PERANCANGAN PEMBUATAN PERANGKAT LUNAK WORKFLOW DESIGNER BERBASIS J2EE

TUGAS AKHIR

Disusun Oleh:

LINTANG JATI PRASOJO

NRP. 5100 100 081

JURUSAN TEKNIK INFORMATIKA
FAKULTAS TEKNOLOGI INFORMASI
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA

2004

PERANCANGAN DAN PEMBUATAN PERANGKAT LUNAK WORKFLOW DESIGNER BERBASIS J2EE

TUGAS AKHIR

Diajukan Guna Memenuhi Sebagian Persyaratan

Untuk Memperoleh Gelar Sarjana Komputer

Pada

Jurusan Teknik Informatika

Fakultas Teknologi Informasi

Institut Teknologi Sepuluh Nopember

Surabaya

Mengetahui/Menyetujui,

Dosen Pembimbing I	Dosen Pembimbing II
(Yudhi Purwananto, S.Kom, M.Kom)	(Faizal Johan A, S.Kom)
NIP. 132 172 210	NIP. 132 300 414

SURABAYA

JULI, 2004

satu waktu

pernah kita
sama-sama berdiri menentang angin
dan menatap garang wajah sang surya

pernah kita
sama-sama berlari menyongsong waktu
dan sebutir harapan
yang kadang enggan
tuk tampakkan senyumnya

sering kita
bersama merenungi arti hidup
yang selalu berputar dan bergulir
di antara seribu satu deru
jengah nafas kehidupan

seiring kaki-kaki kecil yang terus terayun
seiring waktu yang tak kenal lelah berdetak
selama itu pula kita akan terus berpacu
dengan derasnya laju roda kehidupan

sampai pada satu waktu
di mana semua menampakkan ujud aslinya

sampai pada satu kesempatan
di mana semua yang pernah merasakan
pahit getirnya hidup akan tersenyum puas

sampai pada satu waktu
di mana semua yang pernah tersenyum
akan merasakan sesaknya tangis

sampai pada saat di mana
sang waktu menelanjangi setiap inchi tubuh kita
memaksa kita untuk tertunduk malu
dan meratap
menyesali semua yang pernah kita lakukan
tanpa pernah ada alasan
untuk kembali
dan menulis ulang catatan sejarah
yang terlanjur kita torehkan

sampai pada satu waktu
di mana semua rahasia tersingkap
dan kita semua akan berdiri
terpaku
di balik tirai keabadian

ABSTRAK

Abstrak

Dalam suatu kegiatan dikenal istilah proses, dimana suatu proses terdiri dari sejumlah pekerjaan / aktivitas (task). Misalkan dalam proses pembelian barang terdapat serangkaian pekerjaan antara lain pemeriksaan jumlah persediaan barang, pemberharuan data pada basis data, persetujuan transaksi, dan pengiriman e-mail. Sebuah proses bisa berupa serangkaian pekerjaan yang sederhana atau serangkaian pekerjaan yang kompleks. Dalam suatu struktur organisasi yang mapan, rangkaian proses ini diterjemahkan dalam berbagai aplikasi, salah satunya adalah dokumen workflow.

Pada tugas akhir ini dikembangkan sebuah perangkat lunak workflow designer yang dapat digunakan untuk mendefinisikan dan merancang tata aliran penyusunan dokumen sesuai dengan alur bisnis pada setiap lembaga yang menggunakan aplikasi ini. Dari workflow modeller ini pula nantinya dihasilkan sebuah aplikasi berbasis web menggunakan teknologi J2EE, dalam hal ini JSP, Servlet, dan mengimplementasikan Struts FrameWork dan OR-Mapping menggunakan Hibernate.

Berdasar uji coba yang dilakukan, terbukti perangkat lunak dapat bekerja dengan baik sesuai fungsi yang diharapkan. Pembuatan alur bisnis dapat dilakukan dengan baik untuk bermacam-macam kasus alur dalam skenario ujicoba, dan kemudian dihasilkan pula sebuah aplikasi yang sesuai dengan alur bisnis yang telah dimodelkan tersebut.

KATA PENGANTAR

KATA PENGANTAR

Alhamdulillaahi rabbil 'alamin, segala puji bagi Allah Yang Maha Kuasa yang telah memberikan kekuatan-Nya sehingga penulis bisa menyelesaikan tugas akhir yang berjudul : **"PERANCANGAN DAN PEMBUATAN PERANGKAT LUNAK WORKFLOW DESIGNER BERBASIS J2EE."**

Melalui tugas akhir ini, penulis merasa mendapat pelajaran dan pengalaman baru, karena hampir semua teknologi yang dipelajari dan dikembangkan dalam tugas akhir ini tidak penulis dapatkan di perkuliahan biasa. Penulis juga menyadari bahwa tugas akhir ini masing memiliki banyak kelemahan dan kekurangan. Oleh karena itu penulis sangat mengharapkan kritik dan sarannya bagi pengembangan ke depannya.

Akhirul kalam, penulis mengucapkan terima kasih kepada semua pihak yang telah membantu penyelesaian tugas akhir ini. Semoga Allah membalasnya dengan balasan sebaik-baiknya. Amien.

Surabaya, Juli 2004

Penulis

UCAPAN TERIMA KASIH

Alhamdulillah, puji syukur kehadirat Allah S*ubhanahu Wa Ta'ala* yang senantiasa mencurahkan karunia dan petunjuk-Nya. Tiada daya dan kekuatan melainkan dengan izin-Nya.

Secara khusus, dalam kesempatan ini pula, penulis ingin mengucapkan terima kasih dan penghargaan sebesar-besarnya kepada semua pihak yang telah menyertai dan membantu penulis selama ini, antara lain kepada:

1. Bapak Sunarto, Ibu Retno Puji Astuti dan Mbah Sumarmi selaku orang tua dan nenek penulis. Tanpa cinta dan kasih sayang mereka, penulis tidak akan pernah terlahir ke dunia. Dengan kasih sayang mereka, bimbingan mereka, kesabaran dan suri tauladan yang mereka tunjukkan. Sungguh, semuanya tidak akan bisa untuk membalas jasa-jasa mereka. Setidaknya ini adalah persembahan kecil, yang bisa ananda berikan. Ya Allah bimbinglah mereka, dan sayangi mereka sebagaimana mereka merawatku dengan kasih sayang selama ini.
2. Bapak Yudhi Purwananto dan Bapak Faizal Johan selaku dosen pembimbing penulis, atas segala bantuan dan kesabarannya membimbing penulis selama pengerjaan tugas akhir ini. Mohon maaf bila ada kesalahan-kesalahan, maupun perilaku tidak sopan yang pernah penulis lakukan. Semoga barokallah profesinya, dan amal baiknya dicatat oleh Allah sebagai kebaikan yang bermanfaat di akhirat nanti.

3. Bapak Darlis Heru Murti, Ibu Siti Rochimah, dan Bapak Fajar Baskoro, selaku dosen penguji penulis yang telah sempat membuat penulis kelabakan dalam menjawab pertanyaan-pertanyaan yang diajukan. Terimakasih pula atas kemudahan yang telah diberikan sehingga penulis berkesempatan untuk lulus di semester ini.
4. Retno Aulia Vinarti a.k.a *Pipot*, selaku adik kandung penulis. Jangan nakal sama Bapak dan Ibu ya ? Belajar yang rajin, jangan lupa berdoa, biar nanti dipilihkan Allah jurusan yang terbaik.
5. Segenap keluarga dan kerabat tercinta, di Surabaya (Bude Sum, Mbak Titin, Mbak Peni, Mas Eko, Mas Umar), di Mojokerto (Pakde dan Bude Wadji, mas Iim, mas Eer, mbak Rika), di Jombang (Bude Warni, Mbak Betty), di Madiun (Bude Is, Mbak Dyah, mas Dodik), serta kerabat lainnya yang tidak bisa disebutkan satu persatu saking banyaknya.
6. Segenap staff dosen Teknik Informatika ITS, atas bimbingan dan curahan ilmunya sehingga penulis mendapatkan tambahan wawasan baru dalam dunia informatika, pak Darlis dengan tugas Oraclenya, pak Royyana – atas kesediannya jadi *suhu* Linux penulis, bu Nanik Suciati selaku dosen wali pertama penulis ketika mahasiswa baru – *jadi ingat ibu di rumah* ☺, pak Tohari atas pemaklumannya karena penulis sering tidur di ruangan beliau ☺, pak Agus Zainal atas ilmu baca Al-Qur'an dan tafsirnya, Bu Siti Rochimah, pak Suhadi dan pak Dwi Sunaryono – atas tugas PSI nya yang berat ☺, bu Anny Yuniarti atas diskusinya yang bermanfaat, pak Husni dan pak Arif Djunaidy – mohon maaf apabila dalam acara pengkaderan penulis sering

membuat kesal ☺, pak Imam Kuswardayan – atas ilmu-ilmu berharganya. Teringat masa-masa mengerjakan tugas di Lab. Ternyata semua itu berguna. Terimakasih sekali lagi.

7. Segenap staff Tata Usaha dan Ruang Baca Teknik Informatika ITS, Mas Yudhi – atas kromprominya sehingga penulis kala itu bisa ikut ujian salah satu mata kuliah, pak Sholeh dan pak Jarwo – atas tendangan bolanya waktu TC Cup, pak Karmono dan pak Bagyo - assalamu'alaikum pak ☺ , pak Supri dan pak Ismanu – atas obrolannya yang penuh nasihat kehidupan, mas Gayuh – atas kerjasamanya di LabProg selama ini, mas Hermono – atas sparing partner Linuxnya, mas Dwi AJK - *tibake wong Mojokerto toh*, dan staff lainnya yang tidak bisa disebutkan satu persatu. Terimakasih !
8. Teman-teman noceng, Gunna, Joko, Jakka, Kamal, Haikal, mbah Younan, Bawenang, Hendra, Penthol, Gunawan Unair, Rudi, Ferdian, Deka, Hoirul, Ruli, Shofie, Putri, Ika, Dyah Lintang, Iin, Yunita, dan banyak lagi yang belum tersebut, terimakasih atas persahabatannya selama ini.
9. Teman-teman 98, mas Riz dan mas Aby – yang juga suhu Linux penulis, mas Cphee, mas Kent, mas Dimas, mas Yudha, mas Deni, mas Budi – yang mau menampung penulis selama satu semester ini di *kantor* mereka, mas Danang, mas Bomphie, mas Kakek ☺, mbak Ririn – udah dapet belum yang dicari mbak ? ☺, dan semua yang telah berpartisipasi dalam hidup penulis, terimakasih.
10. Teman-teman 99, macan putih – *ini TA, bukan retorika* ☺, mas CDQ, mas Heru, mas Warna Agung, mas Kemal[*Syah*] – sadar mas, ingat ya 5 tahun lagi

janji ente milih PKS ☺ , mbak Ningrum – atas nasihat-nasihatnya sebagai kakak, mbak Mimiek – *ndhang lulus mbak !*, bu Anny, mbak Aak, dan banyak lagi, terimakasih !

11. Teman-teman 2001, Galih, Hendra, Choy, Fafad, Eva, Rohmah, Rahmat, Salman the Terorist, Izam, Ria, Siti, Dwi, Phe, dan masih banyak lagi, terimakasih !
12. Teman-teman 2002, Taqiyudin, Zhain, Aris, Tama si adik kecil, Henry, Iwan, Huda, Wawan, Hendra Fauzan, Kenip, Hendra Ext, Bima Ext, Ihwana, Feby, Catur, Ratna, Henning, Antie, semuanya terimakasih, doakan kakakmu ini sukses selalu ya ☺.
13. Teman-teman 2003, Rile, Myrizky, Hendra, Fe, Wilud, dan semuanya, mohon maaf atas perilaku yang mungkin kurang berkenan semasa pengkaderan ☺.
14. Teman-teman LabProg, mulai generasi pertama sampai ketiga, Kris, Zainal, Kholimi, Joko, Jakka, Gunna, Iin, Eva, Ratna, Hudew, Henry, Fauzan, Tama, Wawan, Iwan – jangan chatting aja ☺, bang Adjoun *the king of Empire*, semuanya, jangan pernah berhenti mengoprek selama kalian masih di TC, rugi ! Alternatifnya hanya Linux, OpenSource dan Java, buang jauh-jauh oprek Microsoft dan kapitalisme ☺.
15. Teman-teman SI, Macan Putih, Fauzan asli, mas Agung, Mumud, Topan, CDQ, Angga – boleh nih diadu sama kru generasi ketiga LabProg ☺.
16. Teman-teman AJK, bang Yoyok, Prefix, Fadelix, Sinchanix, Dwi Arix, Alifix, Victorix – teman setia praktikum Basdatix ☺, *ayo rek, ojok suwe-suwe, ndhang lulus*.

17. Teman-teman YM ku yang setia siang dan malam menemani pengerjaan tugas akhir ini, a_hadiana, aridaistia, adhiaxa, bluejundi, digitallyduck, fariningtyas, ffox99, java_sigit – atas *intro to servlet* untuk pertama kali, ronnie_grezik, to2mail, truetuti, nabila – *terimakasih buat kenangan SD nya*, monang_ok – *suhu jarak jauh java gue nih*, uwie_dr, davina_kj, netcompetency dan bwidyasanyata – *atas semangatnya untuk segera lulus*, untungrohwadi, vietha_30, yokka, zaydam_25 dan banyak lagi, terimakasih atas diskusi-diskusinya yang menarik dan bermanfaat.
18. Team sukses ku, mas Kent yang dengan sabar menjawab semua pertanyaan penulis dengan kasih sayang ☺, Ruli dan Shofie yang dengan sabar mengedit buku TA sehingga bisa dibentuk daftar isi yang benar ☺ . Semoga kalian selalu diberi kemudahan dalam segala urusan.
19. Teman-teman SITC, BEM, HMTC , JMMI dan LDK dimanapun antum semua berada, semoga selalu dalam petunjuk dan lindungan Allah SWT.
20. Teman-teman sepenafkahan di PJB, bang Kurnix – *kapan aku dapet keponakan nih*, mbak Fani, Ruli, Pak Djoni, mbak Maya – *thanks, that means a lot* ☺, mas Suhariyono, pak Kasbullah, pak Ali, pak Us, Erick, Adi, terimakasih semua !
21. Teman-teman di Xpertize, Ika, Yunita, Kamal, Ferdi, Joko, Jakka, Dimas, Iin, dan Lex, semoga usaha kita barokah.
22. Semua yang telah berpartisipasi dalam hidup penulis, baik sebagai tokoh utama, figuran, maupun tokoh pembantu. Pertemuan dengan kalian membuatku lebih bisa menemukan arti hidup sesungguhnya, terimakasih!

DAFTAR ISI

DAFTAR ISI

DAFTAR TABEL

BAB I
PENDAHULUAN

BAB I

PENDAHULUAN

1.1 Latar Belakang

Dalam suatu kegiatan dikenal istilah proses, dimana suatu proses terdiri dari sejumlah pekerjaan / aktivitas (*task*). Misalkan dalam proses pembelian barang terdapat serangkaian pekerjaan antara lain pemeriksaan jumlah persediaan barang, pemberharuan data pada basis data, persetujuan transaksi, dan pengiriman e-mail. Sebuah proses bisa berupa serangkaian pekerjaan yang sederhana atau serangkaian pekerjaan yang kompleks.

Dalam suatu struktur organisasi yang mapan, dokument *workflow* juga turut berpengaruh dalam efisiensi kinerja organisasi tersebut. Dokument *workflow* yang jelas dan teratur, tentunya akan mempercepat pemrosesan dokumen tersebut sampai ke tujuan sebagaimana yang diharapkan.

Berbagai tuntutan tersebut diatas tentu saja cukup menyulitkan/menambah beban kerja analis sistem/programmer sebagai pengembang sistem karena banyaknya pekerjaan yang harus dilakukan.

Dari gambaran diatas maka dibutuhkan suatu aplikasi yang mampu membantu analis sistem/programer untuk mendefinisikan rangkaian pekerjaan beserta aturan-aturannya dalam suatu proses yang kemudian proses itu dapat

dieksekusi secara otomatis berdasar aturan / urutan rangkaian pekerjaan yang telah didefinisikan sebelumnya.

1.2 Permasalahan

Berdasarkan latar belakang yang telah diuraikan sebelumnya, permasalahan yang akan diselesaikan dalam tugas akhir ini adalah sebagai berikut:

- Bagaimana merancang dan membuat sebuah aplikasi *workflow designer* serta dapat memodelkan alur bisnis yang ada.
- Bagaimana menghasilkan output berupa aplikasi yang interaktif dan siap pakai.

1.3 Tujuan dan Manfaat Pembuatan Tugas Akhir

Tujuan pembuatan tugas akhir ini ialah untuk membuat perangkat lunak *workflow designer* yang dapat digunakan untuk mendefinisikan dan merancang tata aliran penyusunan dokumen sesuai dengan proses bisnis pada setiap lembaga yang menggunakan aplikasi ini.

1.4 Batasan Masalah

Dari permasalahan-permasalahan diatas, maka batasan masalah dalam tugas akhir ini ialah:

- Teknologi yang digunakan untuk aplikasi administrator adalah java applet.
- Teknologi yang digunakan sebagai *back-end* adalah teknologi J2EE dengan menggunakan *Struts Framework*

1.5 Metodologi Metodologi Pembuatan Tugas Akhir

Pembuatan tugas akhir ini terbagi menjadi beberapa tahapan pengerjaan, yaitu:

1. Studi Literatur dan Software

   Studi literatur dilakukan dilakukan untuk mempelajari konsep dari teknologi yang digunakan. Informasi diperoleh baik melalui buku maupun referensi dari internet.

2. Perancangan Sistem dan Aplikasi

   Untuk perancangan aplikasi dan database digunakan tool Rational Rose untuk merancang sebuah aplikasi yang object oriented based.

3. Pengembangan Aplikasi

   Pada tahap ini dilakukan pembuatan perangkat lunak dengan menggunakan konsep dan teknologi yang telah didapatkan melalui metodologi nomor satu diatas. Pengembangan dilakukan dengan menggunakan Java Programming Language diatas Tomcat WebServer.

4. Uji Coba dan Evaluasi

   Setelah tahap pembuatan aplikasi selesai, maka dilanjutkan dengan melakukan simulasi kasus yang sesuai dengan proses bisnis lembaga pemerintahan yang bersangkutan.

5. Penyusunan Buku Tugas Akhir

Pada tahap terakhir ini akan disusun buku sebagai dokumentasi dari pelaksanaan Tugas Akhir. Dokumen ini juga diharapkan dapat berguna bagi orang lain yang memiliki keinginan untuk mengembangkan sistem tersebut lebih lanjut.

1.6 Sistematika Penulisan Tugas Akhir

Sebagai gambaran umum dari laporan Tugas Akhir yang disusun, berikut ini akan dijelaskan tentang sistematika penyusunan laporan Tugas Akhir ini secara garis besar.

BAB I PENDAHULUAN

Bab ini membahas tentang latar belakang pembuatan sistem, permasalahan yang dihadapi, tujuan dan manfaat dari Tugas Akhir ini, ruang lingkup, batasan-batasan dan metodologi yang dipergunakan.

BAB II TEKNOLOGI J2EE DAN WORKFLOW MANAGEMENT SYSTEM

Bab ini membahas tentang teori-teori penunjang yang dipergunakan dalam penyelesaian Tugas Akhir. Teori penunjang ini mencakup penjelasan tentang teknologi J2EE yaitu diantaranya EJB, JSP, JNDI, serta *Struts Application Framework* dengan mengimplementasikan MVC (Model-View Controller). Dalam bab ini juga akan dibahas teori penunjang tentang *workflow modeling*.

BAB III PERANCANGAN PERANGKAT LUNAK

Bab ini membahas tahapan-tahapan dari proses perancangan perangkat lunak. Bagian ini terdiri dari perancangan data dan perancangan proses bisnis serta fungsi-fungsi yang dipergunakan dalam perancangan dan pembuatan perangkat lunak.

BAB IV IMPLEMENTASI PERANGKAT LUNAK

Bab ini membahas tentang proses pembuatan perangkat lunak sesuai dengan perencanaan pada bab sebelumnya.

BAB V UJI COBA PERANGKAT LUNAK

Bab ini menjelaskan proses uji coba dan evaluasi perangkat lunak yang telah dikembangkan dengan mempergunakan data yang telah disiapkan.

BAB VI PENUTUP

Bab ini berisi kesimpulan dan saran bagi pihak-pihak yang akan mengembangkan aplikasi ini lebih lanjut kedepannya.

BAB II
TEKNOLOGI J2EE DAN WORKFLOW MANAGEMENT SYSTEM

BAB II

TEKNOLOGI J2EE DAN WORKFLOW MANAGEMENT SYSTEM

Bab ini akan membahas tentang teknologi yang akan dipergunakan dalam pengembangan sistem, yaitu teknologi **Java 2 Enterprise Edition** (J2EE) beserta beberapa Java API penting lainnya yang dipergunakan. Aplikasi *workflow designer* ini akan mengimplementasikan juga sebuah framework desain untuk membangun aplikasi berbasis web, yaitu *Struts* dari Jakarta Apache Project.

2.1 Java 2 Platform, Enterprise Edition

Teknologi Java 2 Enterprise Edition (J2EE) diperkenalkan oleh perusahaan Sun Microsystems sebagai solusi pengembangan aplikasi berskala *enterprise*. Setelah sekian lama sejak diperkenalkan, teknologi ini telah berkembang menjadi model pengembangan aplikasi terdistribusi yang dianggap paling matang dan memiliki skalabilitas yang baik. Teknologi ini juga menawarkan berbagai fitur untuk pengembangan aplikasi yang besar, aman, transaksional, dan berskalabilitas tinggi. Teknologi ini mendukung pengembangan aplikasi yang berbasis komponen yang dapat memaksimalkan proses pengembangan dengan penggunaan ulang (*reuse*). Sebagai salah satu teknologi yang dikembangkan dari Java, teknologi ini juga dapat dijalankan pada berbagai macam server karena bersifat tidak tergantung pada *platform* perangkat keras maupun sistem operasi tertentu. J2EE menawarkan beberapa teknologi untuk pengembangan aplikasi, diantaranya Enterprise JavaBeans (EJB), Servlets dan Java Server Pages (JSP). Teknologi J2EE ini pada dasarnya

adalah spesifikasi standar saja, sedangkan implementasinya dapat dilakukan oleh pihak manapun, sehingga tidak terbatas pada server, *platform, middleware,* implementasi *application* server atau perusahaan tertentu.

2.1.1 Enterprise JavaBeans (EJB)

Enterprise JavaBeans (EJB) merupakan sebuah arsitektur model komponen terdistribusi yang disediakan oleh J2EE. Teknologi ini memberikan kemudahan untuk mengembangkan komponen-komponen logik bisnis aplikasi yang bersifat aman, transaksional, berskalabilitas tinggi, dan dapat dipergunakan oleh banyak pengguna sekaligus (*multi user*). Teknologi ini memungkinkan untuk memisahkan logika aplikasi dari layanan sistem, sehingga programmer dapat lebih berkonsentrasi pada proses bisnis, sedangkan layanan sistem seperti akses ke basis data akan ditangani oleh Enterprise JavaBeans.

Secara teknis, EJB berupa sekumpulan *class-class* java yang mengikuti aturan-aturan yang telah ditetapkan dalam standar EJB, menyediakan method pemanggilan (*finder method*) tertentu dan sebuah file konfigurasi berbentuk XML, dan dikombinasikan menjadi satu komponen.

Terdapat perbedaan antara Enterprise JavaBeans (EJB) dan JavaBeans, antara lain:

JavaBeans mendefinisikan suatu model untuk mengembangkan komponen multi guna yang biasanya dipergunakan di sisi *client*. Komponen tersebut dapat mempunyai antar muka pengguna (*user interface)*, dapat juga tidak.

Enterprise JavaBeans mendefinisikan model komponen pada sisi server (*server side*) untuk mengembangkan komponen dengan fungsi spesifik untuk logika

bisnis. Komponen EJB tidak menyertakan antar muka pengguna (*user interface*).

Enterprise JavaBeans terdiri dari 4 (empat) elemen penting, antara lain home interface, remote interface, bean implementation class, dan deployment descriptor.

2.1.1.1 Home Interface

Home interface bertanggung jawab untuk mengontrol operasi-operasi yang berkaitan dengan siklus hidup *bean,* misalnya untuk mencari, menghapus atau membuat *instance* bean baru. *Home interface* memungkinkan *client* EJB dapat melakukan hal-hal sebagai berikut :

- Membuat *instance* baru bari *bean*
- Menghapus instance bean
- Mencari *instance* entity bean yang sudah ada

2.1.1.2 Remote Interface atau Component Interface

Remote interface dari sebuah EJB mendefinisikan method-method bisnis EJB yang dapat diakses oleh client. Remote interface mendefinisikan beberapa method sehingga client dapat melakukan hal-hal sebagai berikut :

- Menghapus *instance enterprise bean*
- Membaca data yang tersimpan dalam suatu *instance EJB*
- Memperbarui (update) data yang tersimpan dalam suatu *instance EJB*
- Memeriksa apakah suatu *instance enterprise bean* sama dengan *instance EJB* lainnya.

- Mendapatkan nilai *primary key* dari suatu *instance EJB*

2.1.1.3 Bean implementation class

Class ini berisi semua implementasi aktual logika bisnis sebuah EJB. *Method-method* bisnis yang didefinisikan pada class enterprise bean akan secara otomatis dipanggil oleh container ketika client memanggil method terkait yang terdaftar dalam remote interface.

