"));  
    } catch (IOException ex) {  
        ex.printStackTrace();  
    }
```

```
AnnotationConfiguration config =
    new AnnotationConfiguration();
config.configure("hibernate.cfg.xml");
sf = config.buildSessionFactory();
}
```

Session factory pada initHibernate yaitu sess merupakan sebuah OpenSession dari sf, yang diaktifkan pada saat Postila dipanggil melalui constructornya

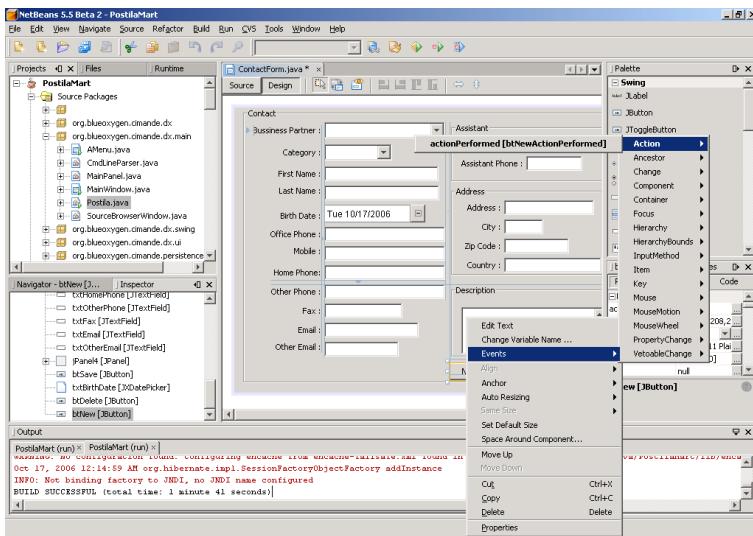
```
public Postila() {
    super();
    initHibernate();
    sess = sf.openSession();
}
```

Session ini yang dilempar dari satu class ke class lain.

Jadi sebenarnya implementasi Hibernate yang dibahas bab 3 tentang Model, adalah sama dengan mekanisme pada Web. Perbedaannya adalah pada Web setiap session diaktifkan perpage, sedangkan pada aplikasi desktop, harus dijaga terus menyala terutama pada aplikasi masih bernyala, session yang terputus dipanel manapun, akan membuat aplikasi terputus dengan database, dan memerlukan inialisasi kembali untuk menjalankannya.

Action pada Swing

Perbedaan pemogramaan Swing dengan pemograman Web adalah pada handling action, bilamana pada pemograman berbasis web, semua dilakukan melalui implementasi form baik dengan metode POST atau GET, sedangkan pada Swing, kita dapat memberikan Event pada setiap objek, dibawah ini adalah contoh objek button yang diberikan Action bernama actionPerformed(btNewActionPerformed), yang artinya pemanggilan metod bilamana tombol New pada ContactFrame ditekan.



Memberikan Action pada Swing di Netbeans

Bilamana kita gali lebih lanjut, pemilihan actionPerformed ini akan menggenerate sebuah fungsi yang memanggil btNewActionPerformed, yang mana ini merupakan model dari Netbeans, berikut adalah kode yang digenerate oleh Netbeans (initComponent), yang merupakan implementasi dari penekanan tombol New pada ContactFrame.

```
btNew.setText("New");
btNew.addActionListener(
    new java.awt.event.ActionListener() {
public void actionPerformed(
    java.awt.event.ActionEvent evt) {
    btNewActionPerformed(evt);
}
});
```

Berikut adalah isi dari btnNewActionPerformed

```
private void btNewActionPerformed(java.awt.event.
```

```
ActionEvent evt) {  
    contact = new Contact();  
    loadContactToForm();  
}
```

Bilamana diperhatikan ada ActionEvent yang dipindahkan dari actionPerformed ke btNewActionPerformed dengan variable bernama evt.

Berikut adalah implementasi dari btSaveActionPerformed, yang merupakan sebuah kegiatan merekan semua isi dari ContactFrame, tentu saja bilamana Contact dipanggil.

```
private void btSaveActionPerformed(  
    java.awt.event.ActionEvent evt) {  
  
    LogInformation log = new LogInformation();  
    log.setCreateDate(  
        new Timestamp(  
            System.currentTimeMillis()));  
    log.setLastUpdateDate(  
        new Timestamp(  
            System.currentTimeMillis()));  
    log.setActiveFlag(1);  
  
    BusinessPartner businessPartner =  
        new BusinessPartner();  
  
    if(!bp.isEmpty()) {  
        businessPartner =  
            bp.get(cbBusinessPartner.getSelectedIndex());  
    } else {  
        businessPartner = null;  
    }  
  
    Category category = new Category();  
    if(!cat.isEmpty()) {  
        category = cat.get(  
            cbCategory.getSelectedIndex());  
    } else {  
        category = null;  
    }
```

```
}

contact.setFirstName(txtFirstName.getText());
contact.setLastName(txtLastName.getText());
contact.setOfficePhone(
    txtOfficePhone.getText());
contact.setMobile(txtMobile.getText());
contact.setHomePhone(txtHomePhone.getText());
contact.setOtherPhone(
    txtOtherPhone.getText());
contact.setFax(txtFax.getText());
contact.setEmail(txtEmail.getText());
contact.setOtherEmail(
    txtOtherEmail.getText());
contact.setBirthDate(
    txtBirthDate.getActionCommand());
contact.setAssistant(txtAssistant.getText());
contact.setAssistantPhone(
    txtAssistantPhone.getText());
contact.setAddress(txtAddress.getText());
contact.setZipCode(txtZipCode.getText());
contact.setCity(txtCity.getText());
contact.setCountry(txtCountry.getText());
contact.setBirthDate(
    txtBirthDate.getDate().toString());
contact.setDescription(
    txtDescription.getText());
contact.setBusinessPartner(businessPartner);
contact.setCategory(category);
contact.setLogInformation(log);
Transaction tx = sess.beginTransaction();
    sess.saveOrUpdate(contact);
    tx.commit();
firePropertyChange(
AdvancedSearchPanel.SELECTED_TAB_CHANGE,
    1, 0);
}
```

Implementasi diatas adalah sebuah kegiatan penyimpanan data menggunakan Hibernate dengan implementasi JTA. JTA adalah implementasi Java untuk transaksi.

Berikut adalah baris yang menjelaskan mengenai implementasi transaksi pada saveActionPerformed

```
Transaction tx = sess.beginTransaction();
sess.saveOrUpdate(contact);
tx.commit();
```

Implementasi yang lebih lanjut tentu saja adalah implementasi rollback bilamana terjadi masalah pada penyimpanan data.

Sedangkan LogInformation adalah class yang juga digunakan pada projek web dengan Cimande, yang mana LogInformation adalah membaca session login yang aktif dan merekamnya kedalam database. LogInformation diperlukan untuk merekam informasi user terhadap data yang direkam.

Bilamana kegiatan adalah Update, maka field updateby akan diisi dengan username dari user yang login.

Sebenarnya ada implementasi yang lebih lanjut, yaitu pada metode Search, yang mana untuk contact person classnya adalah ContactSearchPanel, berikut adalah kodennya:

```
public ContactSearchPanel(DefaultTableModel model,
    Class clazz, Session sess) {
    super(model, clazz, sess);
}

public void loadListToTableModel(
    List list, DefaultTableModel model) {
    while(model.getRowCount() > 0) {
        model.removeRow(model.getRowCount() - 1);
    }
    if(!list.isEmpty()){
        String sContact = "";
        for(Object ob : list){
            Contact ct= (Contact) ob;
```