2.1.1.4 Deployment descriptor

Setelah membuat *home interface, remote interface,* dan *bean implementation class,* selanjutnya diperlukan sebuah *deployment descriptor* sebelum proses *deploy* komponen EJB ke *container* dapat dilaksanakan. *Deployment descriptor* adalah sebuah file XML (harus bernama *ejb-jar.xml*) yang berisi instruksi proses *deployment* dan daftar sumber daya yang diperlukan oleh komponen-komponen EJB untuk proses *deploy* tersebut. Beberapa isi *deployment descriptor* yang penting antara lain :

- *<ejb-jar>*

  Elemen ini merupakan elemen akar (*root*) dari sebuah *deployment descriptor* yang bersifat wajib. Semua elemen lain akan berada di dalam elemen ini.

- *<enterprise-beans>*

  Elemen *<enterprise-beans>* ini berisi deklarasi semua EJB yang akan didaftarkan untuk proses *deployment* ini.

- *<ejb-name>*

Elemen *<ejb-name>* menspesifikasikan sebuah nama dari enterprise bean. Nama ini harus unik di dalam satu file *ejb-jar.xml* yang sama. Pemrograman enterprise bean tidak tergantung pada namanya, karena itu *<ejb-name>* dapat diganti selama proses pengembangan tanpa menganggu fungsi-fungsi dari enterprise bean. Namun demikian, nama inilah yang akan dicari oleh *client* di JNDI sehingga apabila sudah ada aplikasi *client* yang menggunakannya, maka sebaiknya nama ini tidak diubah lagi.

- *<home>*

  Elemen *<home>* berisi nama lengkap dari java *class* home interface sebuah enterprise bean.

- *<remote>*

  Elemen *<remote>* berisi nama lengkap dari java *class* remote interface sebuah enterprise bean.

- *<ejb-class>*

  Elemen *<ejb-class>* berisi nama lengkap dari java *class implementation bean* sebuah enterprise bean.

- *<persistence-type>*

  Elemen *<persistence-type>* menspesifikasikan sebuah tipe pengaturan penyimpanan data entity bean. Terdapat dua tipe pengaturan penyimpanan, yaitu Bean dan Container, masing-masing untuk *Bean Managed Persistence* (BMP) dan *Container Managed Persistence*

(CMP). Sesuai dengan teknologi yang dipilih dalam tugas akhir, untuk selanjutnya maka nilai elemen <*persistence-type*> adalah "Container".

- <*primary-key-class*>

  Elemen <*primary-key-class*> berisi nama *class* lengkap dari tipe data primary key sebuah entity bean. Apabila *primary key* dari table bertipe *Integer*, maka <*primary-key-class*> akan dituliskan seperti contoh berikut ini:

  <prim-key-class>java.lang.Integer</prim-key-class>

- <*reentrant*>

  Elemen <*reentrant*> dapat bernilai false atau true. Jika <*reentrant*> bernilai false, maka sebuah entity bean tidak akan dapat dipanggil ulang selama proses bisnis entity bean tersebut berlangsung. Dan elemen ini sebaiknya di set false untuk setiap entity bean.

- <*abstract-schema-name*>

  Elemen ini berisi nama dari bean yang mengabstraksi table tertentu.

- <*cmp-field*> dan <*field-name*>

  Elemen <*cmp-field*> memberitahukan kepada Container, *field-field* apa yang diperlukan untuk mengatur penyimpanan sebagaimana telah ditentukan di elemen anak <*field-name*>. <*field-name*> haruslah sebuah variabel *public* dari class enterprise bean.

- <*primkey-field*>

  Elemen <*primkey-field*> digunakan untuk menentukan nama dari field primary key untuk sebuah entity dengan pengaturan penyimpanan

Container. <*primkey-field*> harus merupakan salah satu dari field yang dideklarasikan didalam elemen <*cmp-field*> dan tipe dari field harus sama dengan tipe dari primary key. Elemen <*primkey-field*> tidak digunakan jika primary key memetakan ke multiple field. Pada kasus seperti ini, field dari primary key haruslah public dan nama dari field-field tersebut haruslah sesuai dengan nama field dari class entity bean yang mengandung key tadi.

Gambar 2.1 menunjukkan bagan letak aplikasi J2EE yang menggunakan teknologi Enterprise Java Bean ini.

Gambar 2.1 Bagan aplikasi J2EE menggunakan teknologi EJB

Sesuai dengan spesifikasi teknologi EJB 2.0, terdapat 3 macam jenis Enterprise Beans yaitu session bean, entity bean, dan message driven bean.

a. Session Bean

Session bean digunakan untuk memodelkan proses bisnis yang ada. Session bean digunakan untuk merepresentasikan proses, kendali dan alur kerja. Session bean bersifat temporary dan tidak akan survive jika server mengalami crash. Session bean hanya diakses satu client, dan bersifat

transaksional. Session bean biasa digunakan untuk mengakses model database dalam entity bean. Session Bean diklasifikasikan menjadi 2 tipe yaitu :

1. Statefull Session Bean

   EJB ini biasanya digunakan untuk menyimpan business logic untuk single client. Dirancang untuk menyediakan layanan pada klien dengan memegang conversational-state. Statefull bean ini akan dicreate untuk masing-masing klien yang memanggilnya, dan kemudian akan didestroy jika klien membuangnya.

2. Stateless Session Bean

   EJB ini biasa digunakan untuk menyimpan service object. Dirancang untuk menyediakan layanan kepada klien tanpa memegang conversational-state. Stateless bean ini tersimpan dalam pool. Setiap kali klien memanggil method yang disediakan, maka sebuah instance dari bean ini akan bekerja untuk klien tersebut. Kemudian setelah selesai dijalankan, instance tersebut akan dikembalikan ke pool.

b. Entity Bean

Entity Bean merupakan salah satu jenis teknologi EJB yang disediakan oleh Java 2 Enterprise Edition (J2EE). *Entity Bean* merepresentasikan data dalam basis data relasional sebagai object Java. Biasanya setiap *entity bean* mempunyai sebuah tabel dalam basis data relasional dan setiap *instance* dari *bean* terhubung dengan sebuah baris di dalam tabel tersebut. *Entity bean*

bersifat persisten, dapat diakses oleh beberapa *client* secara konkuren (*shared access*), mempunyai *primary key* dan dapat berpartisipasi dalam relasi dengan *entity bean* yang lain. Pendekatan ini dikenal sebagai *object/relational mapping. Entity Bean* diklasifikasikan menjadi dua tipe berdasarkan cara menangani *persistensi* data bean tersebut.

1. Container-Managed-Persistence (CMP)

   Dalam *Container-Managed-Persistence, container* bertanggung jawab untuk menangani semua akses basis data yang diperlukan oleh entity bean. Dalam hal ini, program *bean* tidak perlu secara eksplisit melakukan panggilan mengakses basis data. Teknologi ini juga memungkinkan pemrograman *bean* menjadi tidak tergantung pada mekanisme penyimpanan khusus. Karena kefleksibelannya ini, jika dilakukan *deploy* ulang terhadap *entity bean* yang sama, pada server J2EE dan basis data yang berbeda, maka tidak perlu memodifikasi atau melakukan kompilasi ulang terhadap *bean.*

2. Bean-Managed-Persistent

   Dalam *Bean-Managed-Persistence*, *bean* bertanggung jawab menyimpan dan membaca data dari database, sehingga data yang dipegang oleh entity bean selalu sinkron dengan yang ada dalam database. Di dalam bean implementation dituliskan kode-kode untuk mengakses database.

c. Message Driven Bean

Message Driven Bean adalah komponen EJB yang khusus menangani antrian pesan yang dikirim oleh JMS (Java Messaging Service). Message Driven Bean adalah komponen terbaru dari spesifikasi EJB 2.0. Message Driven Bean ini tidak memiliki home interface, remote interface, maupun local interface. Ini dikarenakan memang tujuan dari message driven bean adalah untuk berkomunikasi dengan teknologi messaging lainnya, seperti MSMQ dari Microsoft.

2.1.2 Container EJB

Container EJB merupakan lingkungan eksekusi (*execution environment*) dimana komponen-komponen EJB dijalankan, yang berjalan dalam sebuah EJB server. *Container* menyediakan berbagai layanan sebagai berikut:

Component pooling dan manajemen siklus hidup bean

Teknik *component pooling* ini dimaksudkan untuk melakukan pengelolaan sumber daya sistem secara efisien. Dengan teknik ini maka sejumlah kecil *instance* dari *bean* dapat melayani sejumlah besar aplikasi *client* dengan cara mengalihkan penggunaan *bean* oleh *client* tertentu yang pasif ke *client* lain yang membutuhkannya.

Manajemen *Client Session*

Container bertanggung jawab untuk menyimpan dan mengembalikan *state* dari *bean*. Sebagaimana dibahas pada poin sebelumnya, *container* dapat memberikan sebuah *bean* ke sebuah client baru. Pada saat itu, *container* secara otomatis akan menyimpan *state* dari client yang lama. Ketika *client* lama tadi kembali memanggil *method* lain dalam *bean*, maka state yang disimpan tadi akan dimuat

kembali dan dikopi ulang ke *bean*, sebelum *bean* tersebut diberikan kepada *client* lama.

Connection pooling basis data

Container EJB memiliki mekanisme untuk mengelola sumber daya koneksi basis data. Mekanisme ini memungkinkan banyak *client* dapat terhubung ke koneksi basis data yang terbatas, sehingga akan memberikan keuntungan berupa penghematan sumber daya.

Manajemen transaksi (*Transaction management*)

EJB mendukung *declarative transaction*. *Declarative transaction* memungkinkan programmer dapat menentukan agar pemanggilan *method* EJB tertentu dilaksanakan secara transaksional. *Container* akan secara otomatis menangani proses transaksi, termasuk memulai transaksi, melakukan *commit* transaksi atau bahkan melakukan *rollback* secara otomatis, misalnya pada saat proses *recovery* setelah server mati secara tiba-tiba.

Pengelolaan Persistensi

Container dapat juga melakukan penyimpanan *state* dari EJB dalam basis data, dengan cara menyalin isi variabel *member* dari EJB ke kolom basis data. *Container* akan menyusun semua pernyataan SQL yang diperlukan untuk proses *update* basis data. Container juga akan memastikan integritas basis data.

2.1.3 Java Server Pages (JSP)

JavaServer Pages (JSP) menyediakan cara yang fleksibel untuk menyisipkan kode Java dalam kode-kode HTML untuk menghasilkan halaman yang dinamis. JSP dapat berisi HTML, kode Java dan komponen-komponen JavaBean. Pada

kenyataannya JSP adalah penyederhanaan dari model pemrograman servlet. Ketika seorang pengguna melakukan *request* terhadap sebuah halaman JSP, server web akan mengkompilasi halaman tersebut ke dalam sebuah servlet. Kemudian server web memanggil servlet tadi dan mengembalikan isi hasil kompilasi ke web *browser*. Sekali servlet dikompilasi dari halaman JSP, server web dapat dengan mudah menjalankan servlet tersebut tanpa perlu melakukan kompilasi ulang. Halaman *JSP* mempunyai dua tipe teks, yaitu *static template data*, yang dapat diekspresikan dalam format berbasis teks apa saja, seperti HTML, SVG, WML, serta XML, dan elemen JSP, yang menyusun *content* yang dinamis. Ekstensi file yang direkomendasikan untuk *source* file dari JSP page adalah `.jsp`. Halaman tersebut dapat terdiri dari sebuah file utama yang menyertakan file lain yang mengandung halaman JSP lengkap maupun pecahan dari halaman JSP. Ekstensi yang direkomendasikan untuk *source* file dari pecahan halaman JSP adalah `.jspf`. Elemen JSP pada sebuah halaman JSP dapat diekspresikan dalam dua sintaks, standar dan XML. Sebuah halaman JSP yang ditulis dalam sintaks XML adalah sebuah dokumen XML dan dapat dimanipulasi dengan menggunakan *tools* dan API untuk dokumen XML.

2.1.4 Java Servlet

Servlet adalah *class* pada bahasa pemrograman Java yang digunakan untuk mengembangkan kemampuan dari server yang menjadi host sebuah aplikasi yang diakses melalui model pemrograman *request-response*. Walaupun servlet dapat merespon semua tipe *request* tetapi servlet digunakan secara umum untuk mengembangkan aplikasi yang di-host oleh web server. Untuk aplikasi tersebut, teknologi servlet Java telah mendefinisikan class servlet HTTP yang spesifik. Paket

`javax.servlet` dan `javax.servlet.http` menyediakan *interface* dan *class* untuk penulisan servlet. Semua servlet harus mengimplementasikan *interface* servlet, yang mendefinisikan daur hidup (*life-cycle*) *method*. Dalam mengimplementasikan sebuah *service* yang generik, dapat digunakan atau mengembangkan dari *class* `GenericServlet` yang telah disediakan oleh Java Servlet API. *Class* `HttpServlet` menyediakan method seperti `doGet` dan `doPost` untuk menangani service yang spesifik pada HTTP.

2.1.5 Java Naming Directory Interface (JNDI)

Layanan penamaan dan direktori sudah menjadi hal yang umum dalam dunia komputasi. Layanan ini disediakan untuk mempermudah pencarian sebuah informasi, bahkan sebuah objek yang kompleks. Dengan layanan ini, informasi tentang sumberdaya seperti data login pengguna, koneksi ke basis data, atau bahkan lokasi printer di jaringan, didaftarkan di sebuah layanan direktori yang terpusat. *Client* yang membutuhkannya dapat memperoleh informasi tersebut dari server menggunakan protokol tertentu. Java Naming and Directory Interface (JNDI) menyediakan antarmuka terhadap layanan penamaan dan direktori. JNDI juga menyediakan sekumpulan API dengan *method-method* untuk menjalankan operasi *standard directory*, seperti mengasosiasikan atribut dengan objek dan mencari objek menggunakan atribut-atributnya. Dengan menggunakan JNDI, sebuah aplikasi dapat menyimpan dan mengambil tipe dari penamaan Java object. JNDI tidak bergantung pada implementasi layanan yang spesifik, namun bersifat mengabstraksi layanan-layanan direktori yang sudah ada selain menyediakan layanan direktori sendiri. Aplikasi Java dapat menggunakan JNDI untuk mengakses beberapa layanan

directori, termasuk penamaan yang ada dan layanan direktori seperti LDAP, NDS, DNS, dan NIS.

2.1.6 Apache-ant

Apache-ant adalah sebuah tools untuk melakukan otomasi dalam proses develop suatu software. Seperti diketahui, proses development suatu software tidak terlepas dari langkah-langkah inisialisasi variable lingkungan, kompilasi program, pembuatan file .jar, dan seterusnya. Proses ini terjadi berulang-ulang, dengan urutan yang tetap. Oleh karena itu, apache-ant menawarkan suatu fitur yang memungkinkan kita untuk mendefinisikan di awal task-task apa saja yang akan dilakukan tersebut, sehingga kemudian nantinya, proses tersebut akan dilakukan secara otomatis oleh apache-ant.

Pengaturan task-task tersebut didefinisikan di dalam sebuah file yang lazimnya diberi nama *build.xml*. Berikut adalah contoh isi dari file build.xml :

```
<project name="MyProject" default="dist" basedir=".">
    <description>
        simple example build file
    </description>
  <!-- set global properties for this build -->
  <property name="src" location="src"/>
  <property name="build" location="build"/>
  <property name="dist"  location="dist"/>

  <target name="init">
    <!-- Create the time stamp -->
    <tstamp/>
    <!-- Create the build directory structure used by compile -->
    <mkdir dir="${build}"/>
  </target>
  <target name="compile" depends="init"
        description="compile the source " >
    <!-- Compile the java code from ${src} into ${build} -->
    <javac srcdir="${src}" destdir="${build}"/>
  </target>

  <target name="dist" depends="compile"
        description="generate the distribution" >
```

```
    <!-- Create the distribution directory -->
    <mkdir dir="${dist}/lib"/>

    <!-- Put everything in ${build} into the MyProject-${DSTAMP}.jar
file -->
    <jar jarfile="${dist}/lib/MyProject-${DSTAMP}.jar"
basedir="${build}"/>
  </target>

  <target name="clean"
        description="clean up" >
    <!-- Delete the ${build} and ${dist} directory trees -->
    <delete dir="${build}"/>
    <delete dir="${dist}"/>
  </target>
</project>
```

Gambar 2.2 Script build.xml

Dalam file konfigurasi apache-ant, setiap buildfile dapat memiliki satu buah project name, dan satu atau lebih target action. Setiap target action ini memiliki nama yang nantinya dapat direfer oleh target action lainnya. Oleh karena itu pula, penamaan setiap target ini haruslah unik.

Project

Sebuah project secara default memiliki 3 buah attribute yaitu :

Tabel 2.1 Tabel Atrribute Project

Attribute	Deskripsi	Required
name	Nama dari project yang bersangkutan	No
default	Default target yang dituju bila tidak ada argument yang diberikan dalam command line.	Yes.
basedir	Direktori dasar dimana semua setting path akan merujuknya sebagai root direktori. Bila nilai dari property ini tidak diberikan, maka akan digunakan nilai direktori dimana saat ini file build.xml berada.	No

Target

Sebuah target dapat memiliki dependensi terhadap target yang lainnya. Di dalam file konfigurasi ini kita dapat mendefinisikan bermacam-macam target,

diantaranya seperti: target untuk mengkompilasi, mengkopi file-file library yang dibutuhkan, mengeset nilai variable CLASSPATH, membuat file .jar, dan sebagainya. Dari sekian banyak target ini, kita dapat mendefinisikan suatu dependensi antar target, sehingga urutan eksekusi task untuk setiap target dapat dilakukan sesuai dengan urutan yang seharusnya. Berikut adalah contoh penulisan yang memiliki dependensi :

```
<target name="A"/>
<target name="B" depends="A"/>
<target name="C" depends="B"/>
<target name="D" depends="C,B,A"/>
```

Gambar 2.3 Contoh Script Dependensi

Sedangkan berikut ini adalah property yang dimiliki oleh target :

Tabel 2.2 Tabel Property Target

Attribute	Deskripsi	Required
name	Nama target yang bersangkutan	Yes
depends	Daftar nama target dependensi yang dipisahkan oleh koma.	No
if	Nama property yang harus diset agar target yang bersangkutan dapan dieksekusi.	No
unless	Nama property yang tidak boleh diset agar target yang bersangkutan dapan dieksekusi.	No
description	Deskripsi singkat dari fungsi target yang bersangkutan..	No

Task

Task adalah bagian dari command yang akan dieksekusi nantinya. Sebuah task dapat memiliki beberapa atribut. Nilai dari atribut ini dapat merefer ke nilai variable di bagian inisialisasi. Struktur umum sintaks dari task ini adalah :

```
<name attribute1="value1" attribute2="value2" ... />
```

Properties

Sebuah project dapat memiliki beberapa property. Nilai-nilai property ini dapat diset di bagian inisialisasi di awal file, ataupun di luar apache-ant. Dalam contoh sebagai berikut :

```
<property name="src" location="src"/>
<property name="build" location="build"/>
<property name="dist"  location="dist"/>
```

Gambar 2.4 Contoh Script Properties

Selain itu, apache-ant juga memiliki property default sebagai berikut :

Tabel 2.3 Tabel Property Default Apache- ant

Nama Property	Keterangan
Ant.file	Path absolute dari buildfile milik apache-ant ini.
Ant.version	Versi dari apache-ant yang digunakan
Ant.project.name	Nama project yang saat ini sedang dieksekusi
Ant.java.version	Versi JVM yang terdeteksi.
Basedir	Path absolute dari project yang bersangkutan.

Kemudian setelah file build.xml ini dibuat, maka untuk mengeksekusi task yang diinginkan tinggal diketikkan di console sebagai berikut :

```
shell> ant nama_task
```

2.1.7 Teknologi Hibernate

Hibernate adalah sebuah teknologi yang mempermudah programmer untuk melakukan proses mapping representasi data dari suatu model object menuju ke suatu data model relasional dengan menggunakan skema SQL. Hibernate tidak hanya menangani dalam proses mapping object-object java ke table secara fisik,

namun juga menyediakan fasilitas query data dan fasilitas penelusuran kembali, sehingga tentunya dapat secara signifikan mereduksi waktu development yang banyak terbuang dalam proses manipulasi data dengan SQL dan JDBC. Gambar 2.4 menjelaskan lebih lanjut tentang posisi hibernate dalam aplikasi.

Gambar 2.5 Posisi hibernate dalam aplikasi

Pengaturan proses mapping dari object menuju tempat penyimpanan secara fisik di database dilakukan oleh hibernate dengan membaca file konfigurasi yang biasanya diberi nama *.hbm.xml. Gambar 2.5 menunjukkan contoh isi file konfigurasi tersebut.

```
<?xml version="1.0"?>
<!DOCTYPE hibernate-mapping PUBLIC
        "-//Hibernate/Hibernate Mapping DTD//EN"
```

```
        "http://hibernate.sourceforge.net/hibernate-mapping-
2.0.dtd">

<hibernate-mapping package="eg">
        <class name="Cat" table="CATS">
                <id name="id" column="uid" type="long">
                        <generator class="hilo"/>
                </id>
                <property name="birthdate" type="date"/>
                <property name="color" not-null="true"/>
                <property name="sex" not-null="true"/>
                <property name="weight"/>
                <many-to-one name="mate"/>
                <set name="kittens">
                        <key column="MOTHER"/>
                        <one-to-many class="Cat"/>
                </set>
        </class>
</hibernate-mapping>
```

Gambar 2.6 Contoh konfigurasi mapping hibernate.

Dari contoh pada gambar 2.6, terdapat bagian-bagian penting dari file yang menangani pemetaan antara object java, dengan field yang bersangkutan secara fisik dalam tabel di database. Dalam hal ini diambil suatu contoh yang melakukan mapping object "Cat" ke dalam table "CATS". Object cat memiliki property-property yang akan dipetakan ke dalam table secara fisik, diantaranya *id*, *birthdate*, *color*, *sex*, *weight*, dan *mate*. Selain itu terdapat juga sebuah field yang bernama *kittens* dan memiliki relasi one-to-many terhadap tabel itu sendiri, dan relasi tersebut diidentifikasikan dengan nama "*MOTHER*".

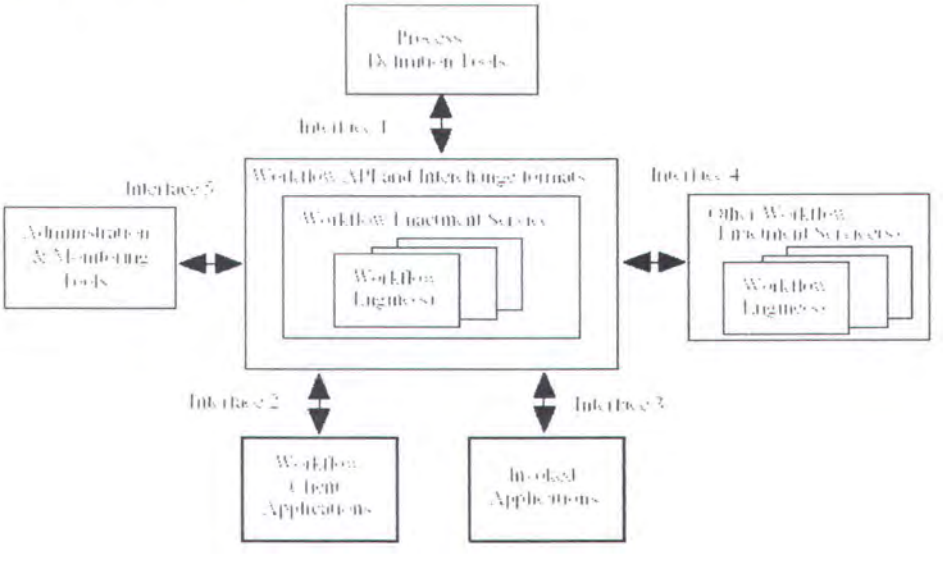
```
<class name="Cat" table="CATS">
        <id name="id" column="uid" type="long">
                        <generator class="hilo"/>
        </id>
        <property name="birthdate" type="date"/>
        <property name="color" not-null="true"/>
        <property name="sex" not-null="true"/>
        <property name="weight"/>
        <many-to-one name="mate"/>
        <set name="kittens">
                <key column="MOTHER"/>
```

```
            <one-to-many class="Cat"/>
        </set>
</class>
```

Gambar 2.7 Contoh konfigurasi mapping object ke table

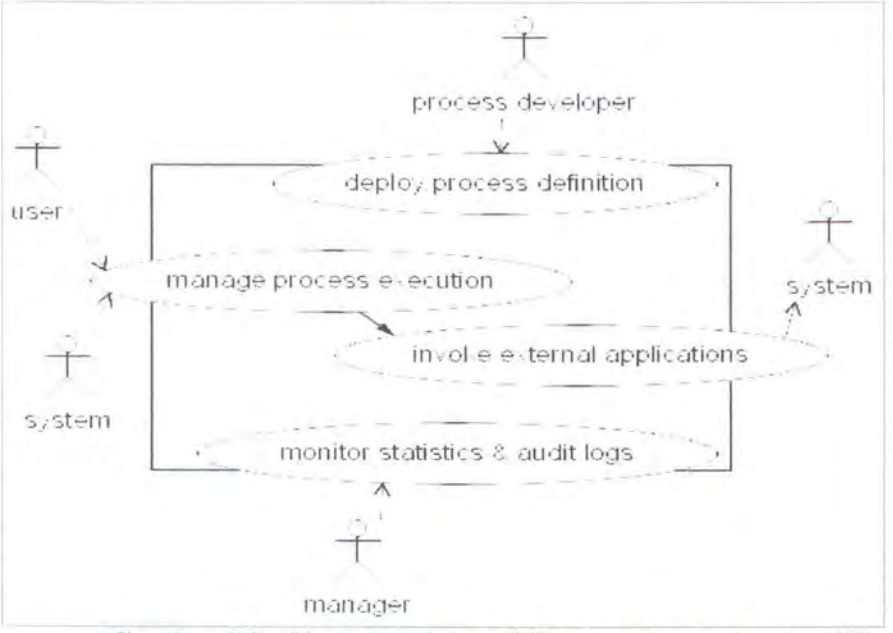
2.2 Workflow Management System

Workflow management system adalah suatu *software* yang menerima inputan berupa deskripsi formal dari proses bisnis serta kemudian mendelegasikan eksekusi dari proses-proses bisnis tersebut pada *resource-resource* yang ada, termasuk sumber daya manusia, dan sumber daya lain yang dimiliki dalam lingkungan tersebut[2]. Intinya adalah otomasi proses bisnis, dimana di dalamnya terdapat dokumen atau informasi mengalir melalui suatu prosedur yang baku. Terdapat 2 definisi dasar yang perlu diketahui, yakni *process definition* dan *process instance. Process definition* adalah definisi formal secara teoritis dari bisnis proses yang ada[2]. Sedangkan bentuk nyata / pelaksanaan dari definisi proses tersebut disebut *process instance*. Secara umum, gambar berikut menggambarkan tentang posisi perangkat lunak *workflow management system.*



Gambar 2.8. Posisi perangkat lunak workflow management system [9]

Sedangkan gambar berikut menggambarkan apa aja yang terjadi dalam suatu perangkat lunak *workflow management system.*



Gambar 2.9 Alur perangkat workflow management system [2]

Berikut adalah daftar istilah yang lazim dipakai dalam *workflow management system*:

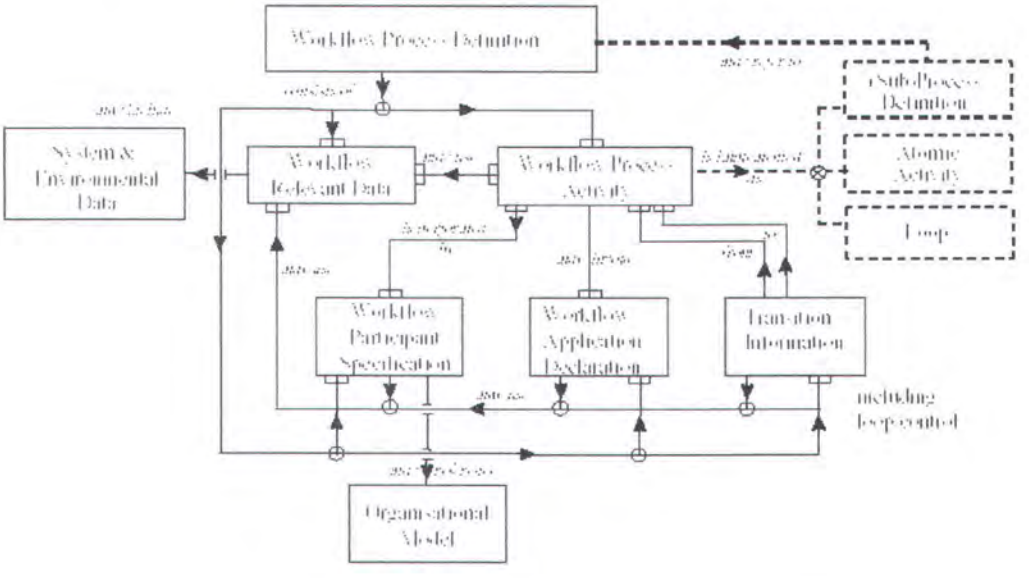
2.2.1 Business Process

Merupakan sekumpulan proses/aktivitas atau lebih, yang secara kolektif menggambarkan tujuan bisnis atau kebijakan tertentu, dan lazimnya menggambarkan struktur organisasi yang mendefinisikan role dan hubungan satu sama lain. Business process biasanya dihubungkan dengan tujuan operasional dan hubungan antar bisnis. Sebuah business process biasanya melibatkan interaksi dari beberapa *workflow participant* sekaligus[9].

2.2.2 Process Definition

Merupakan representasi dari business process dalam sebuah form yang mendukung manipulasi secara otomatis, termasuk didalamnya *modeling* dan pembuatan aturan-aturan tertentu (enactment) oleh *workflow management system.* Process definition ini terdiri atas hubungan erat beberapa aktivitas dengan relasinya masing-masing, kriteria untuk menandakan mulai atau berakhirnya suatu proses, dan informasi tentang masing-masing aktivitas, seperti *workflow participants*, aplikasi-aplikasi IT yang terkait beserta datanya, dsb. [9].

WfMC Process Definition Meta-Model



Gambar 2.10 WfMC Process Definition Layer

Process Definition berisi 4 tingkatan / layers sebagai berikut[2] :

a. State Layer

State layer mengekspresikan kondisi serta aliran control. Dalam standar pemrograman, aliran control diturunkan dari arsitektur Von Neuman. Didalamnya didefinisikan urutan dari instruksi yang harus dijalankan. Aliran control ini ditentukan berdasarkan urutan ketika kita menuliskan instruksi seperti *if-statement*, *looping*, dan sebagainya. Aliran kontrol untuk setiap *business process* pada umumnya sama. Sebuah state, atau pada umumnya lebih dikenal dengan *wait-state* dalam sebuah proses mengindikasikan dependensi terhadap aktor eksternal. Contohnya adalah sebagai berikut : Misalkan terdapat sistem X atau aktor Y yang saat ini harus melakukan sesuatu, namun untuk melakukannya dibutuhkan pemicu berupa tanda dari luar yang menandakan selesainya pekerjaan lain. Suatu state mengindikasikan adanya **dependensi terhadap hasil pekerjaan dari faktor eksternal**. Suatu state dalam process definition juga menjelaskan aktor mana yang bertugas mengendalikan eksekusi suatu pekerjaan. Sedangkan yang dimaksud dengan aliran kotrol dalam suatu *process definition* adalah kumpulan dari state yang dihubungkan dengan relasi antar state itu sendiri. Aturan *logic* dalam state tersebut menentukan, state mana yang berjalan bersamaan, dan state mana yang berhalan secara terpisah / eksklusif.

b. Context Layer

Suatu *process context variable*, atau selanjutnya disebut dengan variable adalah sebuah variable yang berhubungan dengan *process instance*. *Process variable* memperbolehkan *process developer* untuk menyimpan suatu data selama daur hidup dari *instance* tersebut. Variable ini juga dapat digunakan untuk menyimpan referensi / pointer. Sebuah variable dapat mengacu pada suatu

record tertentu dalam database, ataupun sebuah *network drive*. Aspek lainnya yang menarik berhubungan process variable adalah bagaimana *workflow management system* ini merubah dari data menjadi informasi. Tugas *workflow* sebenarnya adalah menggabungkan antara tugas dengan data antar sistem yang berbeda-beda dalam suatu organisasi.

c. Programming Logic Layer

Kembali disebutkan bahwa *action* adalah bagian dari logika pemrograman yang dijalankan oleh *workflow management system* selama suatu *event* tertentu terjadi selama eksekusi proses. Layer ini menggabungkan semua bagian dari *software* dengan informasi yang akan dijalankan ketika suatu event tertentu terjadi. Contoh dari *programming logic* misalnya : pengiriman email secara otomatis, pengiriman pesan pada *message broker*, pengambilan data pada mesin ERP, dan juga proses *update* database.

d. User Interface Layer

Ketika sebuah aktor memicu terjadinya suatu proses menuju *state* akhir, biasanya data-data dari event tersebut ditampung dalam suatu variable. Misalnya ketika seseorang menentukan apakah *state* evaluasi sudah berakhir, maka orang tersebut akan menentukan hasil evaluasi tersebut apakah diterima atau tidak. Dari informasi tersebut, form *user interface* dapat menghasilkan signal yang meminta informasi tertentu pada user.

2.2.3 Activity

Merupakan deskripsi dari masing-masing bagian kerja yang membentuk suatu langkah logikaldalam sebuah proses. Suatu aktivitas dapat termasuk *manual*

activity, ataupun *automated activity*. Sebuah aktivitas workflow membutuhkan dukungan resource manusia atau mesin dalam menjalankannya, dimana nantinya ini akan menjadi bagian dari *workflow participant.* [9].

2.2.4 Automated Activity

Merupakan aktivitas dalam *business process* yang dapat dilakukan secara otomatis oleh komputer menggunakan *workflow management system* selama pelaksanaan *business process* dimana aktivitas tersebut merupakan bagian didalamnya. [9].

2.2.5 Manual Activity

Merupakan aktivitas dalam *business process* yang tidak dapat dilakukan secara otomatis dan oleh karena itu berada di luar lingkup dari *workflow management system.* [9].

2.2.6 Instance

Merupakan representasi dari pelaksanaan suatu proses, atau aktivitas dalam suatu proses termasuk didalamnya data yang terkait. Setiap *instance* merepresentasikan *thread* eksekusi dari aktivitas atau proses yang dapat dikendalikan secara independen. [9].

2.2.7 Workflow Participant

Merupakan sumber daya yang melakukan pekerjaan yang telah digambarkan dalam pemodelan aktivitas workflow. Pekerjaan-pekerjaan ini umumnya berada dalam suatu *work items* yang ditugaskan kepada suatu *workflow participant* melalui

worklist. Pada umumnya, istilah *workflow participant* mengacu pada sumber daya manusia, namun tidak menutup kemungkinan untuk mengikutsertakan juga didalamnya, suatu resource berbasis mesin seperti *intelligent agent*. [9].

2.2.8 Work Items

Merupakan representasi dari pekerjaan yang harus dikerjakan oleh *workflow participant* dalam konteks *activity instance* atau *process instance*. Sebuah aktivitas biasanya menghasilkan satu atau lebih *work item* yang kemudian secara bersama-sama akan dikerjakan oleh *workflow participant*. [9].

2.2.9 Work List

Merupakan suatu daftar yang berisi hubungan antara *work items* dengan *workflow participant*. [9].

2.2.10 Work List Handler

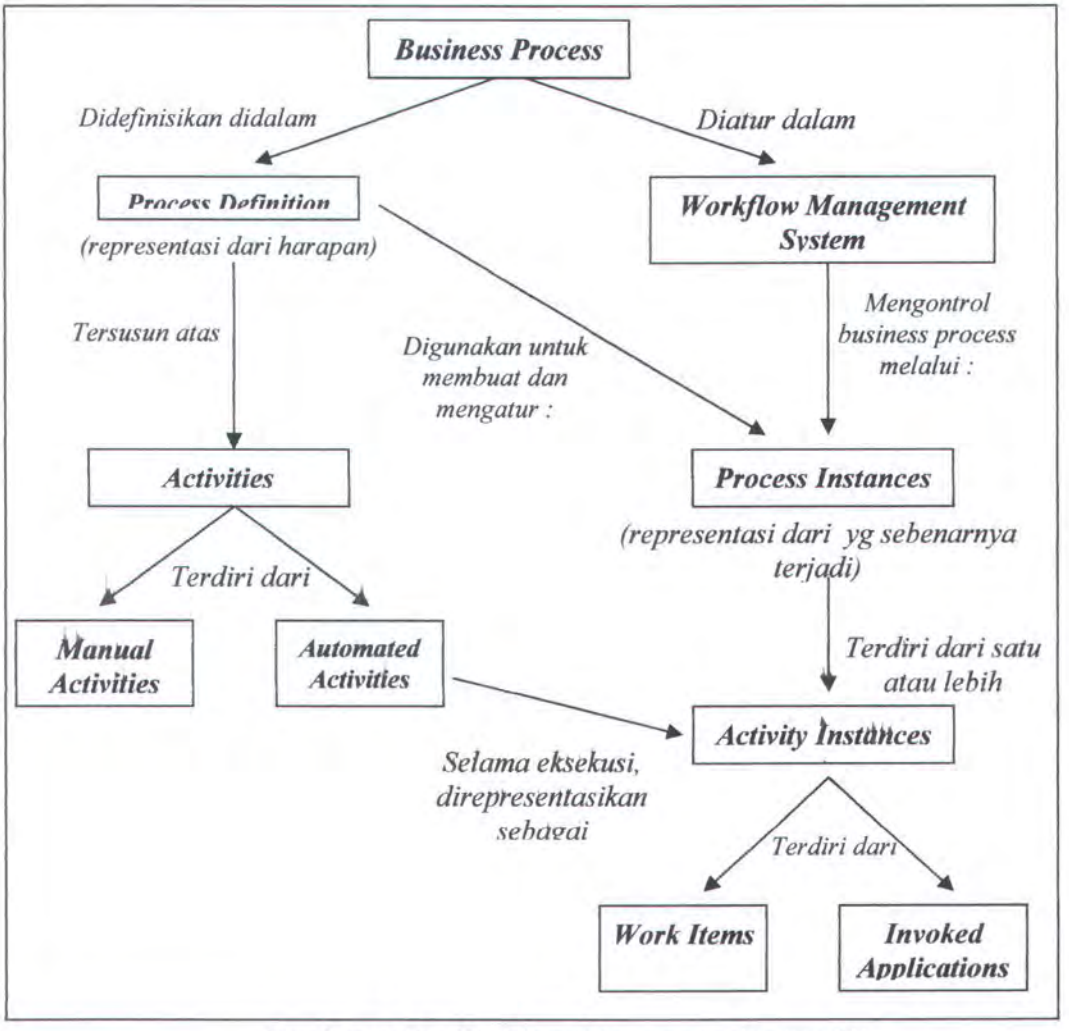
Merupakan suatu komponen dari *software* yang mengatur hubungan antara user dengan worklist yang diatur oleh mesin *workflow*, termasuk didalamnya pendelegasian suatu tugas, dan *return value* dari tugas yang didelegasikan tersebut.

Gambar 2.7 menggambarkan hubungan antar istilah yang ada dalam *workflow management system.*

2.3 Struts Framework

Framework adalah *sebuah lingkungan software yang dibuat untuk memenuhi kebutuhan sebuah domain secara spesifik, termasuk di dalamnya komponen-komponen yang digunakan dalam membangun aplikasi seperti API, service, dan*

tools yang mereduksi kemampuan yang harus dimiliki oleh programmer dalam menyelesaikan tugasnya[10]. Inti dari Struts Framework disini terutama pada fleksibilitasnya menangani layer control berdasarkan spesifikasi MVC. Struts menyediakan komponen Controller nya sendiri, dan dapat berintegrasi dengan teknologi lain untuk menyediakan bagian Model dan View. Untuk sisi Model nya, Struts dapat berintegrasi dengan teknologi standar untuk mengakses database, seperti JDBC dan Hibernate. Sedangkan untuk sisi View nya, Struts dapat berintegrasi dengan JSP, termasuk JSF dan JSTL.



Gambar 2.11 Workflow Management System [9]

2.3.1 Model View Controller

Model-View-Controller adalah Pola Software yang menggambarkan metode untuk memecah software menjadi beberapa class yang berinteraksi dengan cara yang mudah dimengerti[11]. Seperti telah disinggung sebelumnya bahwa MVC mengorganisasi aplikasi menjadi 3 bagian utama, yaitu :

a. **Model**, dengan tugas untuk merepresentasikan data serta menangani urusan business logic. *Business logic* adalah jantung dari aplikasi yang dibuat serta bertanggungjawab secara langsung dalam memenuhi kebutuhan fungsional dari suatu aplikasi, yakni dalam hal akses, manipulasi, dan pemrosesan data. Representasi dari layer ini meliputi *class, package, servlet* dan macam-macam *bean* yang menyediakan fungsi yang dibutuhkan.

b. **View**, dengan tugas untuk merepresentasikan data yang dihasilkan oleh *business logic* serta menangani *request* inputan dari user untuk dikirim ke *Business Logic*. Representasi dari layer ini adalah script JSP dan HTML (javascript,css). Bagian ini bekerjasama dengan layer Model melalui layer *Controller*.

c. **Controller** , dengan tugas untuk mengatur alur kontrol program. Bagian ini berperan sebagai titik pusat kontrol dalam aplikasi. Representasi dari layer ini biasanya adalah sebuah servlet yang bertindak sebagai *dispatcher*, pengatur, yang mendelegasikan setiap permintaan (*request*) yang diterimanya, ke layer lain yang bersangkutan. Dalam implementasinya di teknologi java, servlet menggunakan *interface RequestDispatcher* dari class javax.servlet untuk melaksanakan tugasnya. Keuntungan dari adanya *controller* tunggal dalam

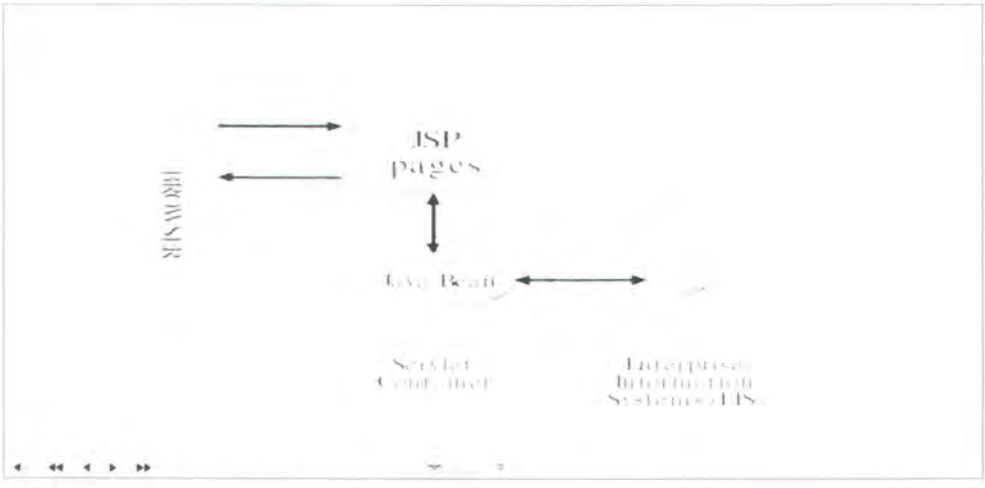
sebuah aplikasi ini adalah kemudahan konfigurasi dari webserver (karena hanya ada 1 buah *mount point* dan *servlet mapping* untuk tiap aplikasi), serta adanya interface dan alur yang konsisten diantara layer-layer dalam aplikasi. Sedangkan keuntungan dalam memisahkan layer presentasi dengan layer business logic yang berkomunikasi melalui controller diantaranya adalah :

1. Kemudahan dalam maintenance code, karena terpisahkan antara code untuk presentasi dengan business logic.
2. Pembagian tugas yang lebih jelas untuk tiap-tiap anggota team. Misalnya : anggota team yang bertugas dalam design interface akan lebih mudah dan tidak terikat dengan anggota team untuk business logic dalam penggunaan tools yang dipakai nantinya.

Dalam prakteknya, MVC dibagi menjadi 2 buah model, yaitu :

1. **MVC Model 1 – Page Centric**

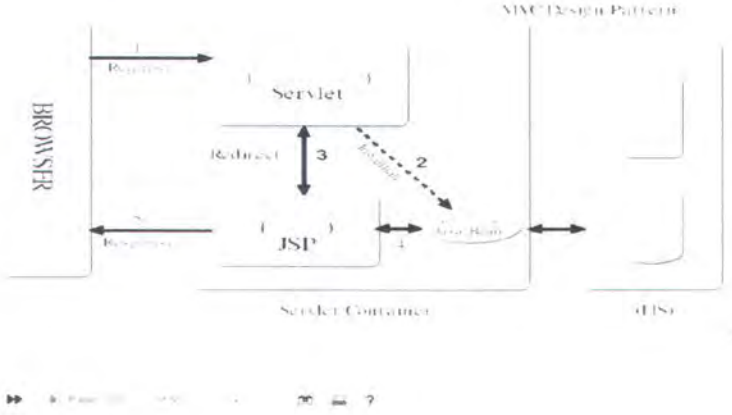
   Model ini tersusun atas sekumpulan halaman JSP. Halaman JSP tersebut nantinya akan menangani seluruh aspek aplikasi, termasuk di dalamnya proses bisnis, control program, serta presentasi dari program itu sendiri. Bagian-bagian aspek tadi dapat berbentuk Java beans, scriptlet, ataupun expresi, yang kemudian akan dipanggil oleh halaman JSP. Selain itu, ciri-ciri lainnya dari model ini adalah di bagian perpindahan halaman. Ada 2 cara bagi user untuk berpindah dari halaman satu ke lainnya, yaitu dengan meng-klik hyperlink, atau bisa juga dengan men-submit form request. Dalam gambar 2.8 digambarkan bagan dari model 1 ini.



Gambar 2.12 Model Page Centric

2. MVC Model 2 – Servlet Centric

Gambar 2 menunjukkan bagan umum dari MVC model 2. Inti dari model 2 ini adalah adanya suatu controller berupa servlet. Servlet ini menerima request data dari user, melakukan pemrosesan awal dengan membentuk instance dari JavaBean mana yang dibutuhkan, dan memforward hasilnya ke halaman JSP tertentu. Halaman JSP disini hanya digunakan sebagai penampil data dalam layer presentasi, sedangkan control dan application logic diatur oleh sebuah atau sekumpulan servlet.



Gambar 2.13 Model Servlet Centric

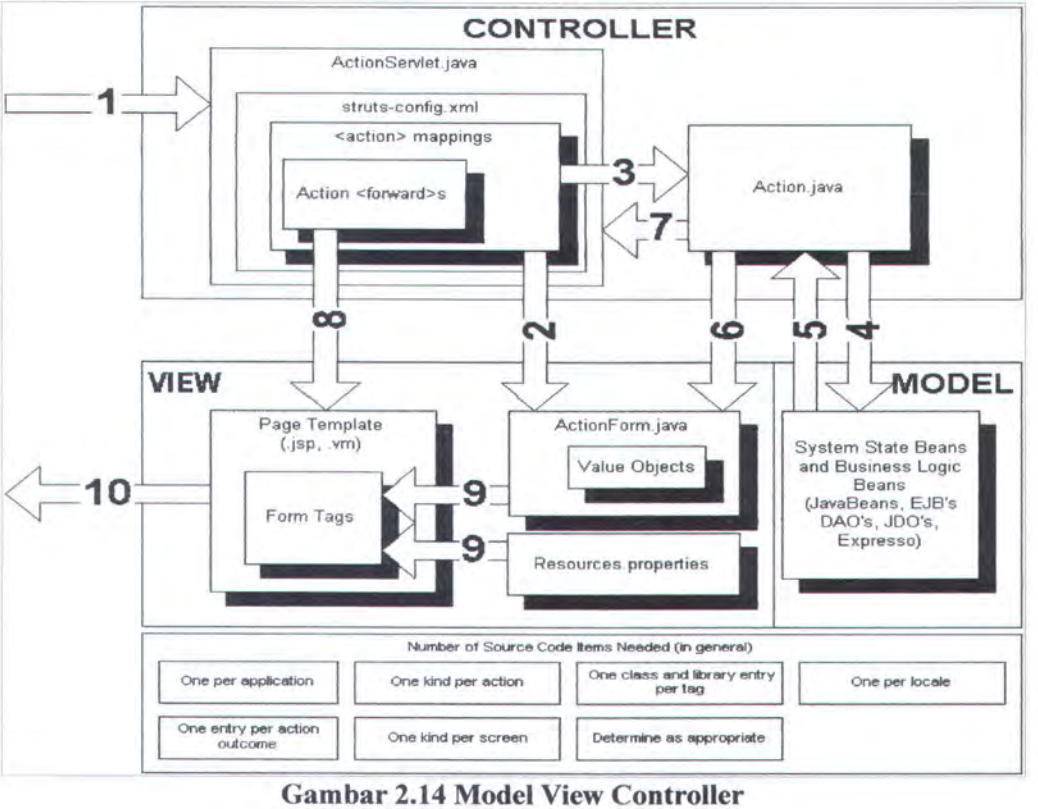
Servlet-servlet yang ada pada model ini bertugas sebagai berikut :

1. Sebagai gatekeeper

   Menyediakan service umum, seperti otentikasi, otorisasi, login, error handling, dsb

2. Sebagai central controller

   Sebagai sebuah state machine yang menentukan bean mana yang harus dibuat instance nya untuk melayani request tertentu dari user. Adanya fungsi dari servlet inilah yang membedakan antara MVC Model 1 dan 2.