```
model.addRow(new Object[]{ct.getFirstName(),
    ct.getLastName(),
    ct.getOfficePhone(),
    ct.getMobile(),
    ct.getHomePhone(),
    ct.getOtherPhone(),
    ct.getFax(),
    ct.getEmail(),
    ct.getOtherEmail(),
    ct.getBirthDate(),
    ct.getAssistant(),
    ct.getAssistantPhone(),
    ct.getAddress(),
    ct.getCity(),
    ct.getZipCode(),
    ct.getCountry(),
    ct.getDescription(),
    ct.getBusinessPartner().getName(),
    ct.getCategory().getName()});
}
}
}
```

Dapat dilihat pada objek model.addRow, disinilah letak dari implementasi MVC yang sudah disediakan oleh Swing. JTable sangat kental dengan MVC.

BAB X

Mengembangkan Aplikasi berbasis SOA dan Memperkenalkan PASIR

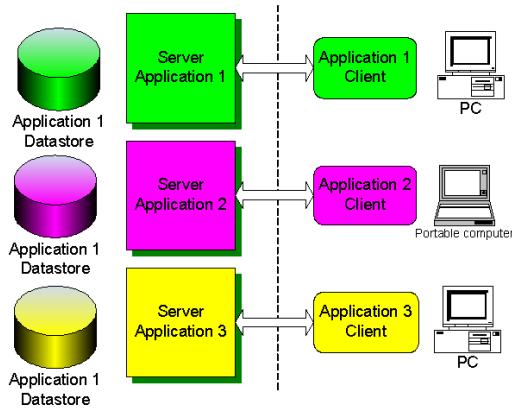
Bab-bab sebelumnya menjelaskan bagaimana mengembangkan sebuah platform berbasis Web, dan bilamana ditambahkan fitur AJAX, berubah menjadi Web 2.0 Platform.

Tetapi tentu saja kita tidak dapat memaksakan kombinasi komponen penyusun framework, seperti yang terulang dibab 1, ada beberapa kombinasi berbasis MVC yang dikembangkan seperti JBoss dengan Seamnya, Oracle juga memiliki dengan ADFnya, lalu Alfresco juga memiliki, serta buku ini membahas kombinasi lain.

Semua kombinasi ini memungkinkan dikembangkannya solusi yang lebih baik daripada pengembangan tradisional dengan model 1 menggunakan JSP, yang bukan hanya lebih lambat tetapi juga kotor.

Kemungkinan ada aplikasi yang dikembangkan disalah satu perusahaan dan sudah stabil baik itu dengan framework MVC atau dengan JSP, berkemungkinan ditemukan. Malahan sebenarnya sebuah solusi dapat dikembangkan bukan dengan teknologi Java. Akibatnya muncul banyak lautan informasi yang tidak terintegrasi, yang mana perpindahan data dari satu lautan informasi ke lautan informasi akan memberikan value yang lebih besar, sebagai contoh adalah birokrasi pemerintahaan, akan semakin baik dengan adanya integrasi sistem, yang mana mungkin saja departemen A mengembangkan dengan FoxPro, departemen B dengan Oracle Developer dan departemen C baru

menggunakan MVC. Semua ini adalah kebebasan memilih dari sang pemakai sistem.



Pulau Informasi

Telah muncul sebuah teknologi yang memungkinkan pulau-pulau informasi dapat saling bertukar data, dan hebatnya teknologi ini memungkinkan terjadi pertukaran data terhadap teknologi yang tidak rukun seperti .NET dan Java, PHP dengan Python. Teknologi ini disebut Web Services, tetapi evolusinya akhirnya berubah menjadi SOA singkatan dari Services Oriented Architecture.

SOA ini adalah sebuah jargon yang sedikit membingungkan, karena berdiri dari sebuah kata services yang berarti pelayanan. Dimana setiap vendor punya hak untuk menganggap setiap bagian kecil komponennya adalah SOA, hal ini karena bersifat melayani.

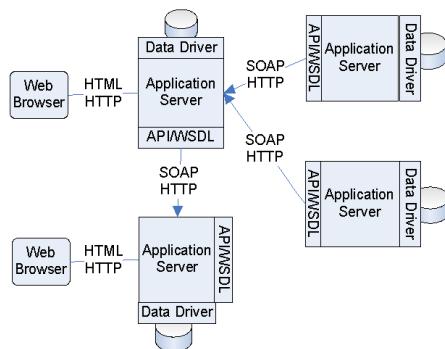
Beberapa implementasi SOA yang mungkin dapat diterima sebagai SOA, diantaranya adalah sebuah scheduler yang bekerja seperti daemon yang melakukan sebuah kegiatan secara berulang. Yang lain, yang paling banyak mendapat sorotan dan

berkembang sangat pesat adalah sebuah teknologi open yang merupakan turunan dari XML, yang dikelola oleh W3C, yaitu SOAP.

SOAP merupakan sebuah meta data yang bekerja seperti amplop dalam surat menyurat, yang bekerja diatas protokol TCP/IP, tetapi banyak juga yang mengextendnya sehingga berjalan diatas protocol HTTP.

Mekanisme yang berjalan diatas HTTP ini yang membuat SOAP menjadi sebuah teknologi yang hot, sampai tahun 2005 dianggap tahun SOA Internasional.

Berikut ini adalah diagram bagaimana SOAP bekerja antara satu sistem dengan sistem lain. Implementasi SOAP ini terhadap pulau-pulau informasi memungkinkan data berpindah dan terjadi interoperabilitas.



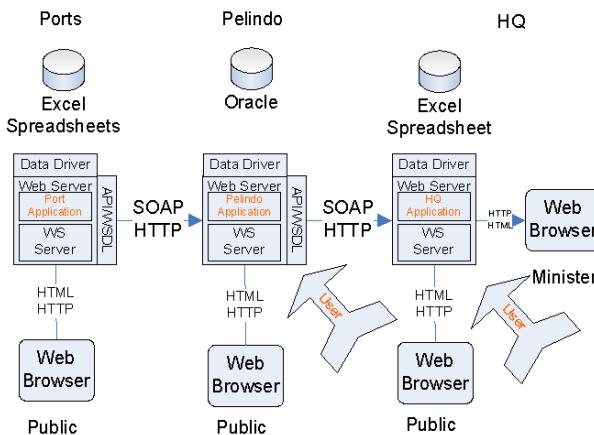
Arsitektur dasar dari SOA untuk Interoperabilitas

SOAP dalam SOA, bekerja seperti halnya request dan response pada HTML, tetapi tentu saja format yang dikirim bukan berdasarkan skema HTML, tetapi SOAP.

Berikut adalah skema interoperabilitas dari 3 entitas didalam

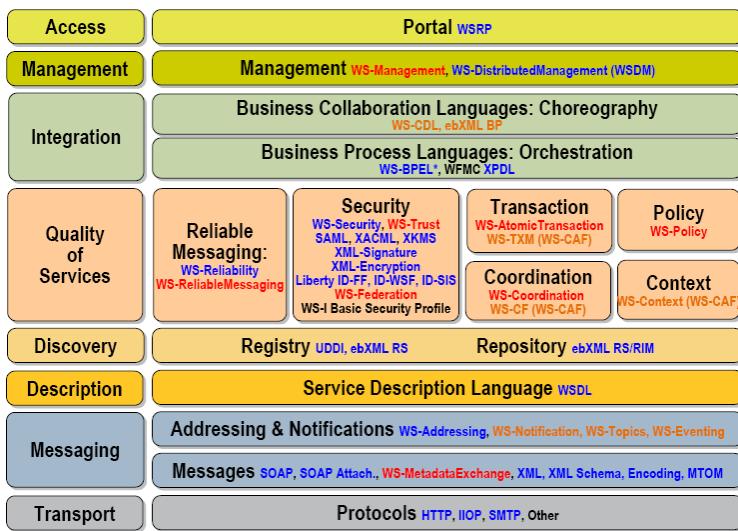
departemen perhubungan, yang merupakan salah satu kasus pengembangan PASIR. PASIR adalah inisiatif seperti SOA yang bermula dari gerakan IGOS, tetapi akhirnya menjadi salah satu dari inisiatif Open Aptel di Depkominfo. Skema ini merupakan salah satu dari prototype yang didanai oleh CIDA (Canada Indonesia Development Aid).

Ke-3 entitas itu adalah Pelabuhan/Galangan Kapal, PT Pelindo dan Departemen Perhubungan. Dimana data dari galangan kapal dikirim ke pelindo, terus dilanjutkan ke departemen perhubungan menggunakan teknologi SOAP.



Prototype Interoperabilitas didalam Departemen Perhubungan

Sebenarnya prototype diatas merupakan prototype yang paling sederhana dari implementasi SOA. Berikut adalah stack dari SOA yang sedang dikembangkan didalam projek PASIR, yang mana diagram tersebut dikembangkan oleh Sun Microsystems.



WebServices Stack dari Sun Microsystems

SOAP bekerja untuk tujuan point-to-point, tetapi untuk sebuah mekanisme interoperabilitas yang lebih komplek, yang mana setiap informasi akan dapat diproses bilamana melakukan fetching terhadap lebih dari SOAP listener, diperlukan sebuah standar baru bernama BPEL. BPEL ini seperti halnya workflow dalam organisasi, yang memonitor dan menentukan logika bilamana proses request ke sebuah layanan menggunakan SOAP telah selesai.

Adapun leveling dari semua layanan yang dianggap sebagai fondasi web services adalah sebagai berikut:

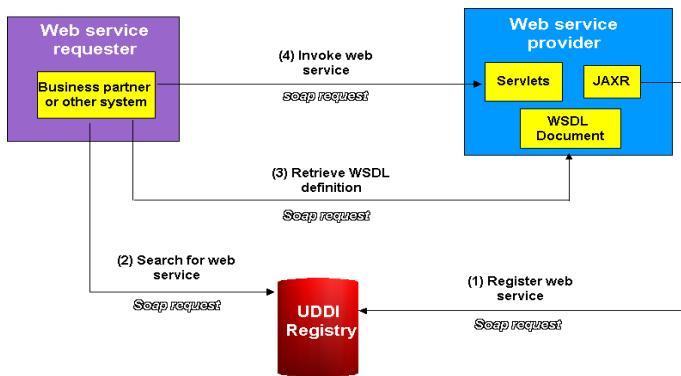


Stack Spesifikasi Web Services Interopabilitas dari WS-I

Terkadang dalam implementasi sebenarnya, terkadang diperlukan banyak sekali layanan web services., setiap layanan yang dideskripsikan dalam bentuk WSDL (Web Services Description Language) ini menjadi sangat membingungkan, karena setiap pihak dapat membuat URLnya sendiri-sendiri dan tidak ada aturannya. Untuk itu lahirlah UDDI, yang secara awam dapat dikatakan sebagai tempat untuk mendaftarkan URL dari WSDL.

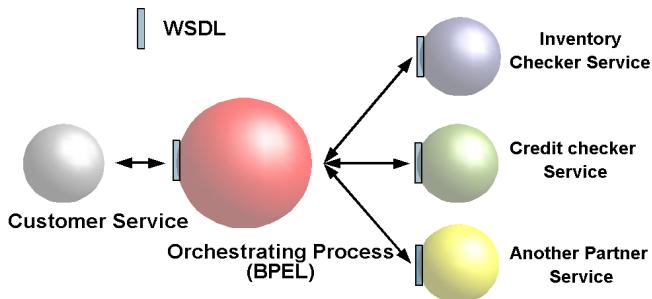
Bilamana UDDI telah dikembangkan, artinya setiap request hanya perlu melakukan pemetaan di UDDI tersebut. Cara kerjanya mirip dengan manajemen domain, seperti .com, .org diatur oleh InterNIC, sedangkan .or.id, atau .co.id dimanage di IDNIC.

Sebaiknya tentu saja bilamana implementasi web services telah banyak, hal ini untuk membuat URL WSDL yang tidak enak dibaca mata, menjadi lebih manusia.



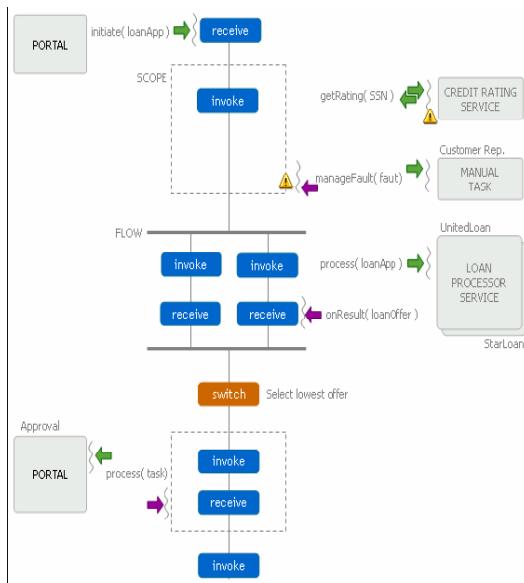
Cara Kerja UDDI dalam proses request dan response pada solusi SOA

Berikut adalah diagram bagaimana tiga layanan berbasis SOAP yang diproses menjadi sebuah informasi menggunakan sebuah BPEL, yang menghasilkan sebuah konsolidasi layanan.



Interaksi BPEL dalam SOA

Inisiatif BPEL merupakan salah satu mekanisme untuk pengembangan solusi Single Windows atau centralisasi operasional. Berikut adalah bentuk dari BPEL, yang mana bilamana kita sedikit cerdik, dapat menggunakan skema BPEL sebagai single window monitoring system



BPEL dengan sumber informasi (SOA)

Object Wrapper

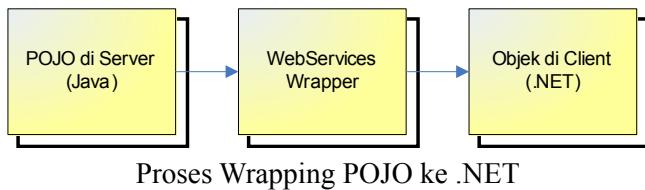
Memang banyak sekali teknologi dibalik SOA, malah berdasarkan pengalaman penulis terhadap dunia IT. Semua teknologi tersebut suka atau tidak suka harus diadopsi, dan ini tentu saja memerlukan resource yang sangat besar untuk mengembangkannya, belum lagi model implementasi yang dapat bermacam-macam. Tentu saja, ini diluar dari kemungkinan sistem dihajar oleh para hacker (Indonesia adalah negara no. 1 didunia untuk urusan per-hacker-an). Ini yang membuat implementasi Web Services menjadi lebih rentan lagi. Akibatnya implementasi SOAP untuk level yang lebih tinggi salah satunya Web Services Security harus dipasang.

Sebenarnya kalau kita membicarakan mengenai ini semua, yaitu interchange data dari satu titik ke titik lain, baik itu satu point ke banyak point, atau one-to-one point, atau banyak point ke satu

point. Hanyalah satu konsep sederhana untuk membuat semua ini menjadi lebih mudah dimengerti, semuanya balik lagi ke POJO (Plain Old Java Objek). Yang mana dalam pengembangan MVC, lebih cenderung disebut sebagai Model.

POJO ini ternyata yang disimpan didalam server. POJO ini yang ternyata yang diakses oleh client pengakses Web Services. Jadi dengan lain kata Web Services adalah sebuah pembungkus objek (Object Wrapper) yang membuat POJO ini berubah menjadi SOAP.

Yang mana, karena SOAP ini merupakan sebuah standar terbuka, membuat SOAP yang diwrap, dimana didalam wrapper tersebut adalah POJO, membuat POJO tersebut dapat diakses oleh aplikasi apapun baik itu .NET, Java, Python ataupun PHP. POJO yang diakses via SOAP ini, akan menjadi sebuah plain object juga untuk pengaksesnya.



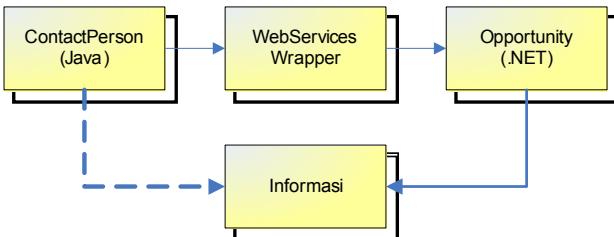
Proses Wrapping POJO ke .NET

WebServices merubah sebuah POJO menjadi object teknologi .NET

Wrapper ini ternyata merubah objek dari teknologi pembuatnya misalnya Java, menjadi teknologi lain yang mungkin tidak dapat memanggil objek tersebut bilamana dipasang dalam satu server, menjadi bagian dari teknologi yang mengaksesnya.

Sebagai contoh adalah, bilamana kita telah mempunyai aplikasi manajemen kontak dengan Java, dan kita hendak mengembangkan versi berikutnya menggunakan .NET yang

bekerja mencatat peluang perusahaan. Sayangnya kita tidak ingin aplikasi yang telah ada dibuat ulang. Dimana aplikasi CRM yang diinginkan adalah kombinasi peluang (.NET) dan nama kontak (Java).



Pengolahan Informasi menggunakan Web Services

Bilamana kasus diatas dilakukan tanpa web services. Hal yang mungkin dilakukan adalah merubah objek Contact yang dikembangkan dengan Java, menjadi COM atau .NET compliance, sehingga aplikasi .NET dapat menggunakannya untuk kebutuhan pemrosesan aplikasi peluang perusahaan yang dikembangkan.

Bagaimana bilamana ternyata setelah itu, kita mengembangkan aplikasi berikutnya dengan Ruby, misalnya aplikasi e-procurement, apa yang akan terjadi? Apakah kita harus merubah semua aplikasi Java dan .NET menjadi sebuah komponen yang dikenal Ruby, tentu saja tidak mimpi buruk yang akan terjadi.

Web Service datang dengan merubah setiap objek yang dikembangkan teknologi lain, menjadi dapat diakses oleh teknologi lain, dan tanpa perlu merubah menjadi komponen yang dikenal, hanya dirubah menjadi Web Services, maka secara otomatis komponen tersebut dapat digunakan.

Dengan Web Services, komunikasi komponen seperti kasus diatas yaitu Java - .NET – Ruby, dapat dilakukan tanpa perlu

kita mencari teknologi wrapper yang ada.

Yang lebih hebat, Web Services ini berjalan pada port 80, yaitu port HTTP. Tentu saja ini artinya kita dapat memasang aplikasi Web Services pada internet. Dengan kombinasi HTTPS dan Web Services yang diencrypt atau diberikan signature (XML Signature), ataupun ditambahakan spesifikasi implementasi Web Services Security, membuat komunikasi antara komponen Java, .NET dan Ruby menjadi lebih aman.

Tentu saja, bilamana dilakukan dengan wrapper tradisional seperti COM converter, akan sangat sulit sekali.

Mengembangkan Web Services Sendiri

Berikut adalah sebuah POJO atau JavaBean yang akan dirubah menjadi Web Services yang akan dibaca oleh aplikasi lain.

Komponen POJO sumber:

```
package org.blueoxygen.gemilang;

public SoapServer {
    private String name="BBook2";
    public String getName() {
        returns name;
    }
    public setName(String name) {
        this.name=name;
    }
}
```

Sedangkan pengakses adalah sebuah aplikasi Java juga, berikut adalah cuplikan kodennya

```
public static void main(String[] args) {
    Service service = new Service();
    Call call;
    try {
```

```
call = (Call) service.createCall();
call.setTargetEndpointAddress(
    "http://localhost:5794/AxisTest/services/
SoapServer ");

try {
    System.out.println("SetNAme");
    call.setOperationName("setName");
    call.invoke(new Object[]{
        new String("Leo") });
    System.out.println("GetNAme");
    call.setOperationName("getName");
    Object name = call.invoke(new Object[] {});
    System.out.println("Client:"+name);
} catch (RemoteException e) {
    e.printStackTrace();
}
} catch (ServiceException e) {
    e.printStackTrace();
}
}