Gambar 2.14 Model View Controller

2.3.2 Struts FrameWork

Berkaitan dengan Model View Controller yang telah dibahas sebelumnya, struts telah menyediakan sebuah servlet yang bertugas sebagai

controller. Dalam file konfigurasi untuk tiap-tiap aplikasi, maka keberadaan controller servlet ini harus diberitahukan sebelumnya. Sehingga di dalam file konfigurasi web.xml harus ditambahkan baris sebagai berikut :

```
<servlet>
  <servlet-name>action</servlet-name>
  <servlet-class>
    org.apache.struts.action.ActionServlet
  </servlet-class>
  <init-param>
    <param-name>config</param-name>
    <param-value>
      /WEB-INF/struts-config.xml
    </param-value>
  </init-param>
  <load-on-startup>1</load-on-startup>
</servlet>
```

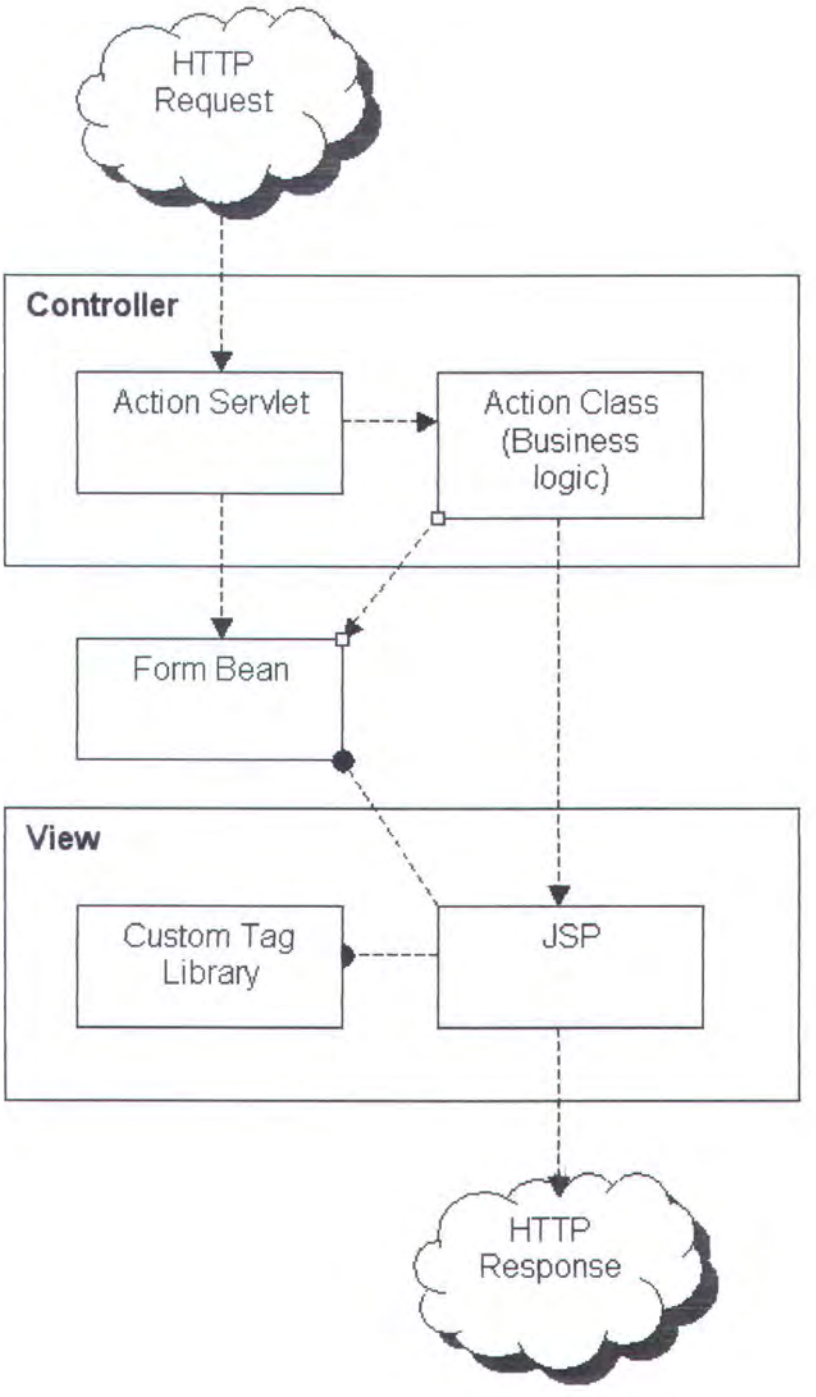
Gambar 2.15 konfigurasi tambahan web.xml

Sedangkan agar controller tersebut dapat dipanggil oleh aplikasi, maka diperlukan mapping request handler yang ditambahkan juga pada file web.xml sebagai berikut :

```
  <servlet-mapping>
     <servlet-name>action</servlet-name>
     <url-pattern>/do/*</url-pattern>
   </servlet-mapping>
```

Gambar 2.16 konfigurasi request handler controller servlet

Untuk selanjutnya, setiap request yang berakhiran tag *.do akan dihandle oleh controller servlet ini, dan selanjutnya akan diproses lebih lanjut oleh struts, sehingga bagan requestnya bisa digambarkan dalam gambar 2.13.



Gambar 2.17 Proses Request Handling oleh struts

2.4 Perbandingan dengan software yang sudah ada

Dalam bagian ini akan dibandingkan dua buah software workflow management system, yang juga mengimplementasikan teknologi J2EE, yaitu JBPM (Java Business Process Modelling) dan Xflow.

2.4.1 JBPM (Java Business Process Modelling)

JBPM adalah sebuah software workflow management system yang extensible, dan flexible. Proses bisnis yang ada, digambarkan di dalam sebuah bahasa berformat XML. Kemudian setelah proses ini digambarkan, sebuah server akan menerima file berisi definisi proses bisnis dalam format XML tersebut, dan mendeploynya menjadi sebuah aplikasi. Definisi proses bisnis dalam format XML ini menggunakan bahasa yang telah didefinisikan sendiri oleh JBPM, yaitu JPDL (JBPM Process Definition Language). Software ini juga mengikutsertakan Jboss Application Server sebagai bagian dari softwarenya. Sedangkan untuk mempermudah user agar tidak terlalu sulit dalam mendefinisikan proses bisnisnya, project open source lainnya, yaitu JGPD (Java Graphical Process Designer) membuat sebuah antarmuka yang cukup interakfit. Dalam antarmuka ini user bisa menggambarkan proses bisnis dengan melakukan drag and drop object-object yang dibutuhkan. Gambar 2.13 menunjukkan antarmuka dari JGPD ini. JGPD merupakan project yang terpisah dari JBPM itu sendiri, sehingga tidak secara khusus dibuat untuk JBPM. JGPD ini dapat mengekspor hasil gambar menjadi beberapa format file, diantaranya yaitu : JPG, GIF, PNG, dan format JPDL itu sendiri.

c. Mendukung hierarki n-level workflow.

d. Mendukung Work Item yang berbasis aturan

e. Work Item dapat berupa Object Java atau XML

f. Mendukung parallel Flow.

g. Kemampuan untuk melihat workflow state kapan saja

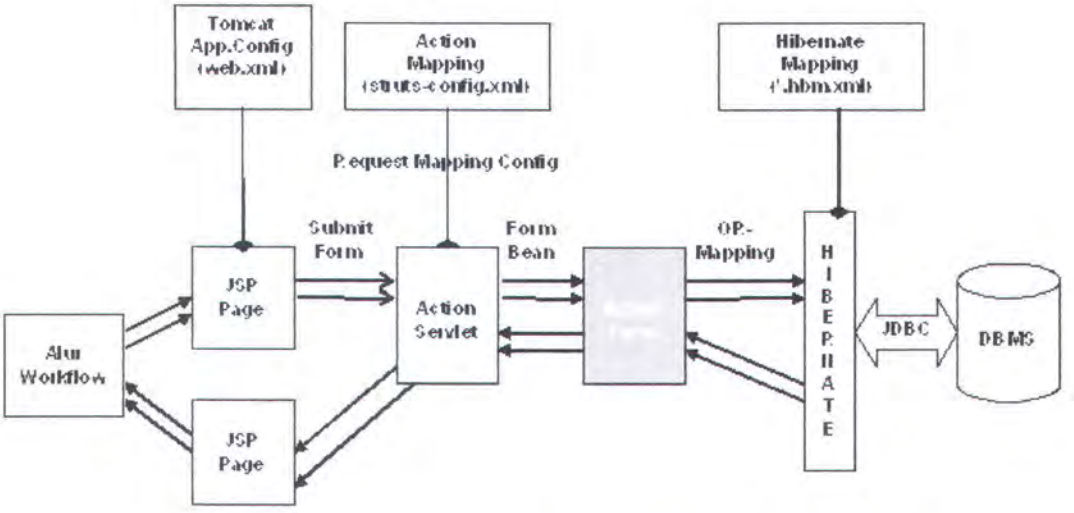
h. Modul keamanan yang dapat dimodifikasi dan ditambahkan sendiri.

BAB III
ANALISA DAN DESAIN PERANGKAT LUNAK WORKFLOW

BAB III

ANALISA DAN DESAIN PERANGKAT LUNAK WORKFLOW

Bab ini menjelaskan tentang elemen-elemen yang berpengaruh, tahapan-tahapan perancangan serta desain perangkat lunak aplikasi *workflow* ini. Gambar 3.1 menunjukkan pemetaan teknologi yang akan diimplementasikan dalam tugas akhir ini.



Gambar 3.1 Rancangan Schema Teknologi yang diimplementasikan

3.1 Elemen Aplikasi Workflow

Elemen-elemen sistem ini adalah hal-hal yang akan terlibat atau menjadi bagian dari sistem aplikasi workflow ini. Elemen-elemen ini kemudian nantinya akan disusun untuk menggambarkan suatu task process dari aplikasi workflow. Berikut adalah tabel berisi elemen-elemen dalam sistem beserta penjelasannya :

Tabel 3.1 Elemen Aplikasi Workflow

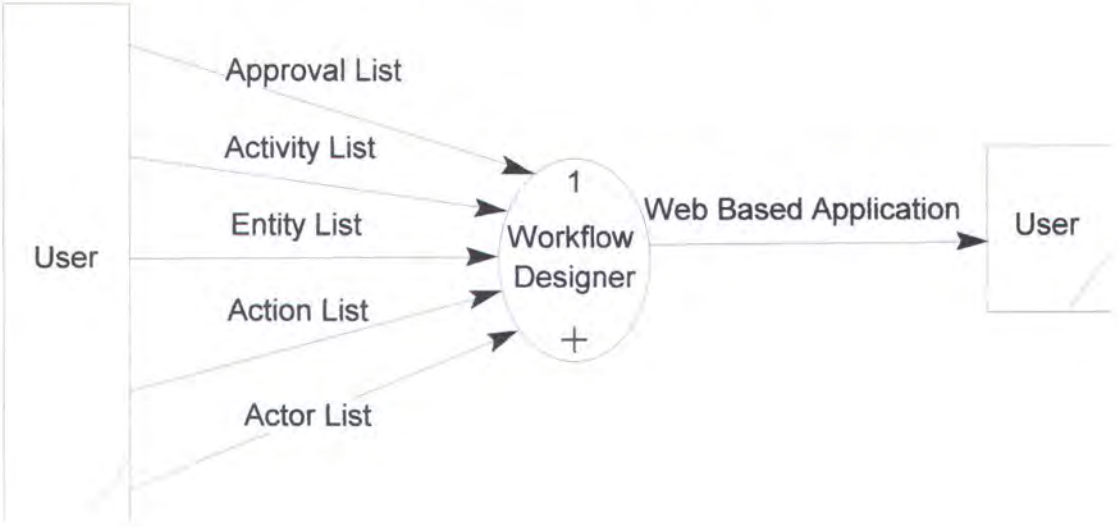
Nama Elemen	Keterangan	Behaviour / Fungsi
Aktor	Merupakan pihak yang menjalani dan menggunakan sistem ini. Aktor disini dapat berupa single person, ataupun suatu lembaga / organisasi. Suatu aktor dapat bertipe entity saja, approval saja, ataupun keduanya.	Antar aktor dihubungkan oleh activity atau action. Jika aktor bertipe entity, maka akan terjadi interaksi input/output data sesuai attribute yang dimiliki. Namun apabila aktor bertipe approval, maka hanya akan terjadi perubahan flag approval.
Entity	Merupakan tipe atribut yang dimiliki oleh tiap aktor.	Berfungsi menampung data atribut yang dimiliki oleh tiap aktor, untuk kemudian nantinya menjadi field di database dalam aplikasi yang dibangun.
Approval	Merupakan tipe flag yang dimiliki oleh aktor.	Berperan sebagai penanda bahwa proses yang dijalani telah melalui aktor yang bersangkutan.
Activity	Kegiatan yang dilakukan oleh aktor untuk menuju ke proses selanjutnya.	Activity disini menunjukkan suatu proses sekuensial antar aktor yang berurutan.

3.2 Perancangan Perangkat Lunak

Pada bagian perancangan perangkat lunak ini untuk pemodelan sistem digunakan Power Designer Process Analyst versi 6.0. Diagram-diagram yang digunakan dalam proses perancangan ini adalah Process Analyst Model.

3.2.1 Perancangan Struktur File Deskripsi untuk Workflow Diagram

Proses yang digambarkan dalam diagram power designer berikut ini menggambarkan langkah-langkah beserta aliran data yang harus dilakukan oleh user dalam aplikasi ini, untuk menghasilkan output berupa aplikasi.

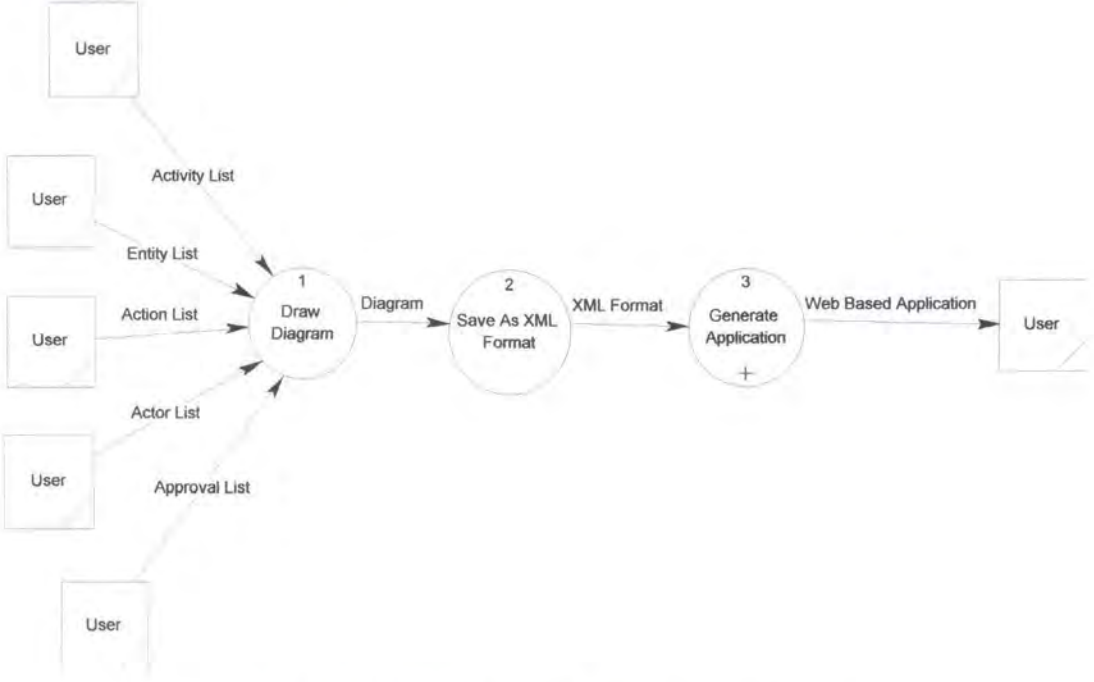


Gambar 3.2 DFD Level 0 Aplikasi Workflow Diagram

DFD level 0 pada gambar 3.1 menggambarkan tentang eksternal entity apa saja yang berpengaruh beserta proses input dan output data secara global. Seperti terlihat dalam gambar, hanya terdapat satu macam eksternal entity / dalam hal ini yaitu user, yang berinteraksi dengan aplikasi ini. Dalam aplikasi ini nantinya, user diharuskan memasukkan inputan berupa:

- approval list
- activity list
- entity list
- action list
- aktor list

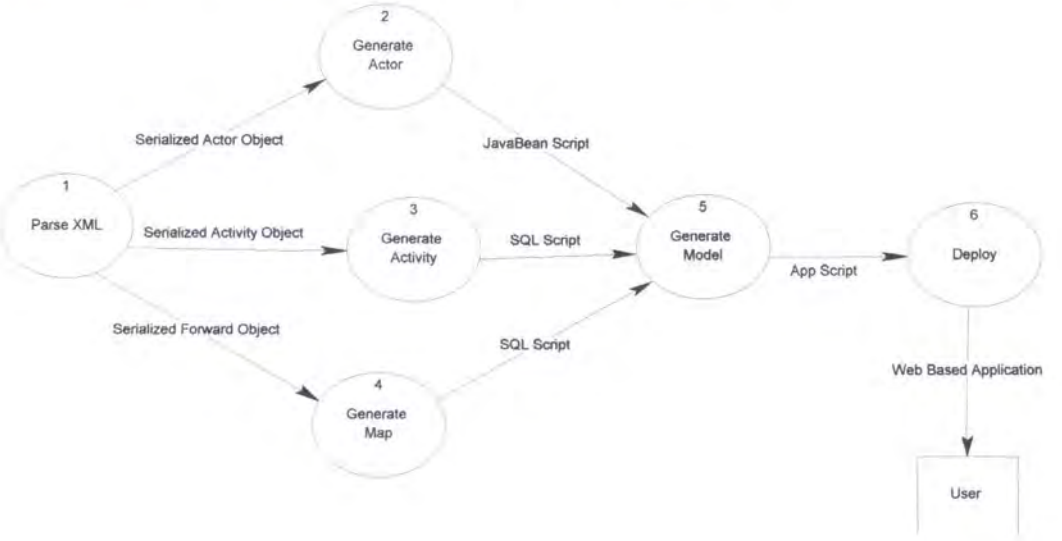
Aplikasi kemudian akan mengolah dan menggenerate source code halaman web, kemudian mendeploynya ke application server. Aplikasi yang dihasilkan nantinya berupa aplikasi J2EE berbasis web. Untuk selanjutnya, proses dalam system workflow designer dapat dijabarkan lebih lanjut seperti digambarkan dalam gambar 3.3 :



Gambar 3.3 DFD Level 1 Aplikasi Workflow Diagram

Di dalam DFD level 1 ini digambarkan proses lebih detail dari pengolahan input sehingga menghasilkan output berupa aplikasi yang berbasis web. Data inputan dari user dimasukkan melalui proses draw diagram, dimana di dalam proses ini, user menggambarkan workflow diagram dari aplikasi yang akan digenerate nantinya, termasuk di dalamnya aktor, entity, approval, activity dan juga action list. Setelah user selesai menggambarkan workflow ini, maka program akan menyimpan konfigurasi ini ke dalam file yang berformat XML (eXtensible Markup Language).

Setelah disimpan, untuk selanjutnya dilakukan proses generate aplikasi yang akan dijabarkan lebih lanjut dalam DFD level 2 pada gambar 3.4.



Gambar 3.4 DFD Level 2 Aplikasi Workflow Diagram

Gambar 3.5 menunjukkan rancangan format file xml yang akan digunakan untuk menampung informasi workflow yang akan dibangun.

```
<?xml version="1.0" encoding="utf-8"?>
<model>
<project>
<title></title>
<author></author>
<created> </created>
</project>

<aktors>
<aktor id="" name="" coordx="" coordy="">
<entities>
<entity>
<name> </name>
<type> </type>
</entity>
</entities>
</aktor>
</aktors>

<activities>
<activity id="" name="" coordx="" coordy="" type="">
<description></description>
<datafields>
```

```
<field form="">
<name></name>
<type></type>
</field>
</datafields>
<forward>
<type></type>
<dest></dest>
</forward>
</activity>
</activities>
</model>
```

Gambar 3.5 Script format XML

Gambar 3.6 menunjukkan DTD (Document Type Definition) yang dapat digunakan untuk mengecek validitas format file xml tersebut .

```
<?xml version='1.0' encoding='UTF-8'?>

<!ELEMENT dest (#PCDATA)>
<!ELEMENT forward (dest|type)*>
<!ATTLIST forward
    type CDATA #IMPLIED
    dest CDATA #IMPLIED
    name CDATA #IMPLIED
    id CDATA #IMPLIED
>

<!ELEMENT field (type|name)*>
<!ATTLIST field form CDATA #IMPLIED>
<!ELEMENT datafields (field)*>
<!ELEMENT description EMPTY>
<!ELEMENT activity (forward|datafields|description)*>
<!ATTLIST activity
    type CDATA #IMPLIED
    coordy CDATA #IMPLIED
    coordx CDATA #IMPLIED
    name CDATA #IMPLIED
    id CDATA #IMPLIED
  >
<!ELEMENT activities (activity)*>
<!ELEMENT type (#PCDATA)>
<!ELEMENT name (#PCDATA)>
<!ELEMENT entity (type|name)*>
<!ELEMENT entities (entity)*>
<!ELEMENT aktor (entities)*>
<!ATTLIST aktor
    coordy CDATA #IMPLIED
    coordx CDATA #IMPLIED
```

```
    name CDATA #IMPLIED
    id CDATA #IMPLIED
  >
<!ELEMENT aktors (aktor)*>
<!ELEMENT created (#PCDATA)>
<!ELEMENT author (#PCDATA)>
<!ELEMENT title (#PCDATA)>
<!ELEMENT project (created|author|title)*>
<!ELEMENT model (activities|aktors|project)*>
```

Gambar 3.6 Script DTD

Berikut adalah penjelasan dari dokumen xml yang terbentuk :

Tabel 3.2 Penjelasan Dokumen XML

`<?xml version="1.0" encoding="utf-8"?>`	Tipe format encoding dari dokumen XML yang dipergunakan.
`<model></model>`	Tag awal untuk menandakan bahwa file xml ini menggambarkan model workflow.

`<project>`
`  <title></title>`
`  <author></author>`
`  <created> </created>`
`</project>`
	Berisi informasi tentang project yang bersangkutan, termasuk di dalamnya judul project, penulis project, serta waktu akhir modifikasi project ini.

<aktors> <aktor id="" name="" coordx="" coordy="">> <entities> <entity> <name> </name> <type> </type> </entity> </entities> </aktor> </aktors>	Berisi informasi tentang aktor yang bersangkutan. Dalam tag **<aktor>** terdapat property yang berisi nomor identitas (*id*), nama aktor (*name*) koordinat x (*coordx*), dan koordinat y (*coordy*). Property *id* dan *name* digunakan sebagai pengenal aktor yang bersangkutan. Sedangkan property *coordx* dan *coordy* digunakan untuk petunjuk koordinat bagi diagram wokflow yang bersangkutan. Aktor juga memiliki entity-entity yang nantinya berperan dalam mapping business process di database. Nama entity dideskripsikan di dalam tag *<name> </name>*. Sedangkan tipe data dari entity tersebut dideskripsikan dalam tag *<type> </type>*. Tipe data yang dimaksud disini adalah tipe data dari java, bukan tipe data dari database.

<activities> <activity id="" name="" coordx="" coordy="" type=""><description></description> <datafields> <field form=""> <name></name> <type></type> </field> </datafields> <forward> <type></type> <dest></dest> </forward> </activity> </activities>	Berisi informasi tentang aktivitas yang terjadi. Tag *activity* memiliki property identitas (*id*), nama (*name*), koordinat x (*coordx*), koordinat y (*coordy*) dan tipe(*type*). Property *id* dan *name* digunakan sebagai pengenal activity yang bersangkutan. Property *coordx* dan *coordy* digunakan untuk petunjuk koordinat bagi diagram wokflow yang bersangkutan. Sedangkan property *type* menjelaskan tentang kemungkinan tipe activity yang ada. Terdapat 4 tipe kemungkinan tipe yang ada, yaitu : a. Operasi insert update delete review b. Operasi approval + review c. Operasi review d. Operasi otentikasi e. Operasi forward page Tag <description> berisi tentang deskripsi/ keterangan dari aktivitas yang bersangkutan. Aktivitas juga berhubungan dengan operasi database. Oleh karena itu terdapat juga deskripsi tentang field-field yang berpengaruh. Ini dideskripsikan dalam tag *<datafields>*. Di dalamnya terdapat tag *<field form="">* yang mendefinisikan tentang bentuk inputan dari form ke database. Nama entity dideskripsikan di dalam tag *<name> </name>*. Sedangkan tipe data dari entity tersebut dideskripsikan dalam tag *<type> </type>*. Tipe data yang dimaksud disini adalah tipe data dari java, bukan tipe data dari database.

Selanjutnya data dari file XML ini akan diparsing oleh workflow engine, membentuk elemen-elemen yang dibutuhkan sesuai dengan framework yang akan diimplementasikan, dalam hal ini yakni Struts FrameWork.

3.2.2 Mapping ke dalam Struts FrameWork

Dalam bagian ini akan dibahas bagian-bagian mana saja dari struts yang nantinya akan dimanipulasi oleh workflow designer.

3.2.2.1 Konfigurasi Struts-config.xml

Dalam Struts FrameWork, terdapat pemisahan yang jelas antara layer presentasi, dalam hal ini JSP dan HTML, dengan business logic, dalam hal ini bean-bean yang mengatur proses bisnis. Struts memiliki sebuah file konfigurasi yang mengatur mapping alur dari aplikasi web yang akan dijalankan. File ini bernama struts-config.xml. Gambar 3.7 menunjukkan contoh dari file struts-config.xml yang belum dikonfigurasi.

```
<?xml version="1.0" encoding="ISO-8859-1" ?>

<!DOCTYPE struts-config PUBLIC"-//Apache Software
Foundation//DTD Struts Configuration 1.1//EN
"http://jakarta.apache.org/struts/dtds/struts-
config_1_1.dtd">

<struts-config>

<!--  ========= Data Source Configuration -->
<!--
 <data-sources>
   <data-source>
     <set-property
        property="autoCommit"
        value="false"/>
     <set-property
        property="description"
```

```
        value="Example Data Source Configuration"/>
     <set-property
        property="driverClass"
        value="org.postgresql.Driver"/>
     <set-property
        property="maxCount"
        value="4"/>
     <set-property
        property="minCount"
        value="2"/>
     <set-property
        property="password"
        value="mypassword"/>
     <set-property
        property="url"
        value="jdbc:postgresql://localhost/mydatabase"/>
     <set-property
        property="user"
        value="myusername"/>
   </data-source>
 </data-sources>
-->

<!-- ============================ Form Bean Definitions -->
    <form-beans>
    <!-- sample form bean descriptor for an ActionForm
        <form-bean
            name="inputForm"
            type="app.InputForm"/>
    end sample -->
    <!-- sample form bean descriptor for a DynaActionForm
        <form-bean
            name="logonForm"
            type="org.apache.struts.action.DynaActionForm">
            <form-property
                name="username"
                type="java.lang.String"/>
            <form-property
                name="password"
                type="java.lang.String"/>
    end sample -->
    </form-beans>

<!-- ==================== Global Exception Definitions -->
    <global-exceptions>
        <!-- sample exception handler
        <exception
            key="expired.password"1
            type="app.ExpiredPasswordException"
            path="/changePassword.jsp"/>
        end sample -->
```

```
    </global-exceptions>

<!-- ================== Global Forward Definitions -->
    <global-forwards>
        <!-- Default forward to "Welcome" action -->
        <!-- Demonstrates using index.jsp to forward -->
        <forward
            name="welcome"
            path="/Welcome.do"/>
    </global-forwards>

<!-- ================== Action Mapping Definitions -->
    <action-mappings>
            <!-- Default "Welcome" action -->
            <!-- Forwards to Welcome.jsp -->
        <action
            path="/Welcome"
            type="org.apache.struts.actions.ForwardAction"
            parameter="/pages/Welcome.jsp"/>

    <!-- sample input and input submit actions -->
        <action
            path="/Input"
            type="org.apache.struts.actions.ForwardAction"
            parameter="/pages/Input.jsp"/>
        <action
            path="/InputSubmit"
            type="app.InputAction"
            name="inputForm"
            scope="request"
            validate="true"
            input="/pages/Input.jsp"/>

    </action-mappings>

<!-- ================ Message Resources Definitions -->
    <message-resources parameter="resources.application"/>

</struts-config>
```

Gambar 3.7 Script Struts-Config.xml

File konfigurasi tersebut dapat dibagi menjadi beberapa bagian, yaitu bagian *header*, konfigurasi data *source*, *form-beans definitions*, *global-exception definitions*, *global-forward definitions*, *action-mapping definitions*, serta *message-resource definitions.*

a. Bagian Header

```
<?xml version="1.0" encoding="ISO-8859-1" ?>

<!DOCTYPE struts-config PUBLIC"-//Apache Software
Foundation //DTD Struts Configuration 1.1//EN
"http://jakarta.apache.org/struts/dtds/struts-
config_1_1.dtd">
```

Gambar 3.8 Script Header Struts-Config.xml

Gambar 3.8 berisi informasi keterangan mengenai versi XML yang digunakan (versi 1.0), tipe encoding (ISO-8859-1), serta DTD yang digunakan (DTD dari http://jakarta.apache.org).