}
```

Bilamana kita lihat secara kasat mata, apakah 2 kode diatas dapat berinteraksi? Jawabannya tidak ada hubungannya. dengan merubahnya menjadi sebuah Web Services, objek POJO kita dapat diakses atau lebih disebut juga di invoke. Dengan merubah POJO pertama menjadi sebuah Web Services, membuat aplikasi lain dengan mengacu pada `http://localhost:5794/AxisTest/services/SOapServer`, membuat aplikasi dapat mengakses POJO yang berada diserver.

Tentu saja, ada perbedaan yang mencolok terjadi, bilamana kita mengembangkan semua objek dalam satu kontainer, dan aplikasi adalah aplikasi Desktop seperti Postila, tentu saja POJO yang diakses akan terus aktif dan isinya akan dapat dipakai oleh seluruh aplikasi, sampai aplikasinya dimatikan.

Sedangkan dengan Web Services ini, ternyata variable didalam POJO tidak aktif, dan setiap request ke services melalui

mekanisme Web Services, akan kembali ke nilai awalnya, alias isi dari variable name pada kasus diatas adalah BBook2, akan terus BBook2.

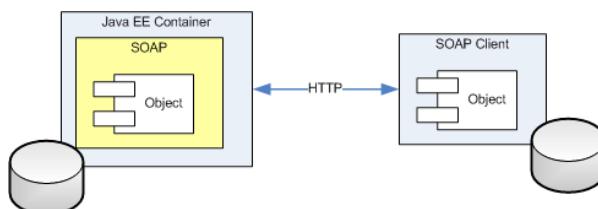
Hal ini terjadi karena Web Services bekerja seperti halnya request dan response pada HTML. Jadi dengan lain kata Web Services client diatas mirip dengan cara kerja browser, hanya dalam kasus browser, yang dihasilkan adalah hasil final, sedangkan dalam Web Services yang dihasilkan adalah objek.

Untuk mengatasi hal ini, setiap POJO yang ditaruh didalam server Web Services harus disimpan didalam session.

Mengembangkan SOAP dengan Eclipse Calisto

SOAP adalah satuan terkecil dari pengembangan aplikasi berbasis Web Services yang paling umum dilakukan.

Sebenarnya mekanisme penggerjaan SOAP adalah merubah sebuah objek menjadi berbasis SOAP. Jadi dapat dikatakan SOAP adalah mekanisme mengakses objek dengan menggunakan XML.



Cara Kerja Interaksi antara aplikasi berbasis SOAP

Dikarenakan mekanisme penggerjaan adalah berbasis XML, dan implementasi SOAP API dapat dilakukan oleh banyak pihak menggunakan teknologi yang bermacam-macam. Hal ini dikarenakan XML yang menyusun SOAP adalah sebuah flat file biasa.

Berikut adalah sebuah objek Buku, yang akan dibungkus dan dirubah menjadi sebuah SOAP, yang kemudian diakses oleh aplikasi lain.

```
package org.blueoxygen.bbook2.webservices;

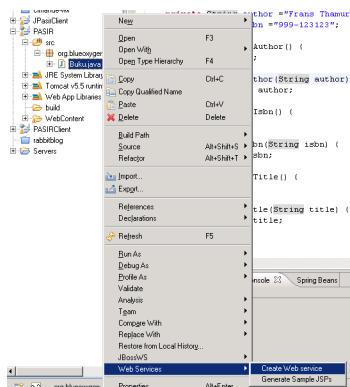
public class Buku {

    private String title = "Cara Cepat
Mengembangkan Java Enterprise dengan Metode MVC";
    private String author ="Frans Thamura";
    private String isbn ="999-123123";

    public String getAuthor() {
        return author;
    }
    public void setAuthor(String author) {
        this.author = author;
    }
    public String getIsbn() {
        return isbn;
    }
    public void setIsbn(String isbn) {
        this.isbn = isbn;
    }
    public String getTitle() {
        return title;
    }
    public void setTitle(String title) {
        this.title = title;
    }
}
```

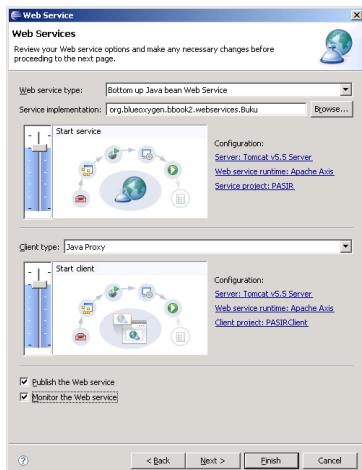
Setelah objek Buku dibuat, klik kananlah file tersebut, dan pilihlah Web Services, untuk masuk ke wizard wrapping dari objek Java menjadi sebuah SOAP.

Cara Cepat Mengembangkan Aplikasi Java dengan Metode MVC



Menu untuk merubah objek menjadi Web Services

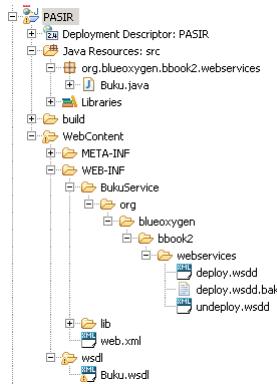
Sebuah wizard akan muncul, dan ada 2 implementasi yang memungkinkan yaitu top down atau bottom up. Bilamana dari objek pilihlah bottom up, bilamana dari WSDL, pilihlah top down.



Web Services Wizard

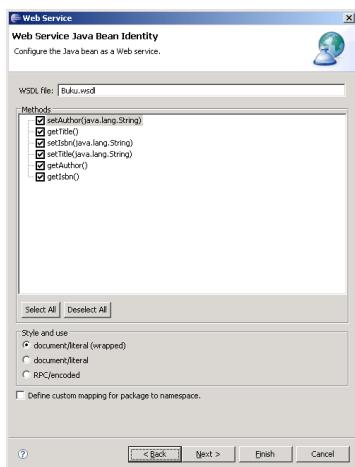
Java Proxy adalah implementasi akses Buku dengan SOAP. Sebenarnya sifatnya adalah pilihan, sehingga tidak perlu diakses.

Proses berikutnya adalah membuat WSDL, jadi objek Buku yang telah dibuat akan dibuat menjadi WSDL. Sebenarnya ada satu step tersembunyi yang tidak terlihat yaitu membuat WSDD (Web Services Description Descriptor), sebuah file yang digunakan sehingga SOAP aktif dapat diakses didalam Axis.



WSDD dalam Project PASIR

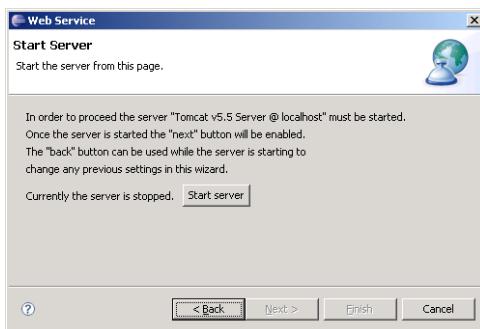
WSDL sebenarnya adalah sebuah XML informasi tentang method dan lokasi server, termasuk portnya, yang bilamana dipakai oleh aplikasi pengakses, dapat dengan mudah mengidentifikasi bagaimana pengaksesannya.



WSDL Generator

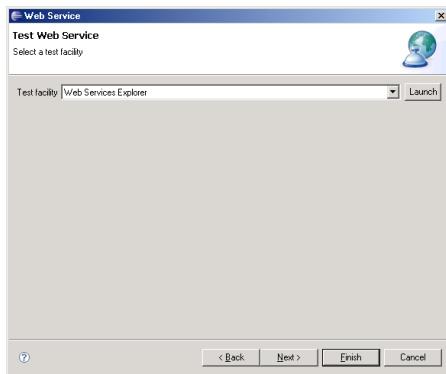
Bilamana pembuatan WSDL telah selesai, server Java EE (dalam kasus ini Tomcat), harus dinyalakan, bilamana belum, ada satu dialog yang meminta untuk menyalakan sebelum mengaksesnya.

Metode ini diperlukan bilamana kita hendak mencoba SOAP yang digenerate oleh Eclipse ini.



Dialog untuk menjalankan Java EE Server

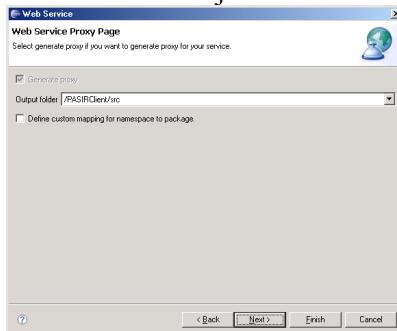
Eclipse datang dengan fitur untuk melakukan testing terhadap SOAP yang dihasilkan.



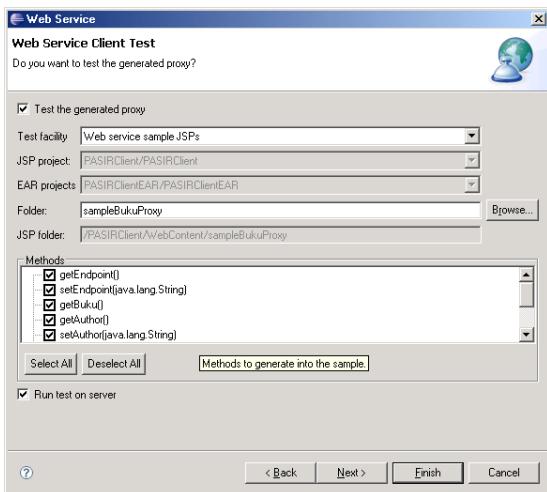
Bentuk dari Aplikasi test Web Services

Umumnya Eclipse akan membuat projek Java (Dynamic) baru yang mengikuti nama projek sebelumnya, bilamana PASIR maka akan dibuat PASIRClient.

Isinya adalah file JSP dan contoh aplikasi mengakses ke server SOAP, dalam kasus ini adalah objek Buku dalam projek PASIR.

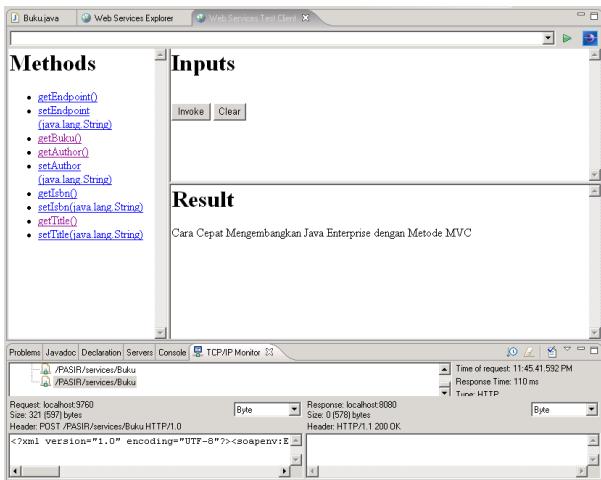


Lokasi Source SOAP Client



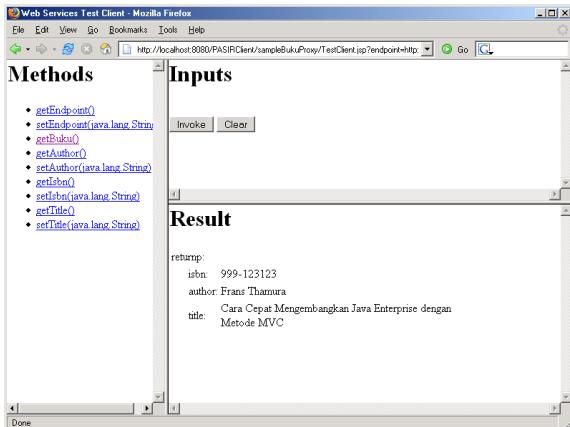
Java SOAP Proxy Generator

Bilamana telah sukses, akan keluar sebuah view berbentuk browser yang akan memunculkan 3 frame, yang kiri adalah objek didalam SOAP, dan yang kanan bawah adalah result dari request.



Aplikasi Test Web Services didalam Eclipse dan

Bilamana hendak mengakses diluar Eclipse, dapat mengetik <http://localhost:8080/namaproject/sampleNamaObjectProxy/TestClient.jsp>.



Aplikasi Test yang diakses melalui browser

Bilamana hendak mempelajari bagaimana request SOAP bekerja, berikut adalah SOAP yang digunakan untuk mengakses server SOAP.

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">

<soapenv:Body>
    <getAuthor xmlns="http://webservices.bbook2.blueoxygen.org"/>
</soapenv:Body>
</soapenv:Envelope>
```

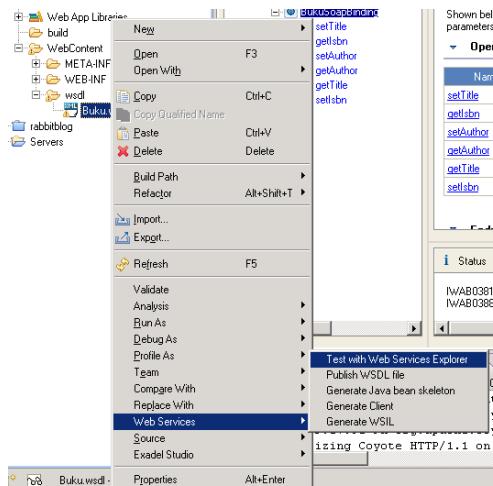
Bilamana diproses, SOAP diatas akan mereturn sebuah SOAP lagi, yang disebut SOAP response.

```
<?xml version="1.0" encoding="utf-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.
xmlsoap.org/soap/envelope/" xmlns:xsd="http://
www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.
w3.org/2001/XMLSchema-instance">

<soapenv:Body>
<getAuthorResponse xmlns="http://webservices.bbook2.
blueoxygen.org">
<getAuthorReturn>Frans Thamura</getAuthorReturn>
</getAuthorResponse>
</soapenv:Body>
</soapenv:Envelope>
```

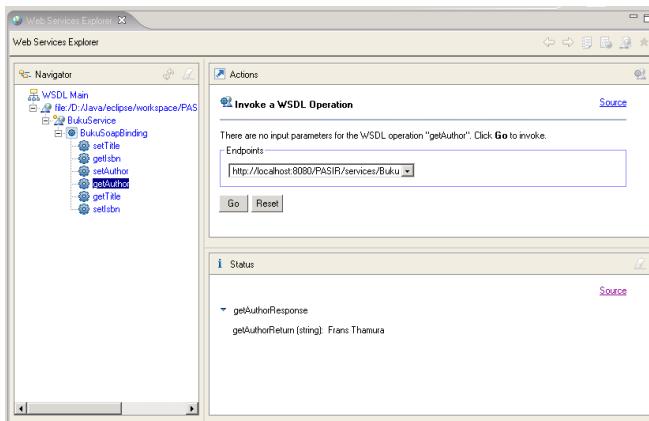
Coba perhatikan tulisan getAuthorReturn, disana ada isi dari jawaban permintaan sebelumnya.

Sebenarnya ada cara lain untuk melakukan testing terhadap SOAP yang dikembangkan, tanpa perlu membuat sample. Yaitu dengan Web Services Explorer. Caranya adalah dengan mengklik kanan file wsdl yang degenerate, umumnya berada pada folder /WebContent/wsdl.



Menu untuk mengakses Web Services Explorer

Bilamana telah dipilih, view WebServices Explorer akan muncul, yang bekerja mirip dengan sample client sebelumnya.



View Web Services Explorer

Bilamana proses testing telah selesai, sebenarnya SOAP yang

dihasilkan dapat digunakan untuk pengembangan aplikasi interoperabilitas.

Berikut adalah sample untuk mengaksesnya

```
package org.blueoxygen.bbook2.ws.pasir;

import org.apache.axis.client.Call;
import org.apache.axis.client.Service;
import javax.xml.namespace.QName;
import java.rmi.RemoteException;
import javax.xml.rpc.ServiceException;

public class PASIRTest {

    public static void main(String [] args) {
        Service service = new Service();
        Call call;
        try {
            call = (Call) service.createCall();
            call.setTargetEndpointAddress(
                "http://localhost:8080/PASIR/services/Buku");

            try {
                System.out.println("setTitle");
                call.setOperationName("setTitle");
                call.invoke(new Object[]{
                    new String("Buku Java Ke-3") });
                System.out.println("getTitle");
                call.setOperationName("getTitle");
                Object name = call.invoke(new Object[] {});
                System.out.println("Client:"+name);
            } catch (RemoteException e) {
                e.printStackTrace();
            }
            } catch (ServiceException e) {
                e.printStackTrace();
            }
        }
    }
```

Adapun returnnya adalah sebagai berikut

```
- Unable to find required classes (javax.activation.  
DataHandler and javax.mail.internet.MimeMultipart).  
Attachment support is disabled.  
setTitle  
getTitle  
Client:Cara Cepat Mengembangkan Java Enterprise  
dengan Metode MVC
```

Harap diperhatikan, Axis memerlukan 2 komponen tambahan yaitu JavaMail dan Java Activation, dimana kedua-duanya ini adalah teknologi proprietary dari Sun, sehingga kita harus mendownload manual. **Kedua komponen diperlukan bilamana kita hendak mengimplementasikan SAAJ yaitu implementasi SOAP dengan attachment.**

Setelah mekanisme interoperabilitas diatas berhasil, step berikutnya adalah mengisi objek yang dihasilkan dengan dengan informasi yang lebih lengkap, atau mengimplementasikan Web Services Security menggunakan WSS4J atau mengimplementasikan BPEL.

PASIR, SOA Versi Indonesia

Implementasi Web Services, atau yang lebih spesifik merubah sebuah komponen menjadi SOAP ready, ternyata tidak sesulit yang dibayangkan. Sebenarnya yang sangat sulit adalah bagaimana mereturn sebuah method menjadi method lain yang kompatible. Dengan lain kata, merubah metod getName agar dapat diinvoke dan returnnya dikenal oleh aplikasi yang memanggil. Untuk itu setiap programmer yang hendak merubah aplikasi atau komponen dari teknologi A ke teknologi B, harus melakukan mapping tipe data yang direturn. Sebagai contoh adalah bilamana yang direturn adalah sebuah Java Collection, tentu saja bilamana didalam teknologi rekanannya baik itu .NET atau PHP atau Ruby, tidak terdapat fungsi ini, maka komunikasi tidak terjadi.

Padahal didunia Web Services, perihal teknologi adalah bukan masalah utama, sebab inti dari Web Services adalah membuat sebuah aplikasi yang dikembangkan dengan teknologi lainnya dapat dibaca oleh teknologi lain, untuk melakukan hal ini didunia swasta telah muncul standar seperti Web Services untuk Kesehatan, atau Web Services untuk mengirim informasi user (SAML).

Sedangkan bagaimana dengan pemerintah? Di Indonesia, mekanisme ini disebut PASIR yang merupakan singkatan dari Program Arsitektur Sistem Informasi dan Inte(R)operabilitas. Sebuah inisiatif yang muncul bermula dari IGOS, dan ternyata karena dalam dunia Web Services, proprietary atau Open Source bukanlah hal utama, melainkan standar interoperabilitas antara aplikasi telah menjadi bagian dari Aplikasi Telematika milik Depkominfo.

PASIR diharapkan dapat menjadi tempat banyak kasus interoperabilitas yang mungkin akan muncul didalam lembaga-lembaga khususnya pemerintah, dan pada umumnya baik itu pemerintah-rakyat, pemerintah-perusahaan atau mungkin perusahaan-rakyat.

Referensi Implementasi PASIR adalah Open Source dan Open Standard, dan berlisensi bebas untuk dimodifikasi. Saat ini kode sumbernya dapat didownload di <http://www.sf.net/projects/pasir>.

Mekanisme penggeraan projek Open Source PASIR adalah terbuka terhadap kolaborasi, bagi siapa saja yang hendak berkontribusi baik itu menggunakan teknologi Java ataupun yang lainnya adalah terbuka. Tim pengembangan PASIR membuka pintu untuk kolaborasi untuk mengembangkan sebuah mekanisme interoperabilitas yang lebih baik.

Lampiran A

Berkenalan dengan

Java SDK dan variannya

Untuk memulai pemograman Java, diperlukan sebuah software development kit yang sering disingkat SDK. Java SDK ini dapat didownload gratis di beberapa website, maklum saat ini banyak perusahaan ataupun komunitas yang mengeluarkan Java SDK baik yang berstandar J2SE ataupun yang tidak.

Sebenarnya SDK ini dibutuhkan bagi mereka yang ingin membuat program, karena didalam SDK terdapat javac alias java compiler. Java compiler ini akan merubah source code Java menjadi sebuah bytecode, dimana bytecode ini siap digunakan.

Semua SDK yang bersertifikat J2SE adalah compatible, artinya semua binary atau bytecode yang dihasilkan adalah sama. Ini dikarenakan J2SE adalah sebuah standar terbuka yang mengutamakan portabilitas, dan kompatibilitas.

Beberapa SDK yang direkomendasikan adalah yang dikeluarkan oleh Sun, IBM, Apple dan Bea.

Sun Java SDK

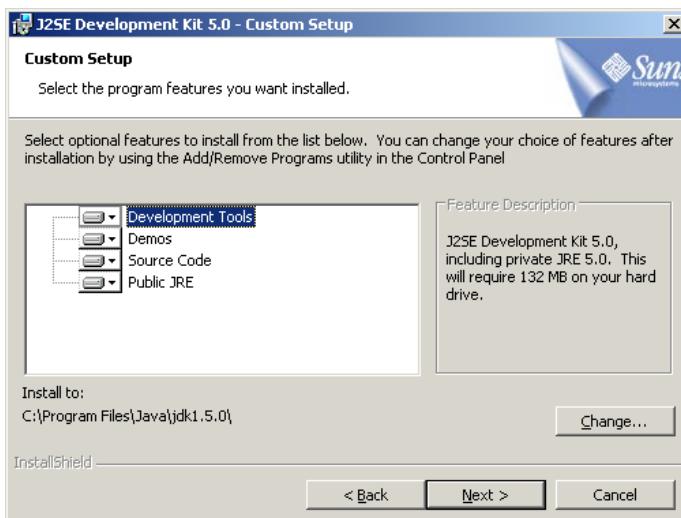
Sun Java SDK adalah Java SDK yang diproduksi oleh Sun Microsystems, dimana Sun adalah pencipta Java itu sendiri. Informasi mengenai Sun Java SDK dapat diakses didalam websitenya <http://java.sun.com>. Sun adalah pemegang hak cipta Java didunia.



Sun Java SDK adalah sebuah produk open source, tetapi tidak bersertifikat OSI. Source codenya dapat didownload diwebsitenya, yang ajaib dari source code Sun Java SDK ini adalah memiliki 3 lisensi yang semuanya tidak mendukung OSI.

Instalasi Sun Java SDK

Untuk melakukan instalasi lingkungan Windows, jalankan JSDK installer, umumnya bentuk filenya adalah jdk-1_5_xx-windows-i586.exe, dimana xx adalah update versinya.



Proses instalasi Java SDK adalah melakukan instalasi Java SDK dan Java RE, dimana didalam Java SDK sendiri terdapat JRE yang sama dengan JRE yang diinstall, ini yang membuat terjadi redundansi. Untuk mengatasi sehingga hanya ada satu JRE dalam setiap PC, gantilah tujuan instalasi ke c:\jdk1.5.0_xx, dimana xx adalah update dari Java SDK.

JRE dibutuhkan untuk programmer yang menghendaki menggunakan aplikasi berbasis Java WebStart. Dimana untuk aplikasi WebStart harus diregister file .JNLP didalam registry.

Instalasi Sun Java SDK di Linux

Proses instalasi Java SDK di Linux terbilang paling mudah, ada 2 versi dari Java SDK yang disediakan oleh Sun, yaitu yang berbentuk binary installer dan berbentuk RPM. Distro berbasis RPM adalah SuSE, RedHat, Fedora, dan Mandrake.

Sedangkan untuk binary installer diperlukan bagi mereka yang menghendaki menginstall Java SDK didalam lingkungan berbasis Debian atau distro Linux umum. Versi RPM tidak berjalan baik di lingkungan linux berbasis Debian. Walaupun telah dilakukan alien, yang konon dapat mengconvert aplikasi .rpm menjadi .deb, tetap saja dalam implementasinya terutama Java SDK ini, tidak akan berhasil. Karena banyak package yang tidak berhasil diconvert.

Sebenarnya file .rpm dari Java SDK lebih sederhana dibandingkan Java SDK berbasis Windows, hal ini dikarenakan .rpm hanya melakukan proses extraksi, dan tidak ada installer windows.

Perintah untuk melakukan instalasi juga sederhana hanya dengan menjalankan jdk-1_5_0-linux-i586-rpm.bin, kemudian file .rpm didapat dan dilanjutkan dengan perintah:

rpm -i jdk-1_5_0-linux-i586-rpm, dan secara otomatis akan didapat folder jsdk-1.5.0 didalam folder /usr/java, tetapi terkadang versi terbaru dari installer berbasis RPM, melakukan extract didalam folder /home/username. Sehingga kita harus memindahkan secara manual ke folder /usr/java.

Ada keuntungan tersendiri menginstall didalam /home/username, terutama yang suka menjalankan aplikasi Java tetapi dengan banyak versi SDK, misalnya /home/frans untuk SDK 1.5, sedangkan /home/cimande untuk SDK 1.4.2. Sedangkan /usr/java umumnya dipakai untuk general use.

Jangan lupa menambahkan JAVA_HOME didalam .bash_profile, setting .bash_profile yang salah akan membuat aplikasi Java dan proses kompilasi tidak berhasil, sedangkan untuk aplikasi berbasis Java WebStart, akan membuat eksekusi .JNLP tidak akan dilakukan.

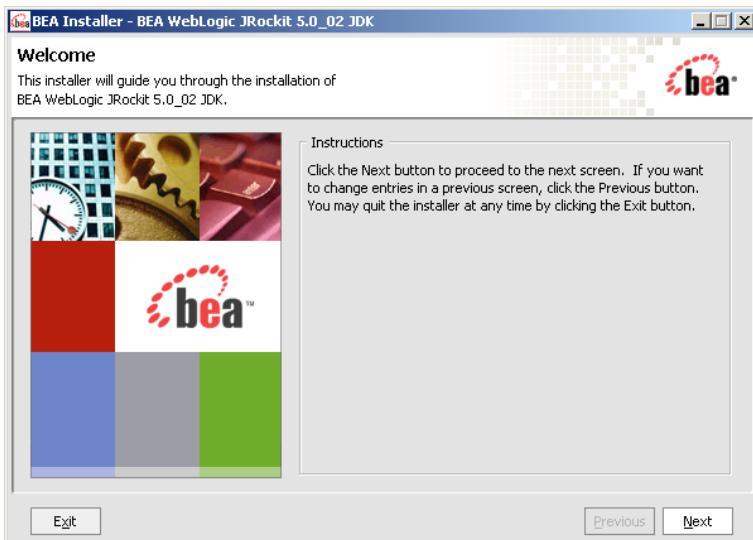
Bea JRockIt

Bea JRockIt adalah Java SDK compatible yang diproduksi Bea Systems, adalah Java SDK compatible artinya yang telah lolos sertifikasi J2SE. Bea JRockIt adalah gratis, dan dapat didownload diwebnya Bea. Bea JRockIt ini memiliki kehebatan tersendiri terutama bilamana dibandingkan dengan Sun Java SDK, dia lebih native, ini dikarenakan Java VM eksekusi standar yang telah dirubah menjadi aplikasi native. Sehingga JrockIt ini lebih cepat dibandingkan Java SDK lainnya pada umumnya.

Bea JRockIt untuk hadir dengan dua versi yaitu versi 32 bit dan versi 64 bit (Itanium) baik untuk Windows maupun untuk Linux. Berita menariknya Sun tidak mengeluarkan Java SDK untuk Itanium untuk versi 5.0, sehingga pemakai Itanium 64bit sebaiknya menggunakan JRockit dari Bea ini.

Instalasi Bea JRockit 5.0 di Windows

Jalankan file jrockit-25.1.0-jdk1.5.02-win-ia32.exe untuk JSDK 32 bit atau 64 bilamana 64 bit. Kemudian akan muncul “choose product directory” yaitu lokasi dimana JRockIt akan diinstall.



Menu Pembukaan Installer Bea JRockit

Kemudian akan muncul pilihan untuk menginstall JRE, JRE ini sebenarnya sudah ada di folder SDKnya, tetapi JRE ini diperlukan bagi mereka yang ingin mengembangkan aplikasi berbasis Java WebStart dan menjalankan aplikasi applet di browser akan berguna. Tips penulis adalah menginstallnya, biarpun akan ada 2 JRE didalam komputer.