Konfigurasi data source

```
<!--  ========= Data Source Configuration -->
<!--
 <data-sources>
   <data-source>
     <set-property
        property="autoCommit"
        value="false"/>
     <set-property
        property="description"
        value="Example Data Source Configuration"/>
     <set-property
        property="driverClass"
        value="org.postgresql.Driver"/>
     <set-property
        property="maxCount"
        value="4"/>
     <set-property
        property="minCount"
        value="2"/>
     <set-property
        property="password"
        value="mypassword"/>
     <set-property
        property="url"
        value="jdbc:postgresql://localhost/mydatabase"/>
     <set-property
        property="user"
        value="myusername"/>
   </data-source>
 </data-sources>
-->
```

Gambar 3.9 Script Konfigurasi Data Source

Gambar 3.9 berisi konfigurasi datasource untuk connection pooling dari database. Biasanya jarang digunakan karena connection pooling sudah bisa dihandle oleh penggunaan bean-bean dalam aplikasi nantinya.

Form-beans definitions

```
<form-beans>
    <!-- sample form bean descriptor for an ActionForm
        <form-bean
            name="inputForm"
            type="app.InputForm"/>
    end sample -->
    <!-- sample form bean descriptor for a DynaActionForm
        <form-bean
            name="logonForm"
            type="org.apache.struts.action.DynaActionForm">
            <form-property
                name="username"
                type="java.lang.String"/>
            <form-property
                name="password"
                type="java.lang.String"/>
    end sample -->
</form-beans>
```

Gambar 3.10 Script Form-beans Definition

Gambar 3.10 berisi daftar form-form yang di*submit* oleh aplikasi nantinya. Bagian ini mutlak ada karena *struts* akan mengecek terlebih dahulu mapping di form ini, untuk kemudian meneruskan mapping menuju *action form* yang dituju. Deklarasi nama form berada pada *tag* pada gambar 3.11.

```
<form-bean name="logonForm" type="
org.apache.struts.action.DynaActionForm">
```

Gambar 3.11 Script Deklarasi nama form

Dari sintaks diatas dapat diketahui bahwa nama form yang akan disubmit adalah *logonForm* sedangkan yang menghandle inputan dari form tersebut adalah class *org.apache.struts.action.DynaActionForm*. Ini adalah setting default dari struts.

Class handler ini secara sederhana hanya berisi variable-variabel yang merupakan mapping dari inputan form, beserta metode setter dan getter untuk variable-variable tersebut. Class handler ini dapat dibuat dan dikompilasi secara manual. Sedangkan di sisi view yakni file JSP harus digunakan tag-tag khusus yang dimiliki struts agar nama form ini dapat dikenali. Sebelumnya di dalam file web.xml harus terlebih dahulu dideklarasikan tag library ini dengan sintaks pada gambar 3.12 :

```
<taglib>
<taglib-uri>struts-html</taglib-uri>
<taglib-location>
/WEB-INF/struts-html.tld
</taglib-location>
</taglib>
```

Gambar 3.12 Script Deklarasi tag library

Lalu kemudian di dalam setiap file JSP dideklarasikan tag pada gambar 3.13 :

```
<%@ taglib uri="struts-html" prefix="html" %>
```

Gambar 3.13 Script Deklarasi tag dalam JSP

Berikut ini adalah daftar tag-tag khusus yang dimiliki oleh struts :

Tabel 3.3 Tabel Tag yang dimiliki Struts

Struts Tag	HTML Tag
html:button	input type=button
html:cancel	input type=cancel
html:checkbox	input type=checkbox
html:file	input type=file
html:form	<form>
html:html	<html>
html:img	<img src='' >
html:link	<a href=''>

html:submit	input type=submit
html:text	input type=text
html:password	input type=password
html:base	<base href=''>
html:textarea	<textarea>
html:reset	input type=reset

Apabila tag-tag ini digunakan, maka struts nantinya akan mengenali tag-tag ini, kemudian mengarahkan mapping action yang bersangkutan sesuai yang ada di dalam struts-config.xml.

Global-exception definitions

```
<global-exceptions>
    <!-- sample exception handler
    <exception
        key="expired.password"
        type="app.ExpiredPasswordException"
        path="/changePassword.jsp"/>
    end sample -->
</global-exceptions>
```

Gambar 3.14 Global Exception Definitions

Gambar 3.14 berisi mapping global untuk halaman *exception*. Contoh diatas menggambarkan *action-mapping* yang terjadi ke halaman */changePassword.jsp* ketika terjadi suatu *error exception app.ExpiredPasswordException,* dan menampilkan isi dari *key expired.password* yang berasal dari *applicationresources.properties.*

Global-forward definitions

```
<global-forwards>
        <!-- Default forward to "Welcome" action -->
        <!-- Demonstrates using index.jsp to forward -->
        <forward
            name="welcome"
```

```
            path="/Welcome.do"/>
</global-forwards>
```

Gambar 3.15 Global forward Definitions

Gambar 3.15 berisi mapping forward yang akan dikenali oleh setiap context yang ada. Dalam contoh ini, terdapat *forward* bernama *welcome* yang akan diforward ke *context /welcome.do.*

Action-mapping definitions

```
<action-mappings>
<!-- Default "Welcome" action -->
<!-- Forwards to Welcome.jsp -->
<action path="/Welcome"
type="org.apache.struts.actions.ForwardAction"
parameter="/pages/Welcome.jsp"/>

<!-- sample input and input submit actions -->
<action path="/Input"
type="org.apache.struts.actions.ForwardAction"
parameter="/pages/Input.jsp"/>

<action  path="/InputSubmit"
         type="app.InputAction"
         name="inputForm"
         scope="request"
         validate="true"
         input="/pages/Input.jsp"/>

</action-mappings>
```

Gambar 3.16 Action Mapping Definitions

Gambar 3.16 berisi mapping untuk nama konteks yang telah ditentukan. Berbeda dengan global-forward yang dapat dikenali di semua konteks, action-mapping disini hanya akan dikenali dalam satu konteks tersebut. Dari contoh dapat dilihat bahwa terdapat sebuah form bernama *inputForm*, yang ketika dilakukan proses submit, akan diarahkan menuju */InputSubmit*. Sedangkan class yang menangani adalah *class app.InputAction*, sedangkan halaman JSP dimana proses submit

terjadi adalah */pages/input.jsp*. Halaman input ini juga melakukan validasi, ditandai dengan adanya sintaks validate=true.

Message-resource definitions

```
<message-resources parameter="resources.application"/>
```

Gambar 3.17 Script Message resource definitions

Gambar 3.17 berisi mapping untuk kumpulan pesan-pesan yang dapat dipilih, untuk ditampilkan dengan berbagai macam setting local. Bagian ini dapat memiliki banyak duplikasi untuk setting yang berbeda pula. Misalkan terdapat file resources.application yang berisi setting untuk pesan-pesan dalam bahasa inggris, kemudian dapat juga didefinisikan file resource untuk seting pesan-pesan dalam bahasa lain.

3.2.2.2 Konfigurasi web.xml

Setting lainnya terdapat pada file bernama web.xml. File ini berfungsi untuk mapping servlet-servlet yang ada dalam suatu aplikasi web. Selain itu, file ini juga berisi konfigurasi tentang dimana tersimpan konfigurasi untuk struts, beserta tag-tag bawaannya. Gambar 3.18 menunjukkan isi dari file web.xml.

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<!DOCTYPE web-app
  PUBLIC "-//Sun Microsystems, Inc.//DTD Web Application
2.2//EN"
  "http://java.sun.com/j2ee/dtds/web-app_2_2.dtd">

<web-app>

<!-- Standard Action Servlet Configuration (with debugging)
-->
  <servlet>
    <servlet-name>action</servlet-name>
    <servlet-
class>org.apache.struts.action.ActionServlet</servlet-
class>
```

```
    <init-param>
      <param-name>config</param-name>
      <param-value>/WEB-INF/struts-config.xml</param-value>
    </init-param>

    <init-param>
      <param-name>application</param-name>
      <param-value>ApplicationResources</param-value>
    </init-param>

    <init-param>
      <param-name>debug</param-name>
      <param-value>2</param-value>
    </init-param>
    <init-param>
      <param-name>detail</param-name>
      <param-value>2</param-value>
    </init-param>
    <load-on-startup>2</load-on-startup>
  </servlet>

  <!-- Standard Action Servlet Mapping -->
  <servlet-mapping>
    <servlet-name>action</servlet-name>
    <url-pattern>*.do</url-pattern>
  </servlet-mapping>

  <!-- The Usual Welcome File List -->
  <welcome-file-list>
    <welcome-file>login.jsp</welcome-file>
  </welcome-file-list>

  <!-- Struts Tag Library Descriptors -->
  <taglib>
    <taglib-uri>/tags/struts-bean</taglib-uri>

    <taglib-location>/WEB-INF/struts-bean.tld</taglib-
location>
  </taglib>

  <taglib>
    <taglib-uri>/tags/struts-html</taglib-uri>
    <taglib-location>/WEB-INF/struts-html.tld</taglib-
location>
  </taglib>

  <taglib>
    <taglib-uri>/tags/struts-logic</taglib-uri>
    <taglib-location>/WEB-INF/struts-logic.tld</taglib-
location>
```

```
  </taglib>

  <taglib>
    <taglib-uri>/tags/struts-nested</taglib-uri>
    <taglib-location>/WEB-INF/struts-nested.tld</taglib-
location>
  </taglib>

</web-app>
```

Gambar 3.18 Script web.xml

File konfigurasi web.xml dapat dibagi menjadi dua bagian, yaitu *standard action servlet configuration*, *standard action servlet mapping* dan *struts tag library descriptors*.

a. Standard action servlet configuration

```
<servlet>
    <servlet-name>action</servlet-name>
    <servlet-class>org.apache.struts.action.ActionServlet
</servlet-class>
```

Gambar 3.19 Script standart action servlet configuration

Gambar 3.19 berisi keterangan tentang servlet yang bertugas sebagai *controller* sesuai dengan MVC model 2 dimana terdapat satu buah servlet yang bertugas sebagai controller untuk suatu aplikasi. Servlet ini bernama *action*, dan jika nama servlet ini dipanggil, maka akan merefer ke class *org.apache.struts.action.ActionServlet* yang telah dihandle oleh struts.

```
<init-param>
    <param-name>config</param-name>
    <param-value>/WEB-INF/struts-config.xml</param-value>
</init-param>
```

Gambar 3.20 Script standart action servlet configuration

Gambar 3.20 berisi parameter agar webserver yang bersangkutan, yakni tomcat, mencari file configurasi untuk struts yang terletak di /WEB-INF/struts-config.xml.

```
<init-param>
      <param-name>application</param-name>
      <param-value>ApplicationResources</param-value>
</init-param>
```

Gambar 3.21 Bagian init-param script struts-config.xml

Gambar 3.21 berisi parameter agar webserver yang bersangkutan mencari file konfigurasi untuk aplikasi yang bernama ApplicationResources.

```
    <init-param>
      <param-name>debug</param-name>
      <param-value>2</param-value>
    </init-param>
    <init-param>
      <param-name>detail</param-name>
      <param-value>2</param-value>
    </init-param>
    <load-on-startup>2</load-on-startup>
```

Gambar 3.22 Bagian init-param script web.xml

Gambar 3.22 berisi setting untuk parameter debug yang bernilai 2, parameter detail yang bernilai 2, serta load on startup yang bernilai 2.

b. Standard action servlet mapping

```
<servlet-mapping>
    <servlet-name>action</servlet-name>
    <url-pattern>*.do</url-pattern>
</servlet-mapping>
```

Gambar 3.23 Script standart action servlet mapping

Gambar 3.23 menjelaskan lebih detail mapping servlet pada bagian sebelumnya. Dalam contoh dijelaskan bahwa servlet dengan nama action dapat diakses via URL dengan mengetikkan akhiran .do. Akibatnya, dimasukkan nama servlet berakhiran .do yang sesuai dengan action mapping pada struts-config.xml , maka otomatis servlet action akan diload.

```
<!-- The Usual Welcome File List -->
<welcome-file-list>
    <welcome-file>login.jsp</welcome-file>
</welcome-file-list>
```

Gambar 3.24 Bagian welcome-file-list script web.xml

Gambar 3.24 menujukkan halaman apa yang pertama kali harus ditampilkan ketika user baru mengetik nama folder yang bersangkutan. Fungsinya sama seperti index.html atau index.php yang biasanya secara otomatis di load kita suatu situs baru pertama kali terbuka. Dalam contoh diatas, ketika dibuka, maka yang pertama kali akan muncul adalah halaman login.jsp

c. Struts tag library descriptors

```
<taglib>
 <taglib-uri>/tags/struts-bean</taglib-uri>
 <taglib-location>/WEB-INF/struts-bean.tld</taglib-
location>
 </taglib>

 <taglib>
    <taglib-uri>/tags/struts-html</taglib-uri>
    <taglib-location>/WEB-INF/struts-html.tld</taglib-
location>
  </taglib>

  <taglib>
    <taglib-uri>/tags/struts-logic</taglib-uri>
    <taglib-location>/WEB-INF/struts-logic.tld</taglib-
location>
  </taglib>
```

Gambar 3.25 Script struts tag library description

Gambar 3.25 menunjukkan daftar deklarasi tag-tag yang dikenal oleh struts. Dalam contoh diatas, tag-tag yang disupport oleh struts antara lain :

Tag struts-bean

Tabel 3.4 Tag struts-bean

Nama Tag	Keterangan
bean:define	Digunakan untuk menduplikasikan property suatu bean dari scope tertentu, ke scope yang lain.

bean:include	Digunakan untuk mengikutsertakan kembali kumpulan skrip yang dapat digunakan ulang, ke dalam halaman yang lain.
bean:message	Digunakan untuk mengambil nilai yang sesuai berdasarkan key yang ada pada applicationresource untuk ditampilkan ke halaman JSP.
bean:page	Digunakan untuk mendefinisikan suatu objek implicit dari JSP sebagai variable skrip.
bean:resource	Digunakan untuk mengambil nilai mentah dari suatu resource yang ada dalam konteks direktori dari suatu aplikasi web.
bean:size	Digunakan untuk menghitung jumlah item yang tersimpan di dalam array, collection, atau hashmap, lalu menyimpannya ke dalam variable yang dideklarasikan oleh attribute *id*.
bean:struts	Digunakan untuk menduplikasikan objek struts, ke dalam variable dengan ruang lingkup akses page, dan scope.
bean:write	Digunakan untuk mengambil nilai dari suatu bean.
bean:cookie, bean:header, and bean:parameter	Digunakan untuk mengambil nilai dari cookies, request header, serta request parameter.

Tag struts-html

Tabel 3.5 Tag struts-html

Nama Tag	Keterangan
html:button	Digunakan untuk menghasilkan objek button.
html:cancel	Digunakan untuk menghasilkan objek button cancel.
html:checkbox	Digunakan untuk menghasilkan objek checkbox.

html:file	Digunakan untuk menghasilkan objek input type file.
html:form	Digunakan untuk menghasilkan objek form.
html:html	Digunakan untuk menghasilkan objek tag html.
html:img	Digunakan untuk menghasilkan objek image.
html:link	Digunakan untuk menghasilkan objek hyperlink.
html:submit	Digunakan untuk menghasilkan objek button submit.
html:text	Digunakan untuk menghasilkan objek input textbox.
html:password	Digunakan untuk menghasilkan objek input password.
html:base	Digunakan untuk menghasilkan objek base html.
html:textarea	Digunakan untuk menghasilkan objek input textarea.
html:reset	Digunakan untuk menghasilkan objek button reset.

Tag struts-logic

Tabel 3.6 Tag struts-logic

Nama Tag	Keterangan
logic:empty, logic:notEmpty	Digunakan untuk memeriksa nilai dari suatu variable, apakah null, string kosong, apakah bertipe collection, atau hashmap.
logic:equal, logic:notEqual, logic:lessThan, logic:greaterThan, logic:lessEqual, and logic:greaterEqual	Digunakan untuk memeriksa nilai dari 2 variable, apakah sama dengan, lebih dari, kurang dari, dst.

logic:forward	Digunakan untuk melakukan forwarding halaman, kecuali sebelumnya telah didefinisikan di ActionForward dalam global-forward di file konfigurasi struts (struts-config.xml).
logic:redirect	Digunakan juga untuk melakukan forwarding halaman, tapi analoginya sama seperti response.sendRedirect()
logic:iterate	Digunakan untuk melakukan iterasi melalui objek collection, enumerator, iterator, hashmap, maupun array. Iterator ini melakukan pemeriksaan setiap isi dari object collection tersebut.
logic:match and logic:notMatch	The logic:match and logic:notMatch tags check to see if two strings have equal parts at the start of the string, at the end the string, or if any parts of the strings are equal.
logic:present and logic:notPresent	The logic:present and logic:notPresent tags check to see if headers, request parameters, cookies, JavaBeans, or JavaBean properties are present and not equal to null.

Tag struts-nested

Tabel 3.7 Tag struts-nested

Tag Name	Description
nest	Mendefinisikan level nesting baru untuk referensi dari tag child berikutnya.

writeNesting	Membuat variable scripting baru dari level nesting yang sekarang.
root	Memulai hierarki form baru.
define	Nested Extension – Mendefinisikan variable scripting baru berdasarkan nilai dari property bean.
message	Nested Extension – Mengatur kembali output dari message yang berasal dari applicationResources
size	Nested Extension – Mendefinisikan bean yang mengandung jumlah elemen dari suatu collection atau hashmap.
write	Nested Extension – Mengatur kembali nilai dari proerty bean yang telah diberikan, ke JSPWriter.
checkbox	Nested Extension - Mengatur kembali property dari input checkbox.
errors	Nested Extension – Menampilkan secara kondisional, jumpulan pesan kesalahan yang ada.
file	Nested Extension - Mengatur kembali input type select file.
form	Nested Extension – Mendefinisikan input form
hidden	Nested Extension - Mengatur kembali property input bertipe hidden field.
image	Nested Extension - Mengatur kembali property input bertipe image.
img	Nested Extension - Mengatur kembali property tag img.
link	Nested Extension - Mengatur kembali property hyperlink dari tag html.

Dari tag-tag struts inilah, tampilan template halaman web yang akan digenerate nantinya direnderr, sehingga dihasilkan file-file yang dibutuhkan,

sesuai dengan implementasi MVC Model 2 dengan struts framework. Berikut selengkapnya file-file yang dibutuhkan oleh struts dan yang akan digenerate oleh workflow designer ini :

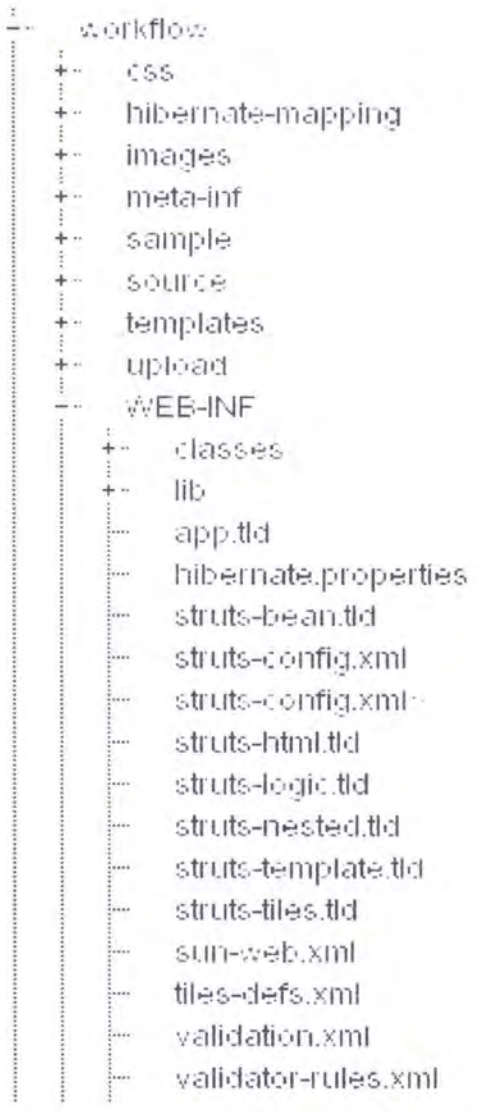
Tabel 3.8 Daftar file yang akan digenerate oleh workflow designer

	Nama File	Tipe File	Direktory	Keterangan
	*.jsp	Java Server Pages	APP_BASE/	Direnderdari file template.
	*Form.java	JavaBeans	APP_BASE/WEB-INF/classes	Digenerate dengan komponen digester yang memparsing file XML yang digenerate oleh workflow designer. Digenerate dari object activity bertipe 1 atau 3.
	*Action.java	JavaBeans	APP_BASE/WEB-INF/classes	Digenerate dengan komponen digester yang memparsing file XML yang digenerate oleh workflow designer.

	struts-bean.tld, struts-html.tld, struts-logic.tld, struts-template.tld, struts-nested.tld, struts-tiles.tld, validation.xml, validator-rules.xml	Definisi tag-tag html, bean, logic, nested, tiles, dan validator yang merupakan bawaan dari struts.	APP_BASE/WEB-INF/	Dikopikan melalui generate script dari apache-ant.

	commons-beanutils.jar, commons-collections.jar, commons-dbcp.jar, commons-digester.jar, commons-fileupload.jar, commons-lang.jar, commons-logging.jar, commons-pool.jar, commons-resources.jar, commons-validator.jar, jakarta-oro.jar, struts.jar	File jar yang dibutuhkan oleh struts.	APP_BASE/WEB-INF/lib	Dikopikan melalui generate script dari apache-ant.
	*.hbm.xml	Hibernate configuration script untuk mapping dari object java, ke penyimpanan secara fisik dalam table.	APP_BASE/WEB-INF/classes	Digenerate oleh aplikasi workflow designer.

Gambar 3.26 menunjukkan rancangan struktur direktori hasil generate dari aplikasi workflow designer ini :



Gambar 3.26 Direktori hasil generate

3.2.2.3 Konfigurasi build.xml

Berikut adalah rancangan file build.xml yang akan degenerate oleh workflow designer untuk melakukan setiap aplikasi yang akan dibuat :

```
<?xml version="1.0" encoding="UTF-8"?>
<project basedir="." default="all" name="myWorkflow">
 <target name="init">

  <property location="WEBINF/classes" name="classes.dir"/>
  <property location="genDir/source" name="src.dir"/>
  <property name="project.name"
value="${ant.project.name}"/>
 </target>
<target depends="init" name="compile">
   <mkdir dir="${classes.dir}"/>
 <javac debug="true" deprecation="true"
destdir="${classes.dir}" srcdir="${src.dir}">
 </javac>
</target>
<target depends="init,jar" description="Build everything."
name="all"/>
<target depends="init" description="Clean all build
products." name="clean">
   <delete dir="${classes.dir}"/>
   </target>
</project>
```

Gambar 3.27 Script Konfigurasi build.xml

3.2.3 Desain class-class

Dalam implementasi, dibutuhkan dua buah object untuk menangani aktor dan activity. Dua buah object inilah yang nantinya akan banyak berpengaruh di dalam proses generate file yang dibutuhkan sesuai dengan framework struts yang digunakan.

Class Aktor

Class aktor mengimplementasikan sebuah object dengan property-property sebagai berikut :

Tabel 3.9 Daftar property dan kegunaan dari class Aktor

Nama Property	Kegunaan
projectName	Nama project yang bersangkutan
authorName	Nama pembuat project
type	Tipe atribut-atribut yang dimiliki oleh aktor yang bersangkutan

name	Nama atribut-atribut yang dimiliki oleh aktor yang bersangkutan
objName	Nama object aktor yang bersangkutan
aktorId	Nomor identitas aktor yang bersangkutan

Gambar 3.28 menunjukkan implementasi dari class aktor beserta method getter dan setter untuk mengakses property dari class ini.