Progress Instalasi

Lokasi untuk mendownload Bea JRockIt 5.0 adalah http://commerce.bea.com/products/weblogicjrockit/5.0/jr_50.jsp

Instalasi Bea JRockIt di Linux

Proses instalasi Bea JRockIt di Linux sebenarnya sama dengan Windows, ini yang membedakan installer Sun Java SDK dan Bea JRockIt, Bea telah memberikan installer grafis sehingga proses instalasi dapat dilakukan dengan Wizard.

IBM Java SDK

Sun dan Bea hanya menyediakan Java SDK untuk prosessor dan sistem operasi yang mereka dukung dan menjadi target marketnya, sedangkan diluar itu, vendor-vendor harus membuat Java SDK sendiri, dan caranya dapat juga dengan membeli lisensinya dari Sun. IBM melakukan hal ini, ini dikarenakan IBM memiliki banyak system operasi dan malah memiliki prosessor sendiri PowerPC yang menjadi prosesor untuk Mac,

RS/6000 dan i5-nya. IBM juga memberikan Java SDKnya secara gratis, dengan websitenya dapat diakses di <http://www-128.ibm.com/developerworks/java/jdk/>, IBM membuat versi Java SDK untuk AIX, Linux, zOS, OS/390, OS/400, Windows. Umumnya semua sistem operasi IBM memiliki Java didalamnya, karena IBM salah satu pemain kuat didunia Java. Hampir semua OS, IBM memiliki implementasi Java SDKnya. Malah gosipnya nilai investasi Java yang dikeluarkan IBM adalah yang terbesar, jauh lebih besar dari sang penemunya Sun Microsystems.

Jikes

Jikes adalah compiler supercepat yang dikembangkan tim IBM Research, saat ini ada 2 versi Jikes yaitu Jikes dan JikesRVM (Research Virtual Machine). Dimana Jikes RVM adalah untuk mengeksekusi Java. Jadi bagi programmer yang merasa sebal dengan proses kompilasi Java SDK yang lambat, karena konsep kerja compiler yang mengkompilasi Java source berdasarkan Java Language Specification menjadi bytecode sesuai dengan Java Virtual Machine Specification, pasti kaget dengan performance Jikes ini. Jikes datang hanya dengan satu file jikes.exe dan bekerja secara terintegrasi dengan Java SDK. Jikes sudah terintegrasi dengan Ant yang mana akan dibahas dibab terakhir buku ini. Ada petua Java informal yaitu "Tanpa Ant Bukan Programmer Java belum dapat disebut Programmer Java", bilamana ditambah dan menggunakan Jikes, dapat dikatakan dengan "Jikes membuat Ant menjadi lebih dari yang ada". Jikes adalah OSI certified, artinya dia tidak terikat lisensi, dan dapat dimodifikasi dan dimasukan sebagai bundle. Ini yang membedakannya dengan SDK yang lain.

Jikes dapat berjalan di lingkungan Linux/IA-32, AIX/PowerPC, OS X/PowerPC, and Linux/PowerPC. Informasi mengenai Jikes dapat diakses di <http://jikes.sf.net> dan untuk Jikes RVM dapat diakses di <http://jikesrvm.sourceforge.net/>

Yang menarik dari Jikes RVM, Jikes dapat berjalan diatas BlackDown Java SDK (SDK Open Source yang melisensi kode sumber dari Sun), ataupun Kaffe (SDK GPL). JikesRVM datang dengan rvmeclipse untuk menjalankan Eclipse IDE dengan RVM.

Implementasi Java Lainnya

MacOS Java, GNU Classpath, Sabre, GCJ, Microsoft. GCJ, TurboJ dan TowerJ malah memungkinkan mengkomplilasi java source code menjadi aplikasi native. Saat ini ada implementasi J2SE, sehingga bilamana project ini rampung, maka Java SDK sudah benar-benar bersertifikat OSI. Project ini dibawahi oleh Apache dengan nama project Harmony dan Javali yang didanai oleh pemerintah Brazil.

Appendix B

Memulai Pemograman berbasis Java dengan Eclipse IDE.

Sekilas mengenai Eclipse

Eclipse adalah bukan sekedar IDE untuk Java saja, tetapi sebuah platform pengembangan. Eclipse merupakan sebuah projek Open Source yang diinisiasi oleh IBM, dimana Eclipse merupakan sebuah teknologi yang bernilai US\$ 40 juta, hal ini dikarenakan Eclipse yang merupakan generasi berikutnya dari Visual Age.

Eclipse merupakan IDE paling popular didunia Java, dan nomor 2 setelah Microsoft Visual Studio. Yang lebih hebat dari Eclipse adalah Eclipse datang dengan source code yang berlisensi bebas, artinya Open Source.

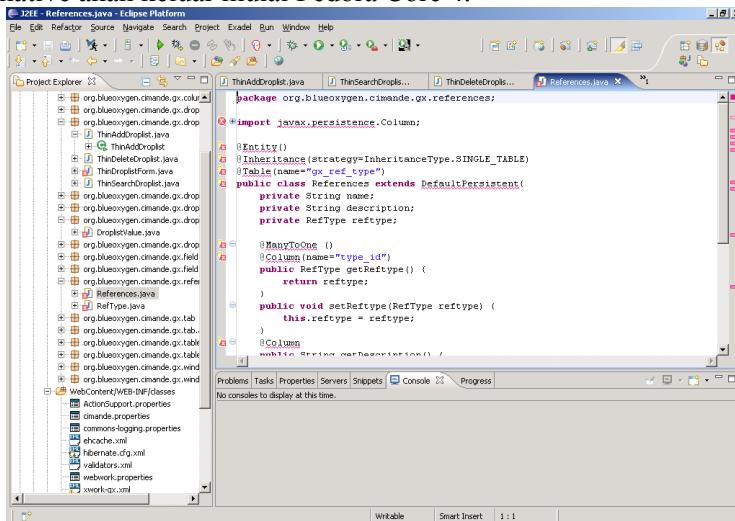
Eclipse diciptakan bukan hanya digunakan untuk mengembangkan aplikasi Java, tetapi juga C++. Bilamana kita mendownload IDE untuk Symbian keluaran Nokia bernama Carbide, sudah jelas ini adalah Eclipse. Bilamana kita hendak mengembangkan teknologi berbasis Netweaver dari SAP, yang sangat terkenal dengan ERP kelas dunianya. SAP Netweaver Studio adalah Eclipse.

Eclipse yang semula adalah projek internal IBM, telah berubah menjadi projek bersama lebih dari 100 perusahaan dibawah naungan Yayasan Eclipse. Sebuah perusahaan nirlaba.

Eclipse datang dengan berbagai macam sub projek, seperti Dali untuk EJB3, Ajax toolkit, AspectJ sampai WTP. Sedemikian

banyaknya plugins Eclipse, membuat Eclipse mengeluarkan bundle plugins + SDKnya menjadi satu, sehingga programmer hanya perlu mengekstrak untuk mendapatkan semua pluginsnya secara terintegrasi. Projek ini bernama Calisto. Buku ini menggunakan Calisto sebagai fondasi pengembangan untuk aplikasi Webnya.

Eclipse merupakan sebuah IDE yang menarik, karena selain bisa dirubah menjadi native IDE, distribusi Eclipse IDE versi native akan keluar mulai Fedora Core 4.



Eclipse IDE

Secara high level, Eclipse memiliki kehebatan diantaranya:

- Multi-Platform (Solaris, Linux, Windows, HPUX, etc)
 - Open Source, CPL permit to distribute
 - Look and Feel, speed of GUI (SWT)
 - Plug-ins Architecture
 - Good Software Design
 - Branding of primary feature
 - Rich UI Framework

- Predefined dialog basis: Wizard, Preferences, Properties
- Other UI: Perspectives, View, Editor, Workbench
- ActiveX support on Win32
- Help System
- Extensible Platform
- Production Quality

Dari semua kehebatan itu, ada yang mengganjal Eclipse yaitu adanya sebuah teknologi yang tidak standar, sebuah mekanisme implementasi yang dapat dikatakan "dosa" di dunia perJavaan. Dan hebatnya yang tidak standar itu adalah inti dari Eclipse adalah SWT yang tidak standar, walaupun tentu saja SWT ini adalah Open Source dan telah membuat momentum di dunia perJavaan yang terkenal dengan kata "lambat". Tanpa SWT seperti Eclipse tanpa nyawa, tidak bisa jalan sama sekali.

Eclipse memang multiplatform, tetapi SWT-nya karena tidak standar, dan tidak termasuk dalam distribusi Java SDK, membuat kita harus menginstall Eclipse berdasarkan versi OS yang ada, jadi kalau SWT-nya belum ada versi OS yang dimaksud, maka bisa dipastikan Eclipse tidak bisa berjalan. Tragis memang untuk sebuah teknologi yang dikatakan multiplatform dan portable.

Jadi bagi para programmer Java yang menginginkan menggunakan Eclipse, sebaiknya melihat kedalam projectnya apakah sudah ada SWT versi OS tersebut, atau bagi mereka yang tidak mau pusing, download versi yang ada sesuai Osnya, dan kalau tidak ada, lupakan Eclipse.

SWT yang merupakan sebuah teknologi yang bukan turunan dari Java, tetapi dari Smalltalk yang sangat populer dengan framework MVCnya dan merupakan teknologi VM paling populer dijamannya saat itu. Sejarah teknologi ini yang membuat pihak Eclipse bersikukuh terhadap SWT, dan pihak

JCP merasa Eclipse adalah teknologi yang merusak moto promosi Java.

Didalam Eclipse ini ada sebuah plugins wizard yang membuat kita mudah membuat plugins baru yang dapat diembed kedalam Eclipse, fitur yang sangat membantu pengembang mengembangkan plugins tambahan untuk Eclipse, hal yang membuat Eclipse menjadi sangat populer didunia perJavaan, pluginsnya banyak, malah ada website yang memanage plugins-plugins Eclipse yaitu Eclipse-plugins.info, walaupun Eclipse memiliki dosa menggunakan teknologi yang standar, yaitu SWT. Dengan kemudahan, kecepatan, sepertinya dosanya sedikit dimaafkan. Malah penulis sendiri lebih terbiasa dengan Eclipse dibandingkan Netbeans.

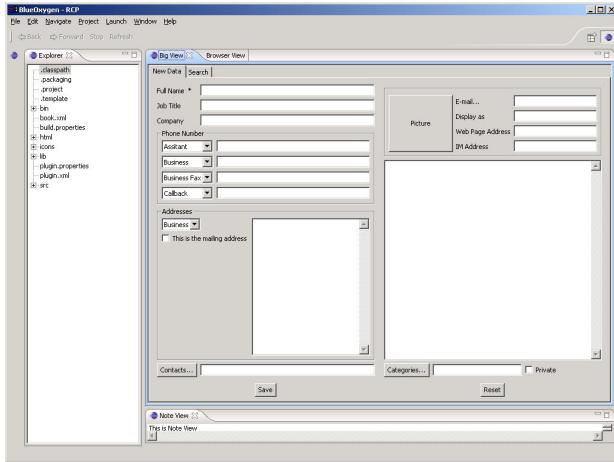
Rich Client Technology

Sebagai informasi tambahan Eclipse adalah sebuah IDE yang berlisensi Common Public License, artinya semua pihak dapat memiliki dan merubahnya dan menjualnya tanpa perlu membayar sepeserpun ke tim Eclipse. Ini menarik sekali, sehingga Eclipse menjadi benar-benar bebas. Malah kebebasan yang diusungnya membuat banyak perusahaan membuat Eclipse sebagai fondasi aplikasinya seperti Lotus Notes yang menggunakan Eclipse sebagai engine untuk Lotus Notes Client versi 7. Jadi, dengan lain kata, Eclipse dapat dirubah dari sekedar IDE menjadi sebuah framwork pengembangan aplikasi standalone atau sering disebut juga Rich Client Technology atau Rich Internet Application.

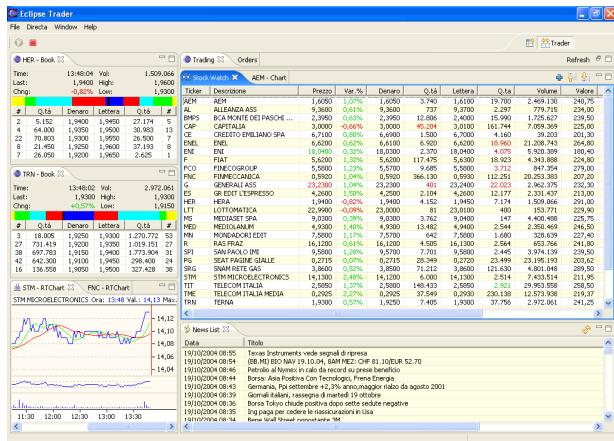
Plugins management Eclipse benar-benar ciamik, dengan total file yang lebih dari 150MB, akan berjalan sangat cepat dienvironment 128MB, hal yang mustahil apalagi didunia perJavaan yang terkenal boros memory. Teknologi ini disebut Eclipse RCP (Rich Client Platform). Teknologi saingen Eclipse RCP adalah Netbeans Platform, yang juga bekerja mirip seperti

ini.

Tentu saja Eclipse RCP ini dipilih untuk mengutamakan kelebihan dari SWT yang cepat dan mengorbankan portabilitas.



Aplikasi SMS yang dikembangkan dengan Eclipse RCP



Aplikasi Online Trading dengan Eclipse RCP

Berkenalan dengan Eclipse

Bilamana hendak mengembangkan aplikasi berbasis Java menggunakan Eclipse, sebenarnya sangatlah mudah, yang diperlukan adalah sebuah Java SDK terbaru dan Eclipse SDK sesuai dengan OS yang dipasang. Untuk lingkungan *nix, dapat dipilih Eclipse yang dikembangkan dengan GTK atau dengan Motif. Eclipse SDK versi Motif jauh lebih cepat tetapi layoutnya kurang enak dipandang.