```
public class WFActorProto
    {

        private String projectName;
        private String authorName;
        private List type = new ArrayList();
        private List name = new ArrayList();
        private String objname;
        private int aktorId;

        public void setName(String newName)
        {
            name.add(newName);
        }


        public List getName()
        {
            return name;
        }

        public void setType(String newType)
        {
            type.add(newType);
        }
        public List getType()
        {
            return type;
        }

        public void setObjName(String newObjName)
        {
            objname = newObjName;
        }
        public String getObjName()
        {
            return objname;
        }

        public java.lang.String getAuthorName() {
            return authorName;
        }
```

```
    public void setAuthorName(java.lang.String authorName) {
        this.authorName = authorName;
    }

    public java.lang.String getProjectName() {
        return projectName;
    }


    public void setProjectName(java.lang.String projectName) {
        this.projectName = projectName;
    }


    public int getAktorId() {
        return aktorId;
    }

}
```

Gambar 3.28 Desain class aktor

Class Activity

Class activity mengimplementasikan sebuah object dengan property-property sebagai berikut :

Tabel 3.10 Daftar property dan kegunaan dari class Activity

Nama Property	Kegunaan
projectName	Nama project yang bersangkutan
authorName	Nama pembuat project
objType	Tipe Activity, terdapat 5 jenis tipe, yaitu : - aktivitas manipulasi database - aktivitas approval - aktivitas otentifikasi - aktivitas forwarder - aktivitas logout
objName	Nama aktivitas yang bersangkutan
fieldType	Tipe data untuk field yang terdapat dalam object activity
fieldName	Nama field yang terdapat dalam object activity.
Desc	Deskripsi / keterangan mengenai activity yang bersangkutan
aktorId	Identitas aktor pemilik activity yang bersangkutan
Relation	Hubungan relasi dengan activity yang lain, digunakan dalam proses approval.

myId	Nomor identitas dari object activity yang bersangkutan

Gambar 3.29 menunjukkan implementasi dari class activity beserta method getter dan setter untuk mengakses property dari class ini.

```
public class WFActivityProto
    {
        private String projectName;
        private String authorName;
        private List fieldType = new ArrayList();
        private List fieldName = new ArrayList();
        private String objname;
        private String objtype;
        private List formType = new ArrayList();
        private List formId = new ArrayList();
        private String Desc;
        private int aktorId;
        private String relation;
        private int myId;

        public void setProjectName(String newProjName)
        {
            projectName = newProjName;
        }
        public String getProjectName()
        {
            return projectName;
        }

        public void setAuthorName(String newAuthName)
        {
            authorName = newAuthName;
        }

        public String getAuthorName()
        {
            return authorName;
        }

        public void setFieldName(String newFieldName)
        {
            fieldName.add(newFieldName);
        }

        public List getFieldName()
        {
            return fieldName;
        }

        public void setFieldType(String newFieldType)
        {
```

```
        fieldType.add(newFieldType);
    }
    public List getFieldType()
    {
        return fieldType;
    }


    public void setName(String newName)
    {
        name.add(newName);
    }

    public List getName()
    {
        return name;
    }

    public void setType(String newType)
    {
        type.add(newType);
    }
    public List getType()
    {
        return type;
    }

    public void setDesc(String newDesc)
    {
        Desc = newDesc;
    }
    public String getDesc()
    {
        return Desc;
    }

    public void setObjName(String newObjName)
    {
        objname = newObjName;
    }

    public String getObjName()
    {
        return objname;
    }

    public void setObjType(String newObjType)
    {
        objtype = newObjType;
    }

    public String getObjType()
    {
        return objtype;
    }
```

```
    public List getFormId() {
        return formId;
    }

    public void setFormId(String formId)
    {
        this.formId.add(formId);
    }

    public List getFormType() {
        return formType;
    }

    public void setFormType(String formType)
    {
        this.formType.add(formType);
    }

    public int getAktorId() {
        return aktorId;
    }

    public void setAktorId(int actId)
    {
        this.aktorId = aktorId;
    }

    public java.lang.String getRelation() {
        return relation;
    }

    public void setRelation(java.lang.String relation)
    {
        this.relation = relation;
    }

    public int getMyId() {
        return myId;
    }

    public void setMyId(int myId) {
        this.myId = myId;
    }

}
```

Gambar 3.29 Desain class activity

Class Forwarder

.Sedangkan untuk mengimplementasi kebutuhan alur aplikasi sesuai dengan alur dalam workflow maka dibutuhkan juga sebuah object yang berfungsi menyimpan proses mapping, dalam hal ini alur perjalanan aplikasi yang akan

dirancang nantinya. Tabel 3.11 menjelaskan tentang property-property yang dimiliki class forwarder ini.

Tabel 3.11 Daftar property dan kegunaan dari class forwarder

Nama Property	Kegunaan
Id	Nomor identitas object forwarder
srcId	Nomor identitas object domain
srcType	Tipe object domain
destId	Nomor identitas object tujuan
destType	Tipe object tujuan
Fieldname	Nama field yang terdapat dalam object activity.
Desc	Deskripsi / keterangan mengenai activity yang bersangkutan
aktorId	Identitas aktor pemilik activity yang bersangkutan
type	Tipe object forwarder yang bersangkutan

Gambar 3.30 menunjukkan implementasi dari class forwarder beserta method getter dan setter untuk mengakses property dari class ini

```
import java.util.*;

public class WFForwarder {

    private int id;
    private int srcId;
    private String srcType;
    private int destId;
    private String destType;
    private String type;
    /** Creates a new instance of WFForwarder */
    public WFForwarder() {
    }

    public int getSrcId() {
        return srcId;
    }

    public int getDestId() {
        return destId;
    }

    public java.lang.String getDestType() {
        return destType;
    }
```

BAB IV
IMPLEMENTASI PERANGKAT LUNAK

BAB IV

IMPLEMENTASI PERANGKAT LUNAK

Bab ini membahas tentang proses pembuatan perangkat lunak sesuai dengan perencanaan pada bab sebelumnya. Implementasi akan dibagi menjadi 2 bagian, yaitu implementasi bagian inti aplikasi (*workflow generator engine*), dan implementasi bagian workflow editor.

4.1 Bagian inti aplikasi

Di bagian ini diimplementasikan proses generate file-file yang dibutuhkan oleh struts framework, dalam hal ini file-file JSP beserta bean-bean dan file konfigurasinya. Alur implementasi bagian utama *workflow generator engine* ini dapat digambarkan dalam gambar 4.1 dalam format pseudocode.

4.1.1 Class utama workflow generator

```
// Buat objek untuk parser
// Set Property yang dibutuhkan untuk objek parser
// Mulai proses parsing file XML

 IF bertemu "<model>" THEN
   CreateObject WFGenerator
 IF bertemu "<model/aktors/aktor>" THEN
   Begin
    CreateObject WFAktor
    Set Property Aktor_name
    Set Property Aktor_id
    IF bertemu "<model/aktors/aktor/entities/entity/name>" THEN
     Set Property EntityName
    IF bertemu "<model/project/title>" THEN
     Set Property ProjectName
    IF bertemu "<model/project/author>" THEN
     Set Property ProjectAuthor
    IF bertemu "<model/aktors/aktor/entities/entity/type>" THEN
     Set Property EntityType
   End

 IF bertemu "<model/activities/activity>" THEN
```

```
import java.io.IOException;
import org.apache.commons.digester.Digester;
import org.xml.sax.SAXException;
import java.io.*;
import java.util.*;
import java.util.List;

Begin
    CreateObject WFActivity
    SetProperty ActivityID
    SetProperty ActivityName
    SetProperty ActivityType
    SetProperty AktorOwner
    SetProperty ActivityRelation
    IF bertemu "<model/activities/activity/description>"
     SetProperty ActivityDesc
    IF bertemu "<model/activities/activity/datafields/field/id>"
     SetProperty ActivityFormID
    IF bertemu
   "<model/activities/activity/datafields/field/name>"
     SetProperty ActivityFormName
    IF bertemu
   "<model/activities/activity/datafields/field/type>"
    IF bertemu "<model/activities/activity/forward/type>"
     SetProperty ActivityForwardType
    IF bertemu "<model/activities/activity/forward/dest>"
     SetProperty ActivityForwardDest
   End
   IF Parsing selesai THEN
   Begin
    CreateAktor
    CreateActivity
   End
```

Gambar 4.1 Pseudocode Workflow Generator Engine

Workflow Generator Engine ini nantinya akan menerima parameter berupa file XML yang digenerate oleh workflow editor. Bagian workflow generator engine tersebut diimplementasikan dalam metode `createAktor()` dan `createActivity()`. File-file konfigurasi dari workflow editor yang berformat XML akan diparsing dan jika bertemu dengan tag-tag yang telah didefinisikan, maka akan dijalankan suatu fungsi. Proses parsing dan *function dispatcher* ini dilakukan oleh komponen *digester* dan SAX parser yang telah didefinisikan sebelumnya. Tabel 4.1 menjelaskan sebagian fungsi dari komponen digester yang digunakan dalam bagian ini.

Tabel 4.1 Daftar fungsi object digester dan kegunaannya

Nama Fungsi	Kegunaan
setValidating	Memerintahkan object digester agar melakukan validasi terlebih dahulu terhadap input file XML yang akan diterima. Menerima parameter berupa Boolean true atau false.
addObjectCreate	Memerintahkan pembuatan object tertentu apabila bertemu dengan tag XML tertentu. Menerima parameter berupa tag XML yang akan dicari dan nama class yang akan dijadikan object.
addSetProperties	Mengeset nilai property milik object tertentu yang telah dibuat sebelumnya dalam tag addObjectCreate. Nilai property ini terdapat dalam property tag XML.
addCallMethod	Memerintahkan untuk memanggil suatu fungsi ketika menemui tag XML tertentu. Menerima parameter berupa tag XML yang dicari, serta fungsi yang akan dijalankan bila tag tersebut ditemukan.
addSetNext	Memerintahkan untuk memanggil suatu fungsi apabila tag XML tertentu ditemui tag XML penutup yang telah ditemukan. Menerima parameter berupa tag XML yang akan dicari, serta fungsi yang akan dijalankan bila tag tersebut ditemukan.

4.1.1.1 Method createActor()

Method ini bertugas untuk menggenerate object aktor, kemudian menerjemahkannya dengan menggenerate file-file yang dibutuhkan dalam implementasi framework. Gambar 4.2 menjelaskan pseudocode untuk algoritma dari metode `createActor()`.

```
void createActor(parameter input bertipe object Aktor)
  // tambahkan object ke List Aktor
  // buka file untuk streaming hasil generate
  // persiapkan identitas untuk file hasil streaming
  // generate file untuk mapping entity object ke database
  // generate file untuk class action
  // tutup file hasil streaming
```

Gambar 4.2 Pseudocode untuk metode CreateAktor

Dari metode createAktor ini nantinya akan didapatkan dua buah file, yaitu file untuk action class yang berekstensi .java, dan file mapping entity-entity yang ada dalam object aktor untuk hibernate berekstensi .hbm.xml. Kedua file ini akan diletakkan di direktori kumpulan hasil generate.

4.1.1.2 Method createActivity()

Method ini bertugas untuk menggenerate object activity, kemudian menerjemahkannya dengan menggenerate file-file yang dibutuhkan dalam implementasi framework. Gambar 4.2 menjelaskan pseudocode untuk algoritma dari metode *createActivity()*..

```
void createAcivity(parameter input bertipe object Activity)
   // tambahkan object ke List Activity

   SWITCH Property tipe Object
   CASE 1: // Operasi Manipulasi Database
   // buka file untuk streaming hasil generate
   // persiapkan identitas untuk file hasil streaming
   // generate file hbm.xml nya
   // generate file input JSP
   // generate file update dan delete JSP
   // generate classform struts
   // generate classaction struts

   CASE 2: // Operasi Approval
   // buka file untuk streaming hasil generate
   // persiapkan identitas untuk file hasil streaming
   // generate approval JSP
   // generate classform Struts
   // generate classaction Struts

   CASE 3: // Operasi Otentifikasi
   // buka file untuk streaming hasil generate
   // persiapkan identitas untuk file hasil streaming
   // generate Login JSP
   // generate file hbm.xml nya
   // generate classform Struts
   // generate classaction Struts

   CASE 4: // Operasi FORWARD PAGE
   // buka file untuk streaming hasil generate
   // persiapkan identitas untuk file hasil streaming
   // ambil struktur forward dari file xml
   // generate forward JSP
```

```
CASE 5 :// Operasi LOGOUT
// buka file untuk streaming hasil generate
// persiapkan identitas untuk file hasil streaming
// generate classaction Struts

CASE 6 :// Operasi VIEW
// buka file untuk streaming hasil generate
// persiapkan identitas untuk file hasil streaming
// generate view JSP
// generate classaction Struts
```

Gambar 4.3 Pseudocode untuk metode CreateActivity

Dari metode createActivity ini nantinya akan didapatkan table yang field-fieldnya sesuai dengan input yang dimasukkan oleh user pada bagian property. Secara default terdapat 2 buah field yang tergenerate. Tabel 4.2 menjelaskan tentang hasil field yang digenerate secara otomatis nantinya.

Tabel 4.2 Daftar field yang digenerate

Nama	Tipe Data	Keterangan
Username	String	Berisi field untuk username pada aktivitas login.
Password	String	Berisi field untuk password pada aktivitas login.
Custom Field dari User	String, Int, Double	Tipe data yang didukung adalah tipe data yang dimiliki oleh Java

4.1.1.3 Method createMap()

Method ini bertugas untuk memanipulasi table mapList yang berisi pemetaan daftar hak disposisi yang mungkin dimiliki oleh tiap *workflow participant*.. Pseudocode untuk method ini terdapat dalam gambar 4.6.

```
public void createMap(String tempFile,String outFile) {
      // buat instance dari class helper untuk manipulasi template
      // buat deklarasi object yang menampung daftar action list
      // buat deklarasi object yang menampung forwarder list
      // lakukan iterasi sebanyak jumlah action List bertipe Form
      // parse sintaks {FORM_BEAN}, ganti dengan sintaks mapping
```

```
// lakukan iterasi sebanyak jumlah action List bertipe Action
// parse sintaks {ACTION_MAPPING}, ganti dengan sintaks
mapping
// simpan dan tutup file tujuan
}
```

Gambar 4.4 Pseudocode untuk method doStrutsConfig()

Di dalam method ini digunakan sebuah class helper yang membantu dalam hal parsing file template dan generate file tujuan. Penjelasan lebih lanjut mengenai class helper ini ada pada bagian 4.1.2.

4.1.2 Class Helper

Selain class-class utama yang telah disebutkan, terdapat juga class-class helper yang bertugas mempermudah class utama.

4.1.1.4 Class WFBuildFromTemplate()

Class ini digunakan untuk melakukan parsing file template untuk kemudian dikonfigurasi menghasilkan file seperti yang diinginkan. Class ini bekerja dengan cara mencari sintaks yang telah dispesifikasikan, kemudian mengganti sintaks tersebut dengan sintaks yang diinginkan. Gambar 4.7 menunjukkan pseudocode dari class WFBuildFromTemplate().

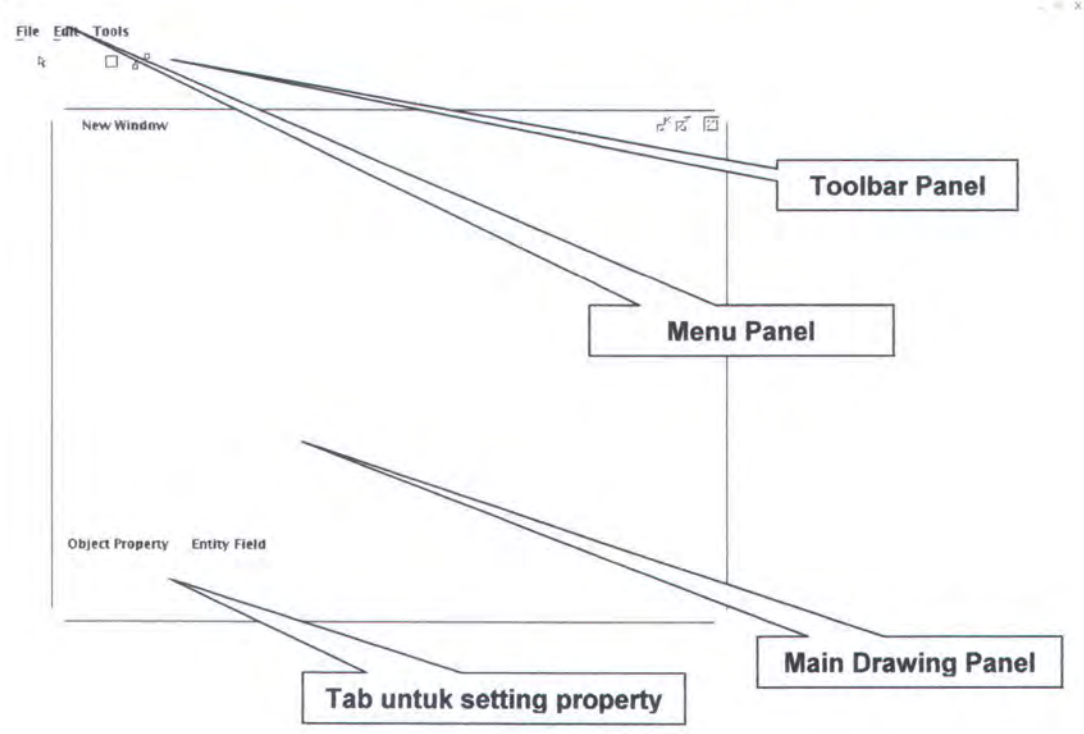
```
public class WFBuildFromTemplate {
private String template;
private String output;
private String resultTemp;
String strInput;
byte[] pInput;