Eclipse IDE untuk Windows

Menginstall Eclipse IDE

Untuk menginstall Eclipse IDE, yang diperlukan adalah sebuah Java SDK dan sebuah archive Eclipse SDK, yangmana Eclipse SDK ini dapat didownload di websitenya yaitu <http://www.eclipse.org>.

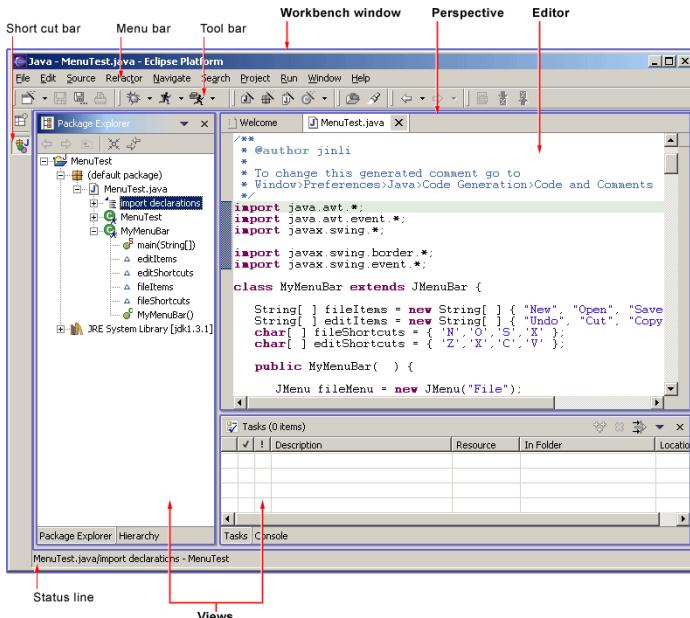
Setelah didownload, extract file kefolder yang diinginkan, dan jalankan file eclipse untuk menjalankannya.

Sebuah dialog box yang mengatakan default lokasi dari

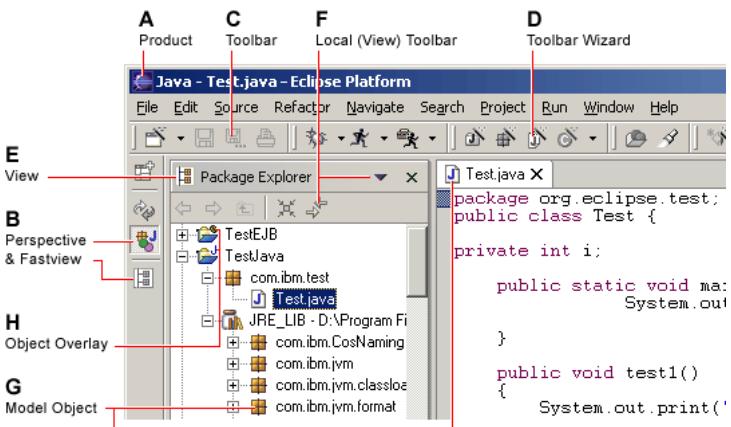
workspace Eclipse akan muncul, isilah atau pilih Ok untuk melanjutkan. Setiap melakukan eksekusi Eclipse, akan dibentuk folder configuration, dimana semua setting Eclipse akan diset disini, seperti salah satunya adalah lokasi workspace

Standarisasi pada Eclipse IDE

Eclipse memiliki banyak standar yang benar-benar sangat membantu dalam pengembangan seperti icon tanda silang merah untuk error code, perspective yang plugable, serta beberapa kemampuan seperti icon-icon berwarnanya.



Eclipse IDE Workbench



Eclipse IDE dengan Java Perspective lebih detail.

Eclipse IDE datang dengan 8 perspective standar yaitu:

- CVS Repository Exploring Perspective
- Debug Perspective
- Java Perspective
- Java Browsing
- Java Type Hierarchy
- Plug-in Development
- Resource Perspective
- Team Synchronizing

Perspective ini akan bertambah searah dengan banyaknya plugins yang dipasang didalam Eclipse.

Icon-Icon standar Eclipse yang diperlukan saat melakukan pemograman. Setiap icon didalam eclipse akan diberikan imbuhan icon saat dipakai, seperti kalau dalam pemograman Java berbasis kelompok menggunakan CVS, akan ditambahkan icon kecil berwarna kuning. Atau bila ada sebuah code Java yang error, akan ditambahkan imbuhan icon berwarna merah

berbentuk tanda silang dibawah icon standar diatas. Coba lah mencoba beberapa kasus untuk membiasakan diri dengan icon-icon yang ada.

create, new		compare		forward		jar		plugin	
save		debug		backward		WAR		extension	
cut		run, execute		previous		EAR		extens'n point	
copy		import		next		window		thread	
paste		export		project		perspective		process	
add		play, resume		open project		property sheet		mapping	
remove		suspend		folder		table		error	
delete		terminate		open folder		database		warning	
erase, clear		stop		file		repository		alert	
search		undo		library		class		conflict	
find		redo		package		interface		public	
help		refresh		session bean		attribute		protected	
edit		filter		server		element		private	
								default	

Icon-icon standar Eclipse

Memulai bekerja dengan Eclipse

Kita tahu Eclipse IDE adalah sebuah Java IDE yang handal yang memungkinkan setiap pemakainya mendapatkan produktifitas yang tinggi dan standar, hal ini terjadi hanya dengan membiasakan menggunakan IDE dalam pemogramman. IDE akan sangat terasa gunanya bilaman kita mengembangkan sebuah projek Java yang bersifat kelompok, dimana satu aplikasi Java dipakai beramai-ramai oleh lebih dari 2 orang, dan kadang kala setiap perubahan harus dapat didistribusikan ke programmer lainnya.

Untuk memulainya, downloadlah Eclipse IDE SDK sesuai dengan OS yang dimaksud, tetapi sebelumnya jangan lupa Java

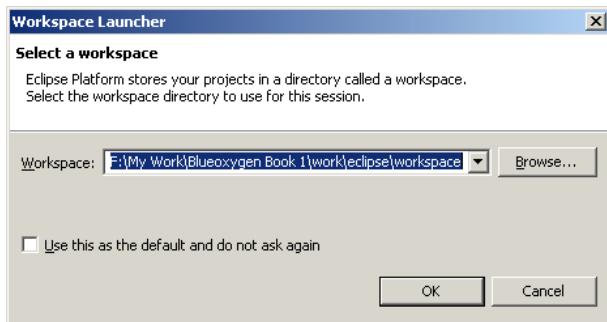
SDKnya baik itu Sun Java SDK, IBM Java SDK, ataupun BeJrockit harus sudah terpasang di komputer, dan berjalan baik.

Setelah itu extractlah file SDK tersebut dan extractlah ke dalam sebuah folder baru, misalnya C:\work.

Secara otomatis file tersebut akan membuat folder eclipse, plugins, dan feature. 3 folder ini adalah folder utama, kalaupun ada folder yang lain yang lain dapat diabaikan. Untuk mengextract dapat menggunakan WinZip atau file extractor lainnya.

Setelah itu bukalah folder eclipse, di root atau di folder eclipse tersebut akan didapat file executable eclipse.exe, jalankan file ini. Secara otomatis Eclipse akan berjalan, bila ada error, umumnya Java SDKnya belum tersetting dengan baik.

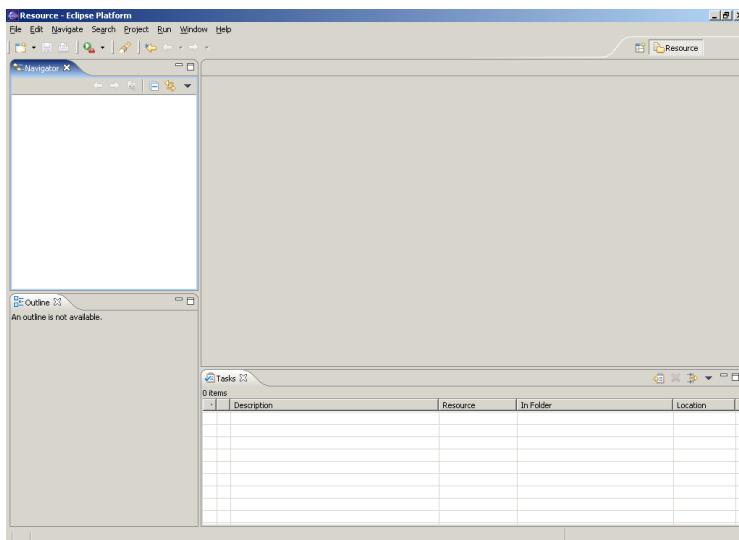
Untuk pertama kali eksekusi, Eclipse akan meminta file lokasi workspace, workspace ini artinya tempat default atau tempat awal bila sebuah project Eclipse akan dimulai. Tekan tombol Ok bila ingin melanjutkan.



Setelah itu Eclipse Welcome Page akan muncul.

Welcome page ini hanya akan muncul satu kali saja, tekan

tombol silang (x) di pojok kiri atas, dan secara otomatis Eclipse Workbench akan muncul. Eclipse Workbench adalah tempat dimana kita akan memulai pemrograman Java.

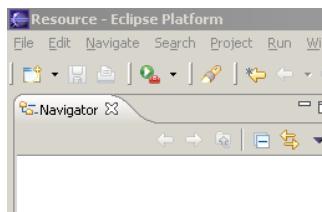


Eclipse Workbench

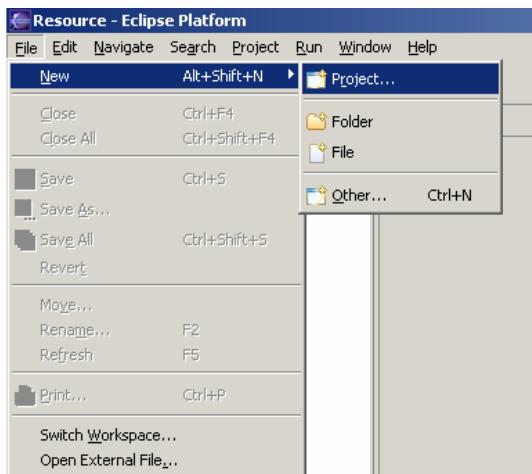
Eclipse Workbench akan secara otomatis masuk kedalam modus Resource, yaitu perspective untuk browsing folder dan file. Biarkan saja modus ini tanpa perlu dimodifikasi. Eclipse akan secara otomasi masuk ke modus Java bilamana kita bekerja membuat program Java.

Memulai Projek Java

Untuk memulai project Java, dapat dilakukan dengan menekan tombol New di toolbar atau dengan memilih New lalu Project pada menu File.

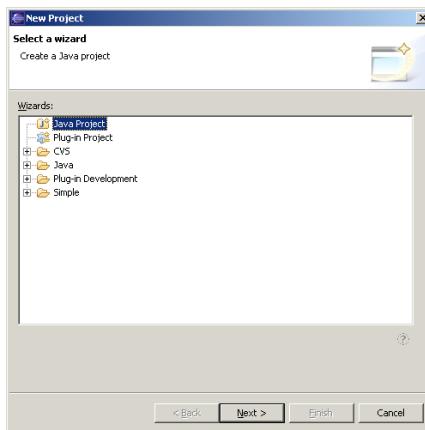


New Icon untuk membuat Project



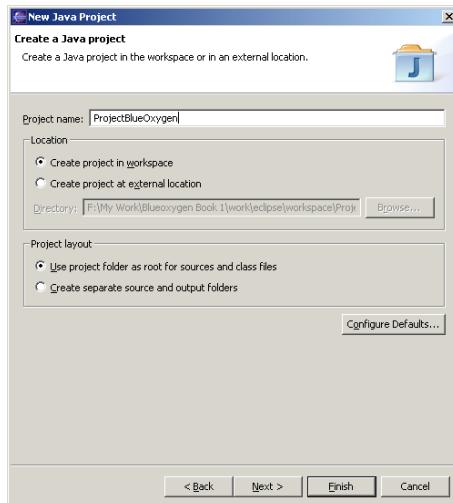
New Project

Setelah menu Project dipilih, maka secara otomatis sebuah dialog New Project akan muncul. Ada beberapa pilihan didalamnya, semuanya disebut Wizard. Setiap Wizard memiliki fungsi masing-masing, untuk memulai project Java, pilihlah Java Project dengan icon berhuruf J.



New Project Wizard

Setelah itu tekanlah tombol Next, untuk masuk kedalam Java Project Wizard.



Java Repository Dialog

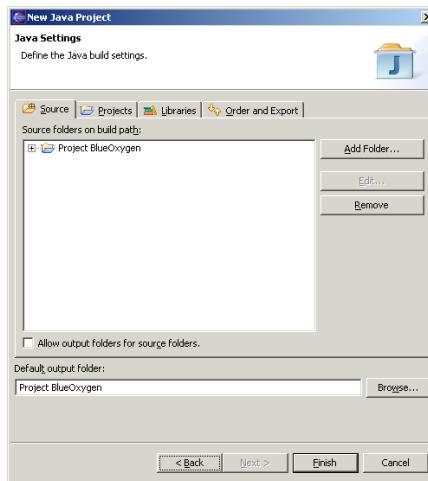
Setelah itu akan muncul sebuah dialog yang menanyakan nama

project yang akan dibuat, isilah dengan nama project yang diinginkan, spasi diperbolehkan.

Coba perhatikan isian kedua, disana tertulis, Create project in workspace, artinya project Java ini akan disimpan di repository Java, ingat saat pertama kali menjalankan Eclipse, nah dilokasi tersebut project Java ini akan disimpan.

Sebenarnya repository project java tersebut dapat dilihat di isian Directory, tetapi terkadang karena foldernya terlalu dalam struktur direktorinya, maka kita tidak tahu dimana project itu disimpan, sehingga lokasi workspace yang diminta saat Eclipse pertama kali berjalan adalah titik awalnya. Project repository ini terkadang diperlukan saat kita hendak membackup project atau mendistribusikan source code.

Setelah itu dapat menekan Enter untuk langsung bekerja, tetapi sebaiknya tekanlah tombol Next untuk masuk ke build setting. Build setting ini sebaiknya dibiasakan disetting dahulu, walaupun sifatnya hanya tambahan, ini untuk membuat kita bekerja lebih terorganisir.