public WFBuildFromTemplate(String tempFile, String outFile)
public void copy(String srcFile, String dstFile) throws IOException
public void parse(String strSearch,String strReplace,int sisa)
public void parse(String strSearch, String strReplace)
public final byte[] ReadFile(String strFile) throws IOException
public final void WriteFile(String strFile, byte[] pData) throws
IOException
```

Gambar 4.5 Pseudocode Deklarasi class WFBuildFromTemplate()

4.2 Bagian workflow editor

Di bagian ini akan dijelaskan tentang implementasi workflow editor yang digunakan untuk menggambarkan proses workflow yang bersangkutan. Dalam pembuatan Graphical User Interface dari workflow editor ini digunakan library *jgraph* yang telah menyediakan library-library yang mendukung fungsi-fungsi grafik. Gambar 4.8 menunjukkan rancangan tampilan awal dari workflow editor ini.



Gambar 4.6 Rancangan tampilan workflow editor

4.2.1 Toolbar Panel

Bagian object toolbar berisi object-object yang nantinya digunakan untuk menggambarkan proses workflow. Di dalam toolbar ini berisi beberapa object yang nantinya akan ditaruh diatas main drawing panel, yaitu : Object aktor, object activity, object connector, dan object cursor.

4.2.1.1 Object Aktor

Object ini nantinya digunakan untuk menggambarkan object aktor dalam workflow. Object aktor ini digambarkan dengan icon seperti pada gambar 4.7. Object aktor ini nantinya akan memiliki property-property seperti id, dan nama. Selain itu, object ini juga dapat memiliki entity field. Property serta entity tadi dimasukkan dalam tab setting property yang terletak di bagian paling bawah dari child window yang muncul.

Gambar 4.7 Icon untuk object Aktor

Bila nantinya toolbar bagian object aktor ini diaktifkan, lalu di taruh dalam *main drawing panel*, maka object ini akan tampil di bagian dimana mouse diclick, seperti ditunjukkan dalam gambar 4.8. Secara otomatis, tab property panel akan aktif dan menunjukkan property dari object aktor yang bersangkutan.

Gambar 4.8 Object Aktor pada main drawing panel

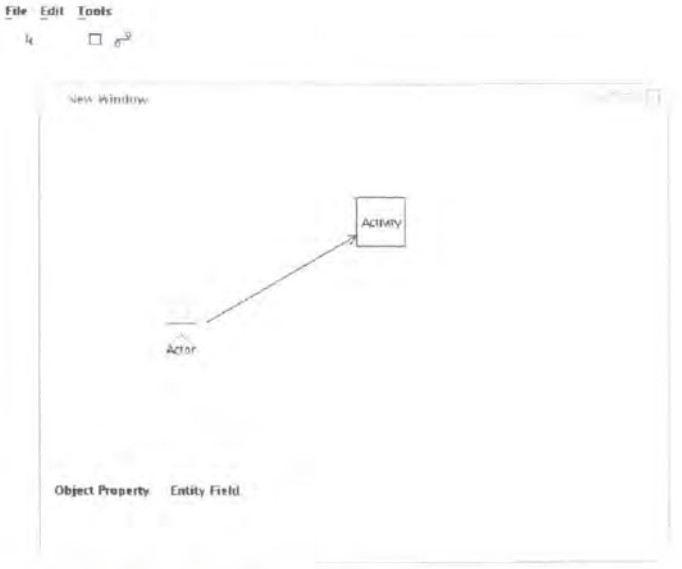
4.2.1.2 Object Activity

Object ini nantinya digunakan untuk menggambarkan object activity dalam workflow. Object activity ini digambarkan dengan icon seperti pada gambar 4.9. Object activity ini nantinya akan memiliki property-property seperti id, nama, type, dan aktorId . Selain itu, object ini juga dapat memiliki entity field. Property serta entity tadi dimasukkan dalam tab setting property yang terletak di bagian paling bawah dari child window yang muncul.

Gambar 4.9 Icon untuk object activity

Bila nantinya toolbar bagian object activity ini diaktifkan, lalu di taruh dalam *main drawing panel*, maka object ini akan tampil di bagian dimana mouse diclick, seperti ditunjukkan dalam gambar 4.10. Secara

otomatis, tab property panel akan aktif dan menunjukkan property dari object activity yang bersangkutan.



Gambar 4.10 Object Aktor dan activity pada main drawing panel

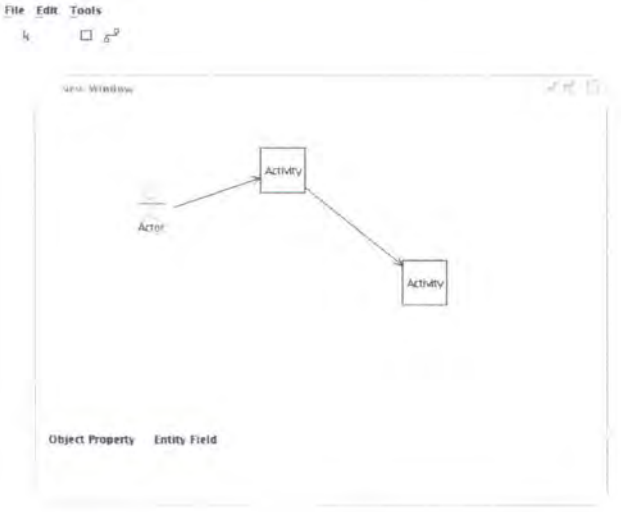
4.2.1.3 Object Connector

Object ini nantinya digunakan untuk menggambarkan object connector yang menunjukkan alur dalam worklfow. Object ini nantinya akan menghubungkan, baik antara aktor dengan activity, maupun antara activity dengan activity. Arah panah yang ditunjukkan oleh object ini menggambarkan urut-urutan eksekusi suatu activity dalam workflow.

Gambar 4.11 Icon untuk object connector

Object connector ini digambarkan dengan icon seperti pada gambar 4.11. Object ini tidak memiliki property yang dapat dimanipulasi oleh user, karena seluruh nilai propertynya langsung diambil dari workflow editor. Bila

nantinya toolbar bagian object connector ini diaktifkan, dan digunakan untuk menghubungkan dua buah object aktor dengan activity, atau activity dengan activity, maka akan terbentuk suatu garis konektor. Gambar 4.12 menunjukkan contoh penggunaan object ini.



Gambar 4.12 Seluruh object pada main drawing panel

4.2.1.4 Object Selector

Object ini digunakan untuk operasi cursor biasa, seperti select object, kemudian melakukan operasi delete object, move object, dan sebagainya. Gambar 4.15 menunjukkan icon untuk object Selector yang terletak di dalam toolbar.

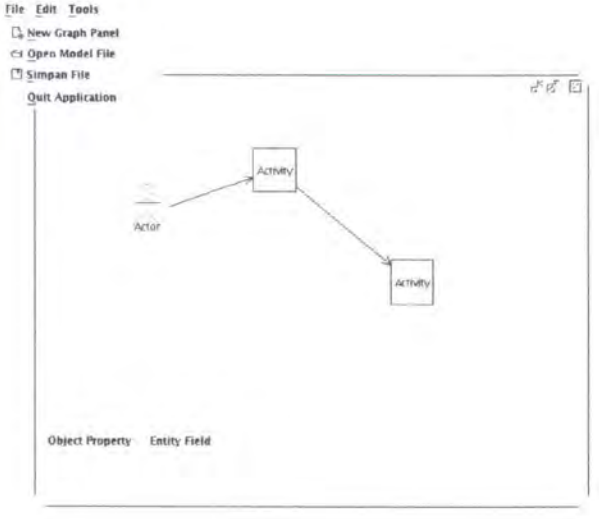
Gambar 4.13 Icon untuk object Selector

4.2.2 Menu Panel

Panel ini terdiri atas 3 buah menu, yaitu menu File, Edit, dan Tools. Masing-masing menu ini memiliki sub menu yang berhubungan dengan menu diatasnya.

4.2.2.1 Menu File

Menu File ini memiliki 4 buah sub menu, yaitu sub menu *New Graph Panel, Open Model File, Simpan File, dan Quit Application.* Gambar 4.14 menunjukkan tata letak menu dan sub menu ini.



Gambar 4.14 Menu File dan sub menunya

a. Sub Menu New Graph Panel

   Sub menu ini digunakan untuk membuka jendela anak baru yang berisi main drawing panel tempat meletakkan object-object dari toolbar panel..

b. Sub Menu Open Model File

Sub menu ini digunakan untuk membuka model file yang telah tersimpan sebelumnya, untuk kemudian di load di main drawing panel.

c. Sub Menu Simpan File

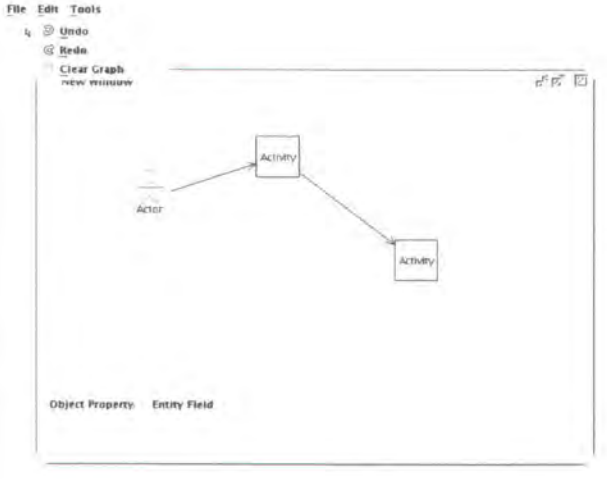
Sub menu ini digunakan untuk menyimpan hasil model yang telah digambar dalam main drawing panel, ke dalam suatu file berformat XML.

d. Sub Menu Quit Application

Sub menu ini digunakan untuk keluar dari aplikasi, dan kembali ke sistem.

4.2.2.2 Menu Edit

Menu Edit ini memiliki 3 buah sub menu, yaitu sub menu *Undo, Redo, dan Clear Graph.* Gambar 4.14 menunjukkan tata letak menu dan sub menu ini.



Gambar 4.15 Menu Edit dan sub menunya

a. Sub Menu Undo

Sub menu ini digunakan untuk melakukan proses undo terhadap event yang dilakukan sebelumnya.

b. Sub Menu Redo

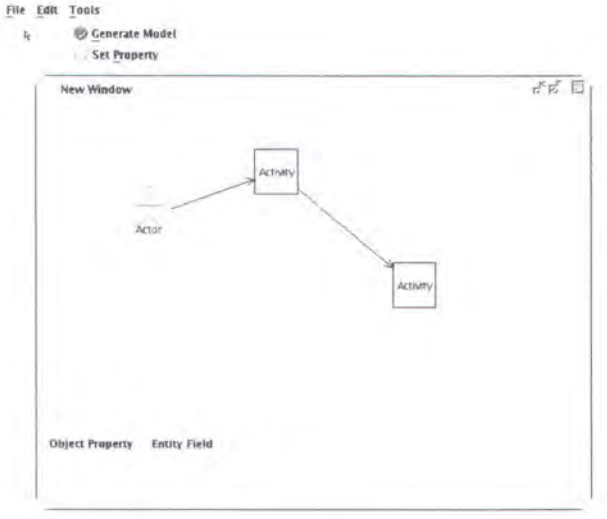
Sub menu ini digunakan untuk melakukan proses redo terhadap event yang dilakukan sebelumnya.

c. Sub Menu Clear Graph

Sub menu ini digunakan untuk melakukan penghapusan terhadap seluruh object yang telah berada diatas drawing panel.

4.2.2.3 Menu Tools

Menu Tools ini memiliki 2 buah sub menu, yaitu sub menu *Generate Model dan Set Property.* Gambar 4.15 menunjukkan tata letak menu dan sub menu ini.



Gambar 4.16 Menu Tools dan sub menunya

a. Sub Menu Generate Model

Sub menu ini digunakan untuk melakukan proses generate untuk model yang telah dibangun sebelumnya.

b. Sub Menu Set Property

Sub menu ini digunakan untuk menginputkan isi property dalam suatu project, seperti nama author dan description project yang bersangkutan.

4.2.3 Main Drawing Panel

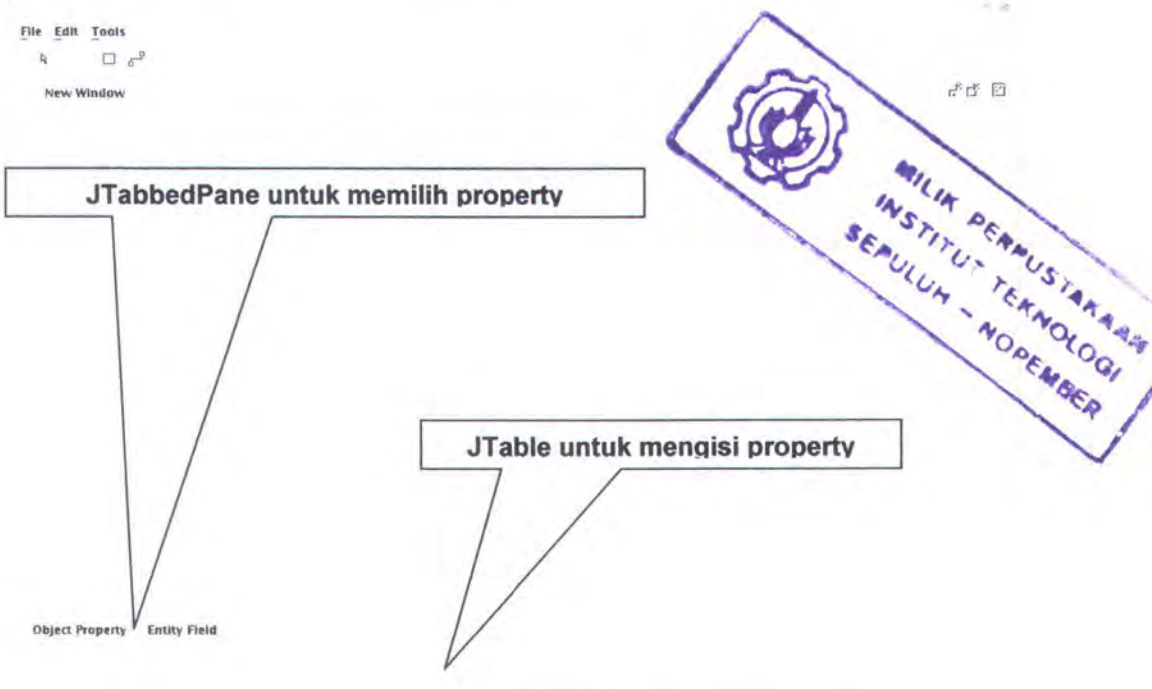
Panel ini terletak di bagian tengah aplikasi, dan akan diload apabila terjadi event yang meminta pembuatan model baru. Di *main drawing panel* inilah nantinya seluruh object akan digambarkan satu persatu menjadi sebuah workflow. Gambar 4.16 menunjukkan bagian main drawing panel ini.

Gambar 4.17 Main Drawing Panel

Panel ini akan muncul ketika dipilih menu File -> New Model. Event ini akan menginstansiasi class workflow.actions.FileNew yang diturunkan dari class abstractAction. Class FileNew ini selanjutnya akan menginstansiasi class workflow.actions.Newmodel yang merupakan turunan dari class JInternalFrame. Sedangkan untuk bagian *drawing panel* yang terletak di tengah, digunakan class DefaultGraphModel dari library jgraph.

4.2.4 Property Tab

Panel ini terletak di bagian bawah *main drawing panel*, dan melekat menjadi sati bersama-sama dengan *main drawing panel*.



Gambar 4.18 Property Tab Panel

Di dalam panel ini terdapat dua macam object yaitu JTable untuk mengisi property, dan JTabbedPane untuk memilih bagian property mana yang akan diisi.

BAB V
UJI COBA DAN EVALUASI

BAB V

UJI COBA DAN EVALUASI

Bab ini menjelaskan proses uji coba dan evaluasi perangkat lunak yang telah dikembangkan dengan mempergunakan data yang telah disiapkan. Adapun tujuan dari proses ujicoba ini adalah apakah program workflow modeler ini dapat berfungsi dengan baik dengan segala parameter yang nantinya akan diujikan.

5.1 Lingkungan Uji Coba

Uji Coba untuk aplikasi ini dilakukan dengan menggunakan lingkungan dengan spesifikasi sebagai berikut :

5.1.1 Spesifikasi Sistem

- Sistem Operasi Linux Mandrake 10.
- Sistem Operasi Windows 2000 Advanced Server.
- DBMS MySQL yang terletak di localhost.
- Web container bawaan JWSDP(Java Web Service Developer Pack) versi 1.3 yang telah terintegrasi dengan Apache Tomcat 5.0.29
- Browser Opera versi 7.5 dan Microsoft Internet Explorer 6.0.

5.1.2 Spesifikasi Perangkat Keras

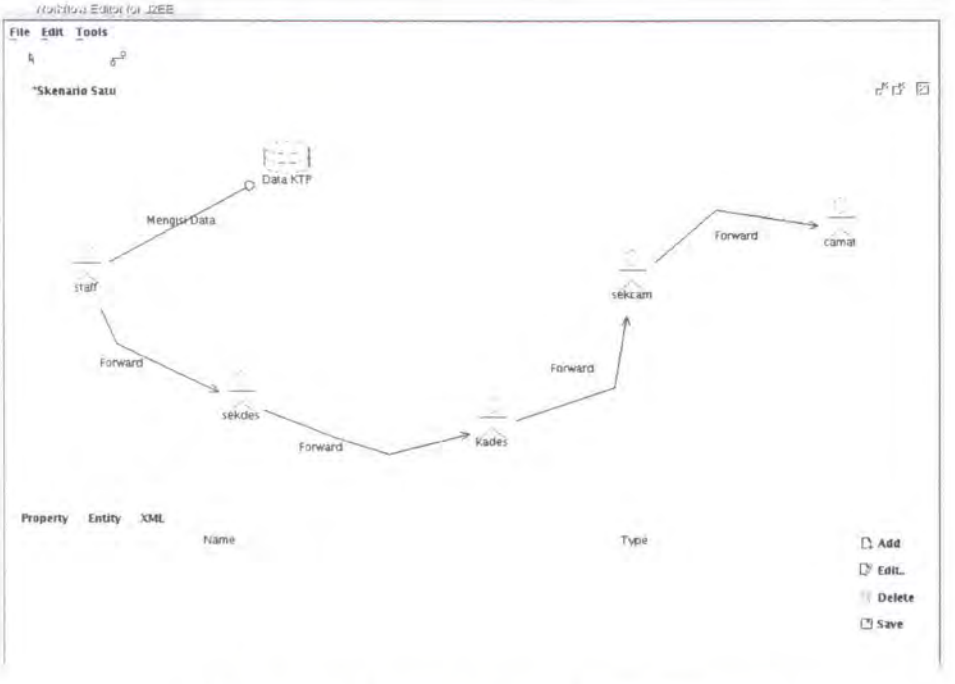
- Intel Celeron 1,1 Ghz
- RAM 504 MB
- 20 Gb Hardisk

5.2 Skenario Uji Coba

Di dalam proses ujicoba ini akan dijalankan tiga buah skenario dengan bobot masalah yang berbeda untuk memperoleh informasi seberapa jauh validitas software ini dalam menggenerate sebuah aplikasi workflow.

5.2.1 Skenario Pertama

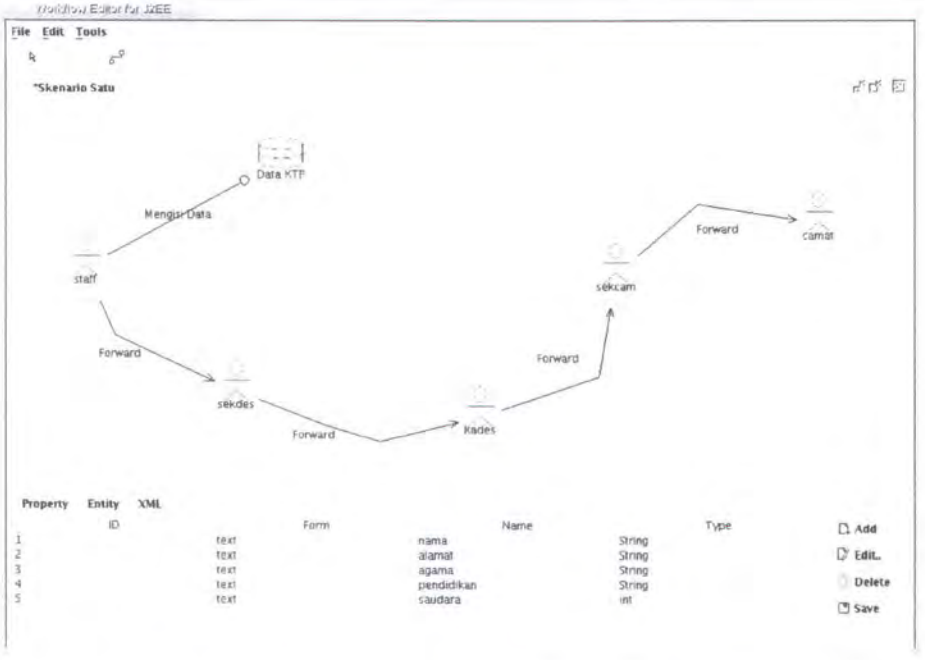
Dalam skenario pertama akan dicoba apakah aplikasi workflow modeler ini dapat menggenerate sebuah aplikasi yang alurnya sederhana dan satu arah tanpa ada percabangan flow. Gambar 5.1 menunjukkan diagram yang akan dicoba untuk digunakan dalam skenario pertama ini. Direktori yang akan digunakan dalam proses generate model ini adalah di /usr/local/jwsdp-1.3/webapps/workflow.



Gambar 5.1 Model Diagram Skenario Pertama

Di dalam gambar 5.1 tersebut terdapat 5 buah aktor dan sebuah database yang saling terangkai satu sama lain membentuk alur tunggal. Diagram ini dibentuk

dengan cara *drag and drop* komponen yang bersangkutan ke dalam layer panel yang berada di tengah. Setelah semua aktor diletakkan pada tempatnya, maka langkah selanjutnya adalah mengisi property-property yang dimiliki oleh tiap-tiap object yang bersangkutan. Hal ini ditunjukkan dalam gambar 5.2.



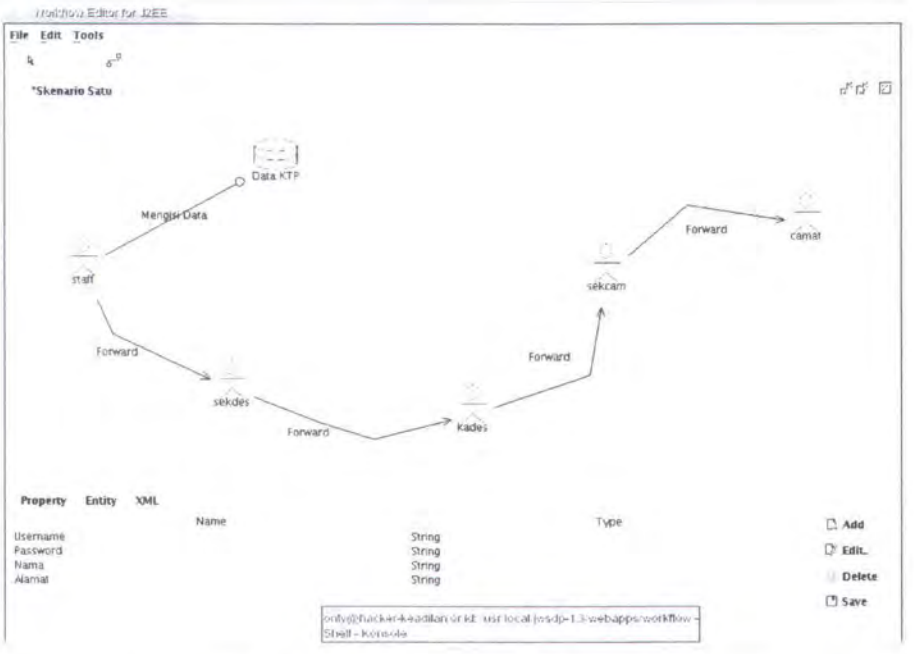
Gambar 5.2 Pengisian Property Database Skenario Pertama

Dalam tabel 5.1 ditunjukkan lebih detail pengisian property untuk object Database sebagai berikut :

Tabel 5.1 Daftar pengisian property untuk object database skenario pertama

ID	Name	Type
1	Nama	String
2	Alamat	String
3	Agama	String
4	Pendidikan	String
5	Saudara	int

Sedangkan pengisian property untuk object aktor dapat dilihat pada gambar 5.3. Pengisian property object aktor ini sama untuk semua aktor.



Gambar 5.3 Pengisian property aktor skenario pertama

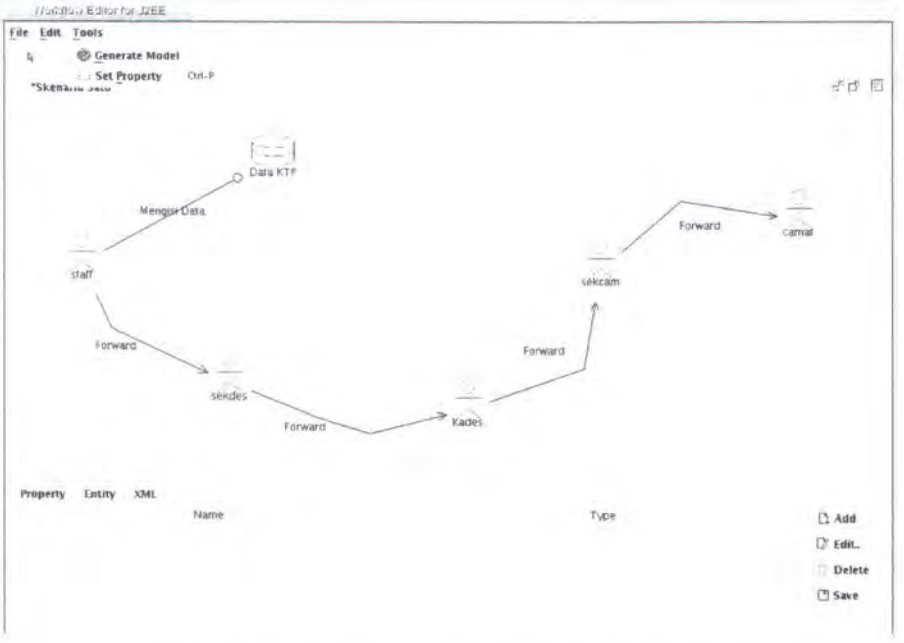
Dalam tabel 5.2 ditunjukkan lebih detail tentang pengisian property untuk object aktor ini sebagai berikut :

Tabel 5.2 Daftar pengisian property untuk object actor skenario pertama

Name	Type
Username	String
Password	String
Nama	String
Alamat	String

Langkah selanjutnya adalah pengisian property lain yang diperlukan dalam proses generate aplikasi, diantaranya yaitu property identitas project, serta property setting untuk koneksi dengan database. Hal ini ditunjukkan dalam gambar 5.4.

Terdapat dua macam kategori property yang harus disetting dalam hal ini, yaitu bagian identitas project, dan setting koneksi ke database. Bagian identitas project terdiri dari field Title dan Author. Bagian Title akan digunakan oleh aplikasi untuk pesan-pesan yang nantinya akan ditampilkan dalam antarmuka aplikasi, seperti bagian window title dan nama database. Sedangkan bagian author hanya digunakan sebagai penanda pembuat dokumen dalam file XML.



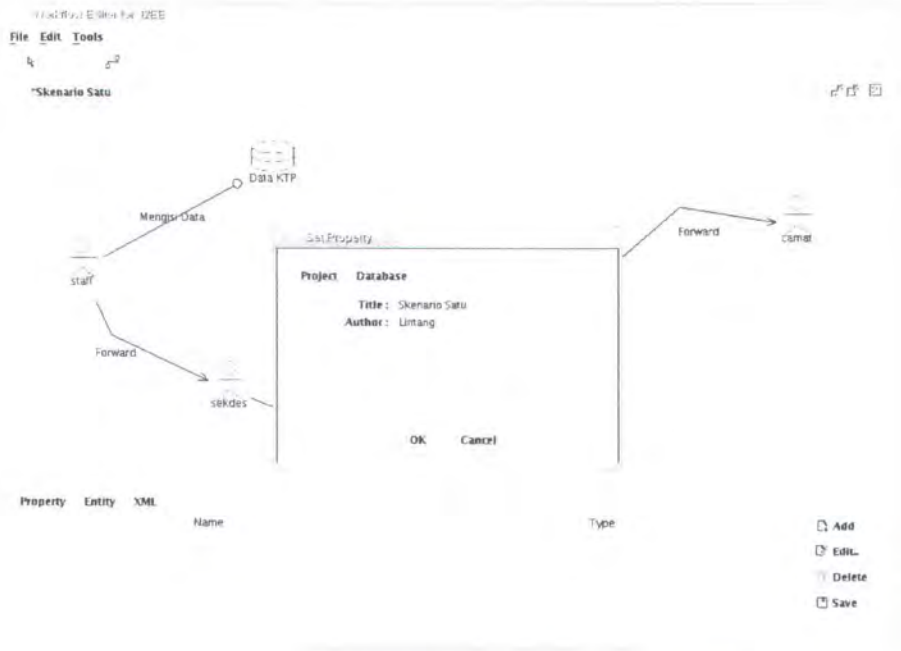
Gambar 5.4 Pemilihan menu untuk setting property.

Sedangkan untuk mengeset property koneksi database ditunjukkan dalam gambar 5.6. Tabel 5.3 menunjukkan lebih detail tentang pengisian setting untuk property koneksi database ini.

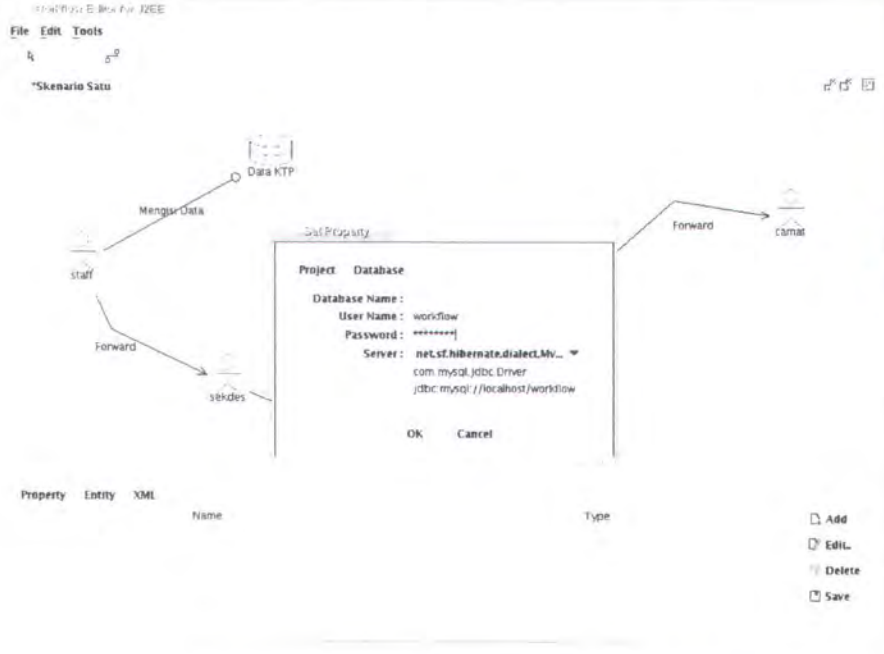
Tabel 5.3 Pengisian property setting database

Database Name	Nama database yang bersangkutan
User Name	Nama user yang digunakan untuk koneksi ke database
Password	Password untuk koneksi ke database
Server	1. Setting dialect bahasa SQL untuk hibernate

	2. Setting driver JDBC untuk koneksi database 3. URL koneksi untuk database.

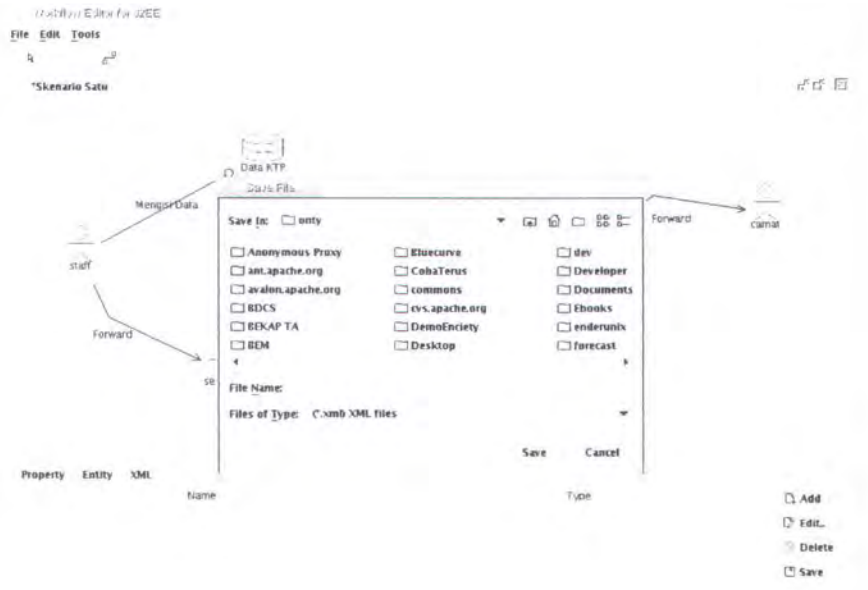


Gambar 5.5 Pengisian property project

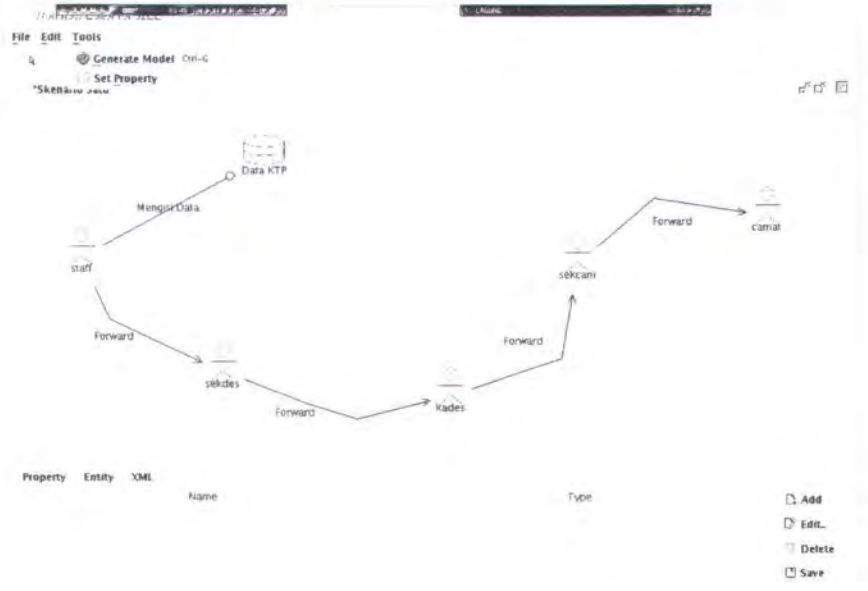


Gambar 5.6 Setting property koneksi database.

Setelah semua setting siap, maka akan dilakukan proses generate model. Namun sebelumnya model yang telah dibuat ini harus disimpan terlebih dahulu. Proses save dan generate ini ditunjukkan dalam gambar 5.7 dan 5.8.



Gambar 5.7 Proses save model ke dalam file.



Gambar 5.8 Proses generate model

Proses diatas akan menghasilkan dokumen XML yang berisi informasi tentang model diagram yang bersangkutan. Gambar 5.9 menunjukkan isi hasil dokumen XML yang terbentuk.

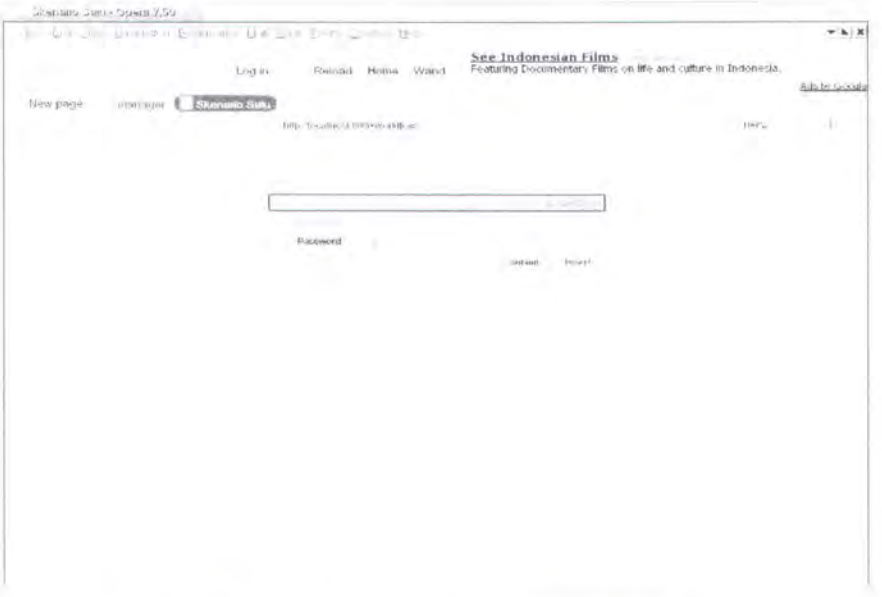
```
<?xml version="1.0" encoding="utf-8"?>
<model>
	<project>
		<title>Skenario Satu</title>
		<author>Lintang</author>
		<created>7/25/2004 4:28:49 PM</created>
	</project>
	<aktors>
		<aktor id="0"  name="staff"  rect="57,144,50,75" >
			<entities>
				<entity>
					<name>Username</name>
					<type>String</type>
				</entity>
				<entity>
					<name>Password</name>
					<type>String</type>
				</entity>
				<entity>
					<name>Nama</name>
					<type>String</type>
				</entity>
				<entity>
					<name>Alamat</name>
					<type>String</type>
				</entity>
			</entities>
		</aktor>
		<aktor id="1"  name="sekdes"  rect="225,281,50,75" >
			<entities>
				<entity>
					<name>Username</name>
					<type>String</type>
				</entity>
				<entity>
					<name>Password</name>
					<type>String</type>
				</entity>
			</entities>
		</aktor>
		<aktor id="2"  name="kades"  rect="497,309,50,75" >
			<entities>
				<entity>
					<name>Username</name>
					<type>String</type>
				</entity>
```