Setting Source - Binary

Ada satu kelebihan Eclipse yang sangat berguna yaitu, kemampuan untuk memisahkan project Java dengan hasil kompilasi. Hasil kompilasi yang terpisah akan berguna sekali bila hendak membuat distribusi Java, karena dalam deployment, source code tidak diperlukan.

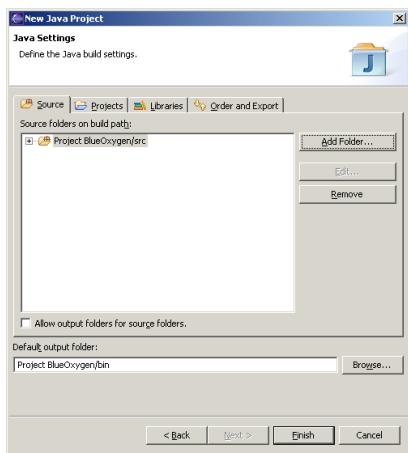
Caranya adalah dengan menambah folder “src”, yaitu dengan menclik tombol Add Folder. Isilah dengan “src” dan tekan ok.



Membuat Folder untuk Binary

Secara otomatis, Eclipse akan membuat folder “bin”, yang berarti binary. Folder bin ini adalah tempat hasil kompilasi Java disimpan. Coba perhatikan, icon pada Source folders on build

path akan berubah.



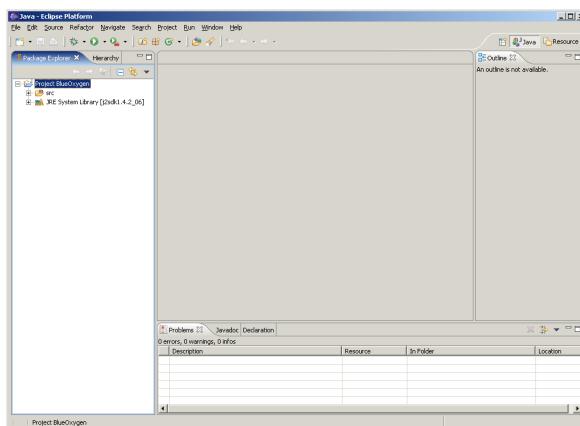
Source dan Binary yang terpisah

Memungkinkan dalam sebuah project Java ini memiliki lebih dari satu folder source.

Setelah selesai, tekan tombol Finish. Selamat, proses membuat project Java didalam Eclipse telah selesai.

Sebuah pertanyaan akan muncul, yang isinya kurang lebih adalah menanyakan apakah akan masuk ke modus Java untuk memulai project. Pilihlah Yes, artinya kita akan masuk modus pemograman Java.

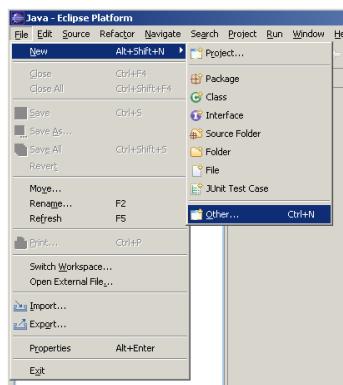
Eclipse akan mengaktifkan Java perspectivenya, dan ini artinya pemograman telah selesai.



Eclipse dengan Project Java dalam modus perspektif Java

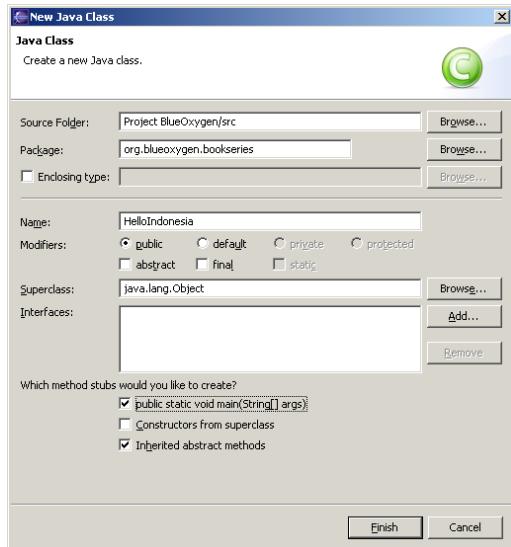
Membuat Aplikasi Java Pertama.

Sekarang kita hendak membuat sebuah aplikasi Java didalam project yang telah berhasil dibuat. Caranya adalah dengan memilih New pada menu File, coba perhatikan, sekarang sub menu New bertambah, ada beberapa pilihan tambahan. Dimenu tersebut Package, Class, dan Interface, yang mana ini berarti kita dapat membuat package atau folder dalam project yang telah dibuat, membuat sebuah Class baru atau membuat Interface.



Sub Menu New

Saat ini kita akan membuat sebuah aplikasi Java sederhana, untuk membuatnya pilihlah Class. Sebuah dialog New Java Class akan muncul, terdapat beberapa isian untuk mengisi parameter awal sebuah class Java.

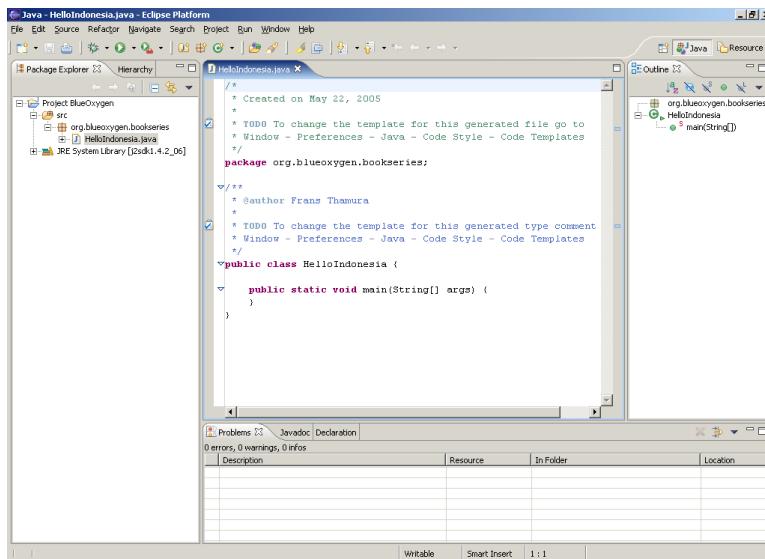


Membuat Class baru

Isilah package untuk mempermudah pengelompokan aplikasi Java, package sangat diperlukan untuk pemograman Java yang sesungguhnya, penamaan package akan sangat mempengaruhi kecepatan kita mengembangkan aplikasi terutama saat melakukan debugging.

Setelah itu isilah Name dengan nama aplikasi yang diinginkan, misalnya diisi HelloIndonesia, maka akan dibentuk file HelloIndonesia.java difolder “src”.

Pilihlah “public static void main(String[] args)” dalam pilihan “Which method stubs would you like to create?”, pilihan ini akan membuat file yang dibentuk adalah berbentuk aplikasi siap ekskusi, bilamana pilihan ini tidak dipilih, hasil file yang dibentuk oleh dialog New Java Class ini adalah sebuah Class Java yang tidak siap eksekusi.

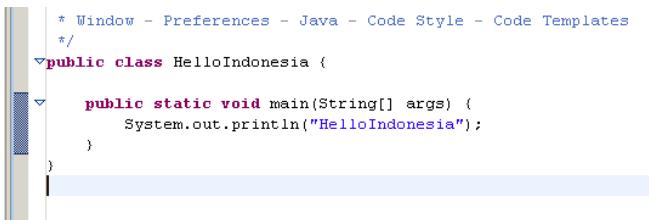


Workbench dengan Class Baru

Setelah tombol Finish ditekan, secara otomatis, file yang terbentuk yaitu HelloIndonesia.java akan langsung terbuka didalam editor Java.

Tambahkan satu baris diantara “public static void main(String[] args) {“ dan “}” dengan perintah dibawah ini:

```
System.out.println("HelloIndonesia");
```



```
* Window - Preferences - Java - Code Style - Code Templates
*/
public class HelloIndonesia {
    public static void main(String[] args) {
        System.out.println("HelloIndonesia");
    }
}
```

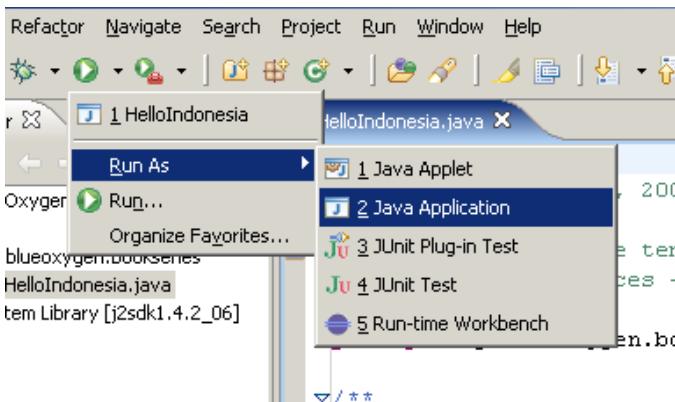
Java Editor

Setelah itu simpanlah file tersebut.



Icon Save

Coba tekan tombol “Run” dan kemudian pilih Run As, lalu pilih Java Application.



Menjalankan Java dengan modul Application

Karena saat pertama telah dipilih “public static void main(String[] args)” saat membuat file Java, maka file tersebut dapat dijalankan dalam modus Java Application.

Setelah pilihan Java Application dipilih, secara otomatis, file HelloIndonesia.java akan eksekusi. Sebuah baris akan muncul dibawah editornya, yaitu didalam Console View.



Output dari eksekusi

Ini artinya aplikasi Java pertama kita telah berhasil dibuat. Selamat!

Mendebug Aplikasi Java

Tambahkan baris dibawah System.out.println yang telah kita ketik dengan beberapa baris seperti dibawah ini.

```
int i = 1000;  
System.out.println("Isian i"+i);  
  
i = 1200;  
System.out.println("Isian i"+i);  
  
i = i *25;  
System.out.println("Isian i"+i);
```

Setelah itu eksekusikan kembali aplikasi HelloIndonesia diatas.

Console View akan menampilkan hasil dari dari eksekusi aplikasi HelloIndonesia diatas yaitu seperti dibawah ini:

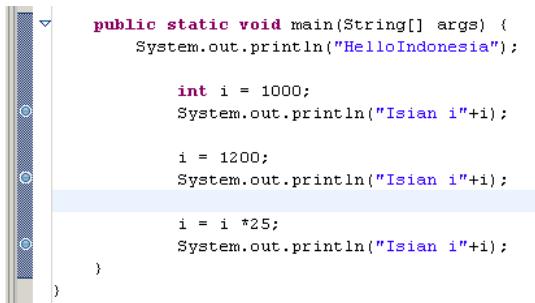
```
HelloIndonesia  
Isian i1000  
Isian i1200  
Isian i30000
```

Setelah itu klik kanan didepan baris setiap System.out.println, dan pilihkan Toggle Breakpoint,



Breakpoint

Lakukan untuk 3 baris yang terdapat System.out.println, sehingga didepan setiap baris System.out.println terdapat tanda bullet atau lingkaran kecil.



```
public static void main(String[] args) {
    System.out.println("HelloIndonesia");

    int i = 1000;
    System.out.println("Isian i"+i);

    i = 1200;
    System.out.println("Isian i"+i);

    i = i *25;
    System.out.println("Isian i"+i);
}
```

Breakpoint pada Source Code

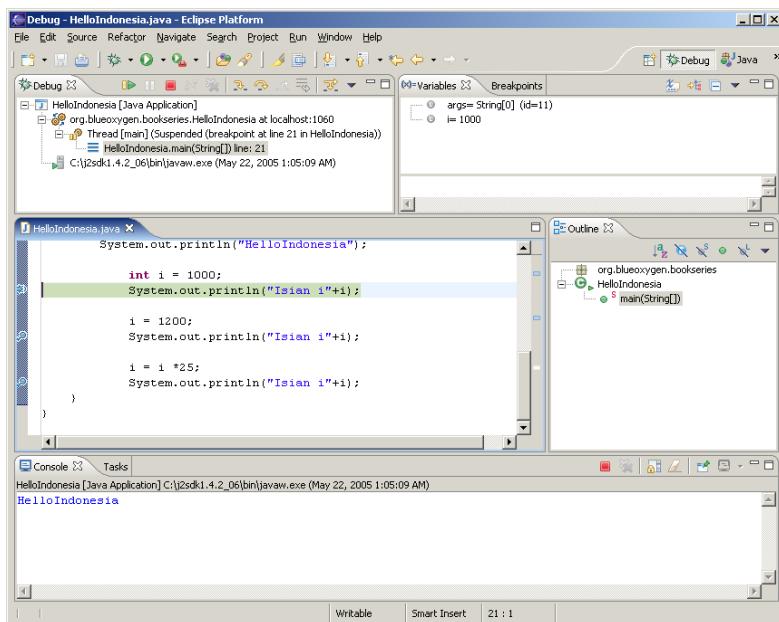
Pindahkan mouse ke tombol debug yang berbentuk “kutu”, kemudian pilihlah HelloIndonesia.



Mengeksekusi dalam modus debug

Secara otomatis, Eclipse akan masuk kedalam modus debug.

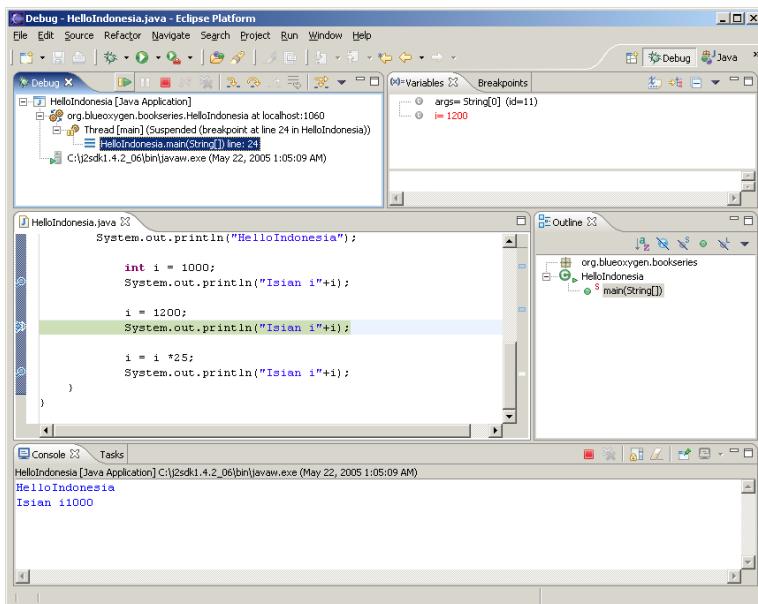
Cara Cepat Mengembangkan Aplikasi Java dengan Metode MVC



Eclipse dalam Perspective Debug

Perhatikan 3 bullet yang telah dibentuk, semuanya iconnya terdapat tanda centang, dan coba ke view paling kanan atas, disana terlihat nilai variable i adalah 1000.

Setelah itu coba tekan tombol resume (), maka secara otomatis Eclipse akan memindahkan baris ke bullet berikutnya.



Breakpoint dalam modus debug

Coba perhatikan Variable View yang berada di kanan atas, variable i berubah dari 1000 menjadi 1200.

Klik lagi icon resume di toolbar, dan lihat variable I berubah menjadi 30000.

Kegiatan ini disebut mendebug, artinya menganalisa baris perbaris yang disesuaikan dengan letak bullet breakpoint. Bisa dibayangkan kalau kode yang diketik telah lebih dari 1000 baris, dan perhitungan matematikanya telah sedemikian kompleksnya.

Nah ini adalah fitur paling produktif dari pemrograman. Dengan modus debugging ini, setiap programmer dapat menganalisis programnya dan juga dapat melakukan pemeriksaan baris

perbaris setiap kode yang ditulisnya.

Pada bab berikutnya akan dijelaskan melakukan debuggin pada aplikasi Java yang lebih kompleks, yaitu remote debugging.

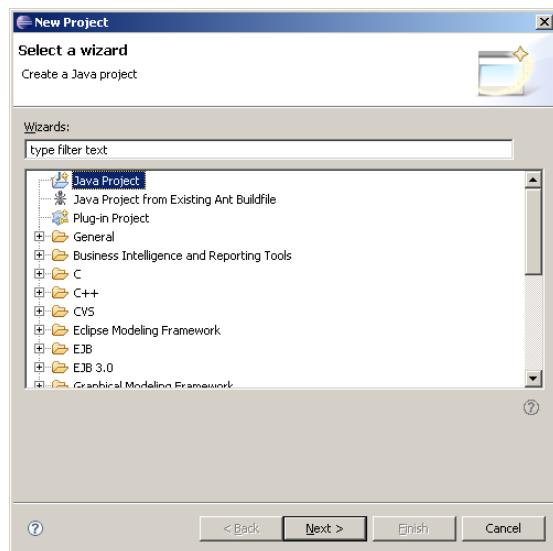
Memulai Projek WebWork (Metode 1) menggunakan Eclipse IDE standar.

Setelah memahami bagaimana WebWork bekerja, mati kita memulai menjalankan semua kode diatas menggunakan Eclipse IDE, saran penulis sebaiknya menggunakan Eclipse Calisto, karena telah terdapat modul editor untuk JSP dan koneksi ke server Java EE.

Untuk memulainya, extractlah file BlueWork.war. Kemudian jalankan Eclipse, buatlah sebuah Java Project, ini adalah step yang paling mudah, bilamana telah lebih mahir boleh mengikuti dengan bentuk projek bernama Dynamic Project, yang didesain khusus untuk solusi Web. Yang mana akan dijelaskan setelah ini.

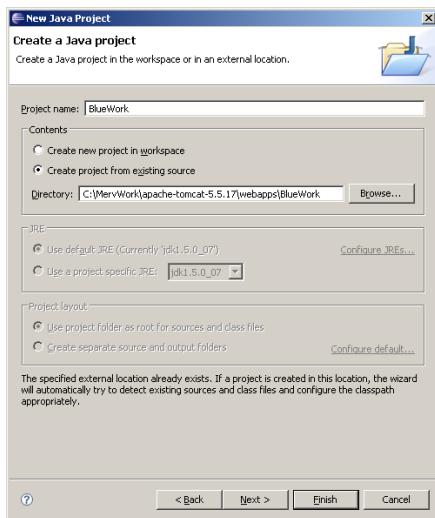
Extractlah file BlueWork.war ke folder /webapps dari tomcat, sehingga kita dapat melakukan selain debug, juga menjalankan aplikasi tersebut, atau dapat juga dengan menjalankan tomcatnya, kemudian mematikan tomcatnya setelah selesai, tomcat akan mengextract file BlueWork.war secara otomatis.

Penulis mengextract tomcat ke folder c:\MervWork, sedangkan BlueWorknya diextract didalam folder C:\MervWork\apache-tomcat-5.5.17\webapps\BlueWork



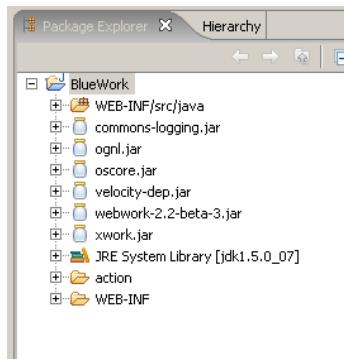
Membuat Project Baru

Setelah itu tekanlan tombol Next, dan isilah Project name dengan BlueWork, sedangkan Directory diisi dengan folder dimana BlueWork diextract, dalam kasus ini adalah C:\MervWork\apache-tomcat-5.5.17\webapps\BlueWork.



Folder BlueWork

Tekanlah tombol Finish, Next tidak diperlukan, karena BlueWork sudah dioptimize untuk pengembangan didalam lingkungan Eclipse. Otomatis Package Explorer Eclipse akan muncul struktur projek.



Package Explorer

Folder template atau presentation ada didalam folder action, sedangkan file xwork.xml ada didalam WEB-INF/src/config. Sedangkan semua kode Action, ada didalam folder WEB-INF/src/java.

Metode ini adalah pendekatan pengembangan aplikasi dengan mengikuti standar Java EE untuk spesifikasi web atau servlet.

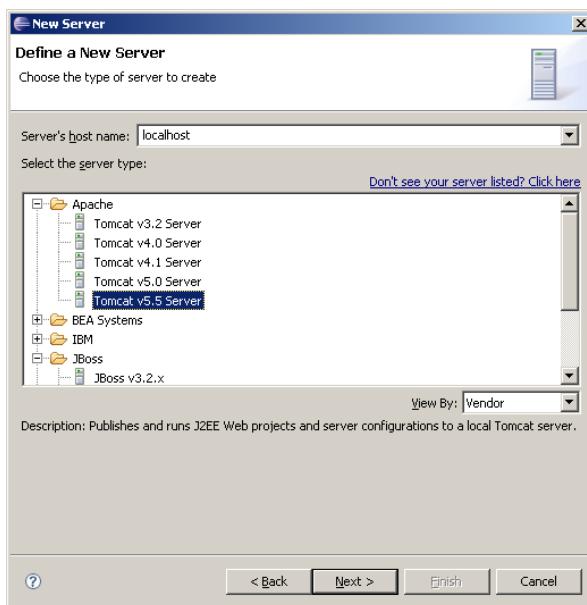
Untuk menjalankannya dapat menginstall Tomcat plugins dari sysdeo atau mengeksekusi startup.bat pada folder bin.

Semula penulis menggunakan metode ini, sehingga dapat berpindah dari IDE satu ke IDE lain, seperti dari eclipse keNetbeans atau sebaliknya.

Memulai Projek WebWork (Metode 2) menggunakan Eclipse Calisto

saat ini tim Eclipse telah mengeluarkan sebuah metode untuk menjalankan mekanisme yang memungkinkan melakukan debug pada container Web. Mekanisme ini disebut Dynamic Web Project, yang mana strukturnya agak berbeda.

Untuk memulainya, buatlah Server project, point projek ini ke folder dimana tomcat diinstall.

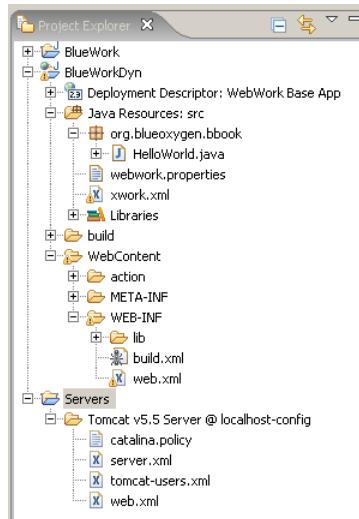


Setting Server dengan Tomcat

Eclipse mendukung banyak Java EE server, seperti Tomcat, Weblogic, Websphere, JBoss, Oracle OC4J, dan Jonas. Sayang sekali Sun Java atau Glassfish tidak didukungnya. Untuk yang hendak menggunakan Apache Geronimo, diperlukan plugins Geronimo, tetapi Calisto secara default seharusnya karena menggunakan WTP 1.5, harusnya sudah mendukungnya.

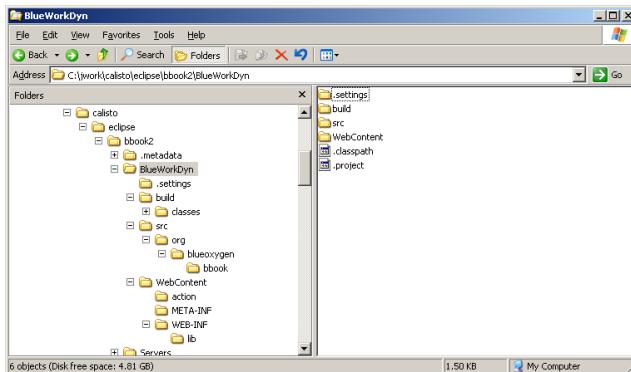
Setelah itu extractlah BlueWorkDyn.zip ke folder yang diinginkan, misalnya C:\MervWork, buatlah sebuah Dynamic Web Project terus pointlah ke folder dimana BlueWorkDyn bekerja.

Secara otomatis, sebuah projek akan muncul di Package Explorernya Eclipse.



Struktur Project dengan modus Dynamic Web

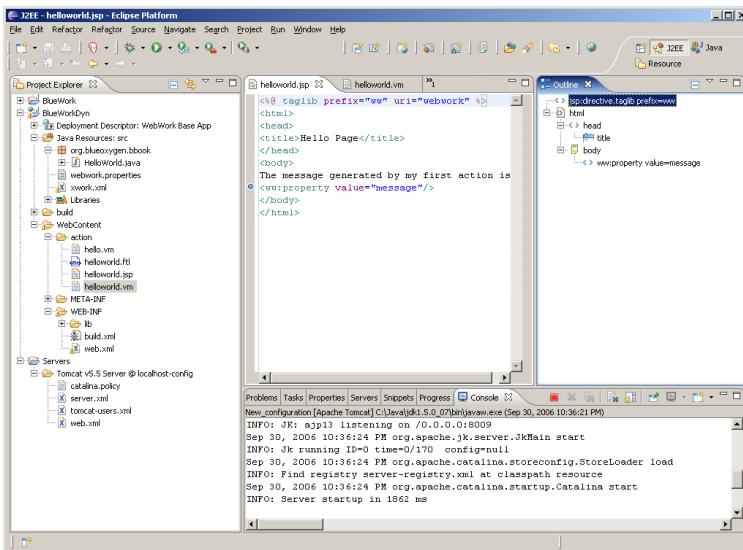
Bilamana kita perhatikan, struktur pengembangan aplikasi berbasis Web di Eclipse Calisto telah berubah. Lihatlah struktur dibawah ini. Source code dan hasil kompilasi dipisah.



Struktur Folder

Perbedaan metode 1 dan metode 2, adalah pada file action JSP,

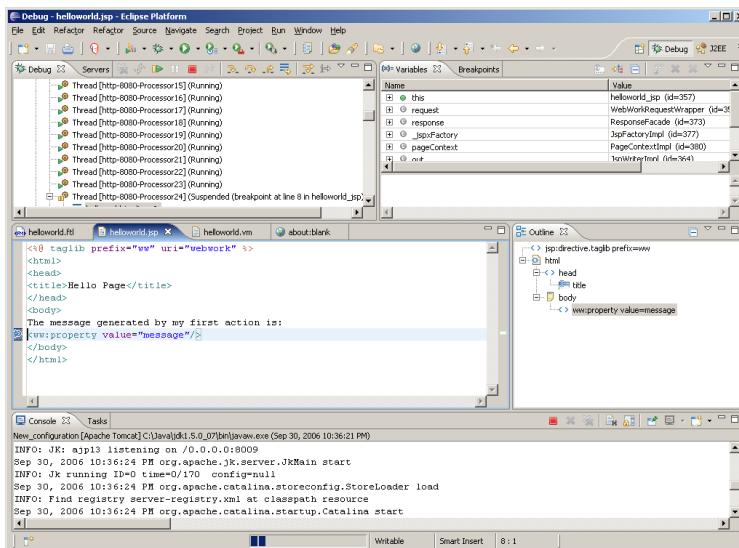
dengan menggunakan Dynamic Project, file jsp dapat didebug.



Workbench dengan hasil eksekusi

Bilamana server telah berjalan dengan baik, umumnya di Console akan ada tulisan INFO: Server startup in 1052 ms, ini artinya tomcat telah berjalan, plus project BlueWorkDyn didalamnya.

Coba jalankan <http://localhost:8080/BlueWorkDyn/action/helloJSP.action>.



JSP dalam modus debug

Cobalah dengan menjalankan aplikasi yang menggunakan Velocity dan Freemarker sebagai presentation layernya, Eclipse tidak dapat mendebugnya.

URL untuk Velocity adalah `http://localhost:8080/action/helloVelocity.action`, sedangkan untuk Freemarker adalah `http://localhost:8080/action/helloFreemarker.action`