```
                        <entity>
                              <name>Password</name>
                              <type>String</type>
                        </entity>
                  </entities>
            </aktor>
            <aktor id="3"  name="sekcam"  rect="649,151,50,75" >
                  <entities>
                        <entity>
                              <name>Username</name>
                              <type>String</type>
                        </entity>
                        <entity>
                              <name>Password</name>
                              <type>String</type>
                        </entity>
                  </entities>
            </aktor>
            <aktor id="4"  name="camat"  rect="875,95,50,75" >
                  <entities>
                        <entity>
                              <name>Username</name>
                              <type>String</type>
                        </entity>
                        <entity>
                              <name>Password</name>
                              <type>String</type>
                        </entity>
                  </entities>
            </aktor>
      </aktors>
      <activities>
            <activity id="5"  name="Data KTP"  rect="263,28,75,75" >
                  <datafields>
                        <field>
                              <id>1</id>
                              <form>text</form>
                              <name>nama</name>
                              <type>String</type>
                        </field>
                        <field>
                              <id>2</id>
                              <form>text</form>
                              <name>alamat</name>
                              <type>String</type>
                        </field>
                        <field>
                              <id>3</id>
                              <form>text</form>
                              <name>agama</name>
                              <type>String</type>
                        </field>
                        <field>
                              <id>4</id>
                              <form>text</form>
                              <name>pendidikan</name>
```

```
                        <type>String</type>
                    </field>
                    <field>
                        <id>5</id>
                        <form>text</form>
                        <name>saudara</name>
                        <type>int</type>
                    </field>
                </datafields>
            </activity>
        </activities>
        <edges>
            <edge id="5" type="1">
                <src>aktor,0</src>
                <des>activity,4</des>
            </edge>
            <edge id="6" type="0">
                <src>aktor,0</src>
                <des>aktor,1</des>
            </edge>
            <edge id="7" type="0">
                <src>aktor,1</src>
                <des>aktor,2</des>
            </edge>
            <edge id="8" type="0">
                <src>aktor,2</src>
                <des>aktor,3</des>
            </edge>
            <edge id="10" type="0">
                <src>aktor,3</src>
                <des>aktor,9</des>
            </edge>
        </edges>
</model>
```

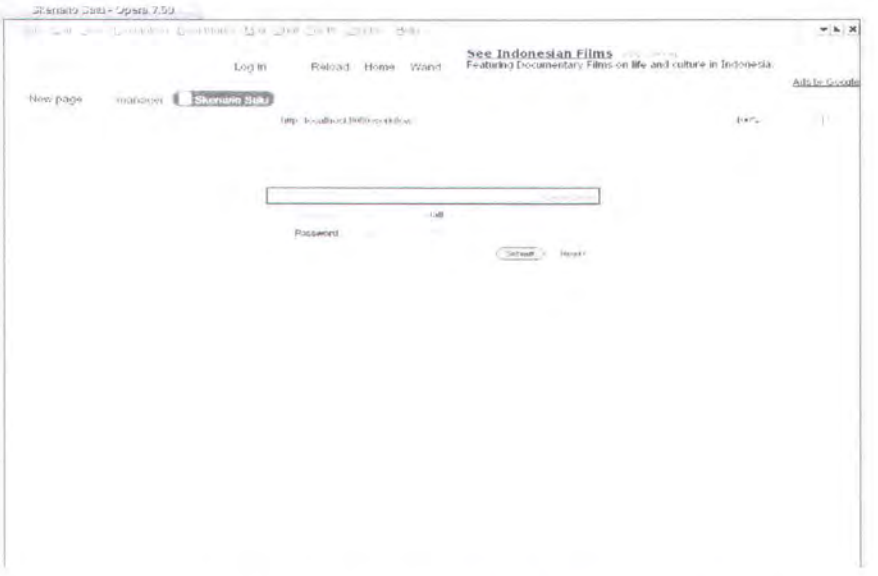
Gambar 5.9 Dokumen XML yang terbentuk untuk scenario pertama

Setelah dilakukan proses generate, maka langkah selanjutnya adalah proses kompilasi beberapa file .java yang digenerate oleh aplikasi, serta proses reload web container yang bersangkutan agar aplikasi yang telah terletak di tempat yang semestinya tersebut dapat terbaca oleh web container. Hasil jadi dari proses generate model ini ditunjukkan oleh gambar 5.10.



Gambar 5.10 Tampilan awal aplikasi yang digenerate

Tampilan awal yang muncul adalah layar login. Layar ini secara default akan muncul sebagai tampilan halaman pertama ketika context root untuk aplikasi workflow diakses melalui browser, dalam hal ini untuk server local digunakan URL http://localhost:8080/workflow/ . Secara default pula, tiap-tiap aktor yang telah digambarkan dalam model diagram akan memiliki account di dalam aplikasi ini. Username dan password yang tergenerate untuk pertama kali adalah sama dengan nama masing-masing aktor yang bersangkutan. Masing-masing aktor ini harus melalui proses otentikasi dengan login ini untuk dapat mengakses database melalui aplikasi workflow ini. Gambar 5.11 menunjukkan proses login dari aktor "staff".



Gambar 5.11 Proses login dari aktor "staff"

Setelah aktor "staff" berhasil login, karena dalam diagram sebelumnya aktor inilah yang terhubung ke database, maka terdapat 3 buah pilihan menu, yang ditunjukkan dalam gambar 5.12.

Gambar 5.12 Pilihan menu untuk aktor "staff"

Proses pertama kali yang harus dilakukan adalah menginputkan data. Gambar 5.13 menunjukkan proses input data untuk pertama kali. Field-field yang harus diisikan disini secara otomatis tergenerate berdasarkan model diagram yang telah dibentuk sebelumnya. Selain itu juga tergenerate sebuah field default yaitu "keterangan" yang nantinya digunakan sebagai referensi disposisi data ke aktor tujuan.

Gambar 5.13 Layar input data

Setelah proses pengisian data selesai, maka secara otomatis aktor akan diredirect menuju halaman view data. Disini dapat dilihat data apa saja yang telah diinputkan, dan siap untuk di disposisikan. Gambar 5.14 menunjukkan bagian view data ini. Dari halaman inilah nantinya aktor dapat medisposisikan data menuju ke aktor lainnya dengan cara meng-klik tombol approve. Setelah tombol ini ditekan, nantinya data yang bersangkutan akan segera terdisposisi ke aktor tujuan. Sesuai dengan model diagram yang telah dibangun sebelumnya, aktor "staff" hanya dapat melakukan disposisi ke "sekdes", maka dalam combo-box yang tersedia hanya ada pilihan disposisi menuju aktor "sekdes".

Gambar 5.14 Layar view data untuk aktor staff.

Hal yang sama juga akan terjadi bila dilakukan proses login sebagai aktor "kades" dan "sekcam". Hanya saja perbedaannya terletak pada hak untuk menginputkan data, yang tidak dimiliki oleh aktor lain selain staff, sesuai dengan yang ditunjukkan dalam model diagram. Gambar 5.15 dan 5.16 menunjukkan tampilan layar view milik aktor "kades" dan "sekcam".

Gambar 5.15 Tampilan layer view untuk aktor kades.

Gambar 5.16 Tampilan layar view untuk login sekcam

Sesuai dengan model diagram, seluruh data bermuara pada aktor "camat". Di dalam tampilan layer view untuk camat, tidak terdapat lagi tombol untuk disposisi, melainkan hanya sekedar review data apa saja yang telah masuk. Gambar 5.17 menunjukkan tampilan layar view untuk aktor "camat". Selain itu, kini data history telah lengkap terisi, dan data history untuk tiap-tiap data yang masuk ini dapat dilihat dengan menekan tombol "Lihat History". Gambar 5.18 menunjukkan hasil history untuk salah satu data yang masuk.

Gambar 5.17 Tampilan layar view untuk aktor "camat"

Gambar 5.18 Tampilan layar history untuk salah satu data yang masuk.

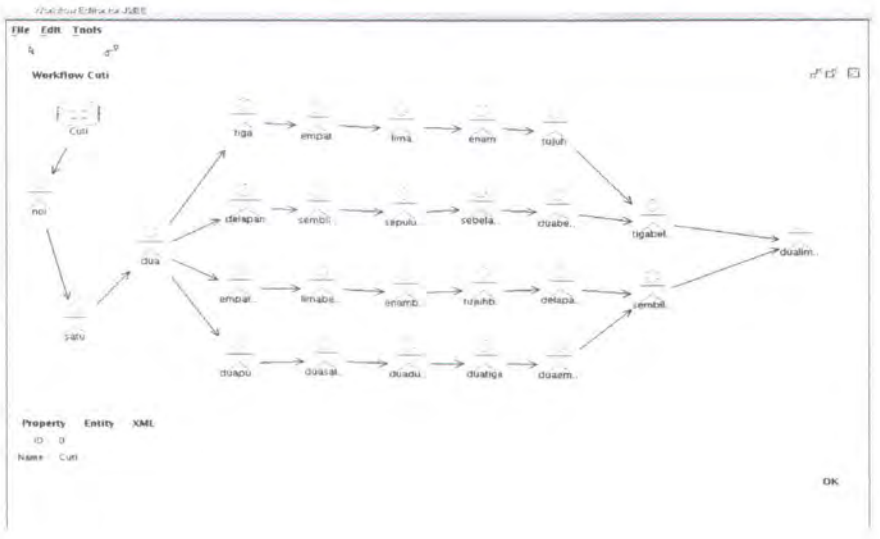
Dari layar history inilah dapat diketahui, urutan perjalanan data sesuai dengan diagram, mulai dari "staff" menuju ke "sekcam", kemudian menuju ke "kades" dan "sekcam", baru kemudian berakhir di "camat".

5.2.2 Skenario Kedua

Skenario kedua ini bertujuan untuk menguji seberapa banyak actor yang mampu ditangani oleh workflow designer ini. Di dalam skenario kedua ini diambil sebuah contoh kasus prosedur pengajuan cuti dengan 26 buah actor.

Sebuah perusahaan X memiliki prosedur untuk pengajuan cuti bagi karyawannya. Untuk setiap pengajuan cuti, surat permohonan harus mendapat persetujuan dari direktur utama (dualima). Setiap pengajuan cuti, karyawan harus mengisi formulir, dan melalui persetujuan beberapa orang atasan. Diagram workflow lebih lanjut dapat dilihat pada gambar 5.19.

Dalam skenario kedua ini yang ditekankan adalah percobaan untuk menentukan seberapa banyak jumlah aktor yang mampu ditangani dalam workflow ini.



Gambar 5.19 Model diagram skenario kedua

Dalam studi kasus perusahaan x ini, setiap permohonan cuti diinputkan oleh aktor "nol", dan berturut-turut harus melalui persetujuan beberapa aktor berikunya sehingga akhirnya permohonan cuti tersebut sampai pada aktor "dualima". Gambar 5.20 menunjukkan dokumen XML yang terbentuk dari skenario 2 ini.

```
<?xml version="1.0" encoding="utf-8"?>
<model>
      <project>
            <title>Workflow Cuti</title>
            <author>Prasojo</author>
            <created>8/4/2004 5:56:50 AM</created>
      </project>
      <actors>
            <actor id="0"  name="nol"  rect="7,119,50,75" >
                  <entities>
                        <entity>
                              <name>Username</name>
                              <type>String</type>
                        </entity>
                        <entity>
                              <name>Password</name>
                              <type>String</type>
                        </entity>
                  </entities>
            </actor>
```

```
<actor id="1"  name="satu"  rect="48,290,50,75" >
      <entities>
            <entity>
                  <name>Username</name>
                  <type>String</type>
            </entity>
            <entity>
                  <name>Password</name>
                  <type>String</type>
            </entity>
      </entities>
</actor>
<actor id="2"  name="dua"  rect="137,186,50,75" >
      <entities>
            <entity>
                  <name>Username</name>
                  <type>String</type>
            </entity>
            <entity>
                  <name>Password</name>
                  <type>String</type>
            </entity>
      </entities>
</actor>
<actor id="3"  name="tiga"  rect="247,11,50,75" >
      <entities>
            <entity>
                  <name>Username</name>
                  <type>String</type>
            </entity>
            <entity>
                  <name>Password</name>
                  <type>String</type>
            </entity>
      </entities>
</actor>
<actor id="4"  name="delapan"  rect="243,129,61,75" >
      <entities>
            <entity>
                  <name>Username</name>
                  <type>String</type>
            </entity>
            <entity>
                  <name>Password</name>
                  <type>String</type>
            </entity>
      </entities>
</actor>
<actor id="5"  name="empat"  rect="333,15,50,75" >
      <entities>
            <entity>
                  <name>Username</name>
                  <type>String</type>
            </entity>
            <entity>
                  <name>Password</name>
```

```
                        <type>String</type>
                  </entity>
            </entities>
      </actor>
      <actor id="6"  name="sembilan"  rect="334,130,50,75" >
            <entities>
                  <entity>
                        <name>Username</name>
                        <type>String</type>
                  </entity>
                  <entity>
                        <name>Password</name>
                        <type>String</type>
                  </entity>
            </entities>
      </actor>
      <actor id="7"  name="limabelas"  rect="338,238,50,75" >
            <entities>
                  <entity>
                        <name>Username</name>
                        <type>String</type>
                  </entity>
                  <entity>
                        <name>Password</name>
                        <type>String</type>
                  </entity>
            </entities>
      </actor>
      <actor id="8"  name="duasatu"  rect="342,338,50,75" >
            <entities>
                  <entity>
                        <name>Username</name>
                        <type>String</type>
                  </entity>
                  <entity>
                        <name>Password</name>
                        <type>String</type>
                  </entity>
            </entities>
      </actor>
      <actor id="9"  name="duapuluh"  rect="241,340,50,75" >
            <entities>
                  <entity>
                        <name>Username</name>
                        <type>String</type>
                  </entity>
                  <entity>
                        <name>Password</name>
                        <type>String</type>
                  </entity>
            </entities>
      </actor>
      <actor id="10"  name="empatbelas"  rect="241,238,50,75"
>
            <entities>
                  <entity>
```

```
                        <name>Username</name>
                        <type>String</type>
                  </entity>
                  <entity>
                        <name>Password</name>
                        <type>String</type>
                  </entity>
            </entities>
      </actor>
      <actor id="11"  name="lima"  rect="432,18,50,75" >
            <entities>
                  <entity>
                        <name>Username</name>
                        <type>String</type>
                  </entity>
                  <entity>
                        <name>Password</name>
                        <type>String</type>
                  </entity>
            </entities>
      </actor>
      <actor id="12"  name="sepuluh"  rect="435,133,50,75" >
            <entities>
                  <entity>
                        <name>Username</name>
                        <type>String</type>
                  </entity>
                  <entity>
                        <name>Password</name>
                        <type>String</type>
                  </entity>
            </entities>
      </actor>
      <actor id="13"  name="duadua"  rect="440,341,50,75" >
            <entities>
                  <entity>
                        <name>Username</name>
                        <type>String</type>
                  </entity>
                  <entity>
                        <name>Password</name>
                        <type>String</type>
                  </entity>
            </entities>
      </actor>
      <actor id="14"  name="enambelas"  rect="436,243,50,75" >
            <entities>
                  <entity>
                        <name>Username</name>
                        <type>String</type>
                  </entity>
                  <entity>
                        <name>Password</name>
                        <type>String</type>
                  </entity>
            </entities>
```

```
            </actor>
            <actor id="15"  name="enam"  rect="527,19,50,75" >
                  <entities>
                        <entity>
                              <name>Username</name>
                              <type>String</type>
                        </entity>
                        <entity>
                              <name>Password</name>
                              <type>String</type>
                        </entity>
                  </entities>
            </actor>
            <actor id="16"  name="sebelas"  rect="525,130,50,75" >
                  <entities>
                        <entity>
                              <name>Username</name>
                              <type>String</type>
                        </entity>
                        <entity>
                              <name>Password</name>
                              <type>String</type>
                        </entity>
                  </entities>
            </actor>
            <actor id="17"  name="tujuhbelas"  rect="528,241,50,75"
>
                  <entities>
                        <entity>
                              <name>Username</name>
                              <type>String</type>
                        </entity>
                        <entity>
                              <name>Password</name>
                              <type>String</type>
                        </entity>
                  </entities>
            </actor>
            <actor id="18"  name="duatiga"  rect="530,341,50,75" >
                  <entities>
                        <entity>
                              <name>Username</name>
                              <type>String</type>
                        </entity>
                        <entity>
                              <name>Password</name>
                              <type>String</type>
                        </entity>
                  </entities>
            </actor>
            <actor id="19"  name="tujuh"  rect="613,22,50,75" >
                  <entities>
                        <entity>
                              <name>Username</name>
                              <type>String</type>
                        </entity>
```

```
                        <entity>
                              <name>Password</name>
                              <type>String</type>
                        </entity>
                  </entities>
            </actor>
            <actor id="20"  name="duabelas"  rect="616,134,50,75" >
                  <entities>
                        <entity>
                              <name>Username</name>
                              <type>String</type>
                        </entity>
                        <entity>
                              <name>Password</name>
                              <type>String</type>
                        </entity>
                  </entities>
            </actor>
            <actor id="21"  name="delapanbelas"
rect="620,239,50,75" >
                  <entities>
                        <entity>
                              <name>Username</name>
                              <type>String</type>
                        </entity>
                        <entity>
                              <name>Password</name>
                              <type>String</type>
                        </entity>
                  </entities>
            </actor>
            <actor id="22"  name="duaempat"  rect="617,342,50,75" >
                  <entities>
                        <entity>
                              <name>Username</name>
                              <type>String</type>
                        </entity>
                        <entity>
                              <name>Password</name>
                              <type>String</type>
                        </entity>
                  </entities>
            </actor>
            <actor id="23"  name="tigabelas"  rect="728,149,50,75" >
                  <entities>
                        <entity>
                              <name>Username</name>
                              <type>String</type>
                        </entity>
                        <entity>
                              <name>Password</name>
                              <type>String</type>
                        </entity>
                  </entities>
            </actor>
```

```
            <actor id="24"  name="sembilanbelas"
rect="727,246,50,75" >
                  <entities>
                        <entity>
                              <name>Username</name>
                              <type>String</type>
                        </entity>
                        <entity>
                              <name>Password</name>
                              <type>String</type>
                        </entity>
                  </entities>
            </actor>
            <actor id="25"  name="dualima"  rect="901,174,50,75" >
                  <entities>
                        <entity>
                              <name>Username</name>
                              <type>String</type>
                        </entity>
                        <entity>
                              <name>Password</name>
                              <type>String</type>
                        </entity>
                  </entities>
            </actor>
      </actors>
      <activities>
            <activity id="0"  name="Cuti"  rect="41,9,75,75" >
                  <datafields>
                        <field>
                              <id>1</id>
                              <name>nama</name>
                              <type>String</type>
                        </field>
                        <field>
                              <id>2</id>
                              <name>departemen</name>
                              <type>String</type>
                        </field>
                        <field>
                              <id>3</id>
                              <name>waktu</name>
                              <type>int</type>
                        </field>
                  </datafields>
            </activity>
      </activities>
      <edges>
            <edge id="0" type="0">
                  <src>activity,0</src>
                  <des>actor,0</des>
            </edge>
            <edge id="1" type="0">
                  <src>actor,0</src>
                  <des>actor,1</des>
            </edge>
```

```
<edge id="2" type="0">
      <src>actor,1</src>
      <des>actor,2</des>
</edge>
<edge id="3" type="0">
      <src>actor,2</src>
      <des>actor,3</des>
</edge>
<edge id="4" type="0">
      <src>actor,2</src>
      <des>actor,4</des>
</edge>
<edge id="5" type="0">
      <src>actor,2</src>
      <des>actor,10</des>
</edge>
<edge id="6" type="0">
      <src>actor,2</src>
      <des>actor,9</des>
</edge>
<edge id="7" type="0">
      <src>actor,9</src>
      <des>actor,8</des>
</edge>
<edge id="8" type="0">
      <src>actor,10</src>
      <des>actor,7</des>
</edge>
<edge id="9" type="0">
      <src>actor,4</src>
      <des>actor,6</des>
</edge>
<edge id="10" type="0">
      <src>actor,3</src>
      <des>actor,5</des>
</edge>
<edge id="11" type="0">
      <src>actor,5</src>
      <des>actor,11</des>
</edge>
<edge id="12" type="0">
      <src>actor,6</src>
      <des>actor,12</des>
</edge>
<edge id="13" type="0">
      <src>actor,7</src>
      <des>actor,14</des>
</edge>
<edge id="14" type="0">
      <src>actor,8</src>
      <des>actor,13</des>
</edge>
<edge id="15" type="0">
      <src>actor,11</src>
      <des>actor,15</des>
</edge>
```

```
            <edge id="16" type="0">
                  <src>actor,12</src>
                  <des>actor,16</des>
            </edge>
            <edge id="17" type="0">
                  <src>actor,14</src>
                  <des>actor,17</des>
            </edge>
            <edge id="18" type="0">
                  <src>actor,13</src>
                  <des>actor,18</des>
            </edge>
            <edge id="19" type="0">
                  <src>actor,15</src>
                  <des>actor,19</des>
            </edge>
            <edge id="20" type="0">
                  <src>actor,16</src>
                  <des>actor,20</des>
            </edge>
            <edge id="21" type="0">
                  <src>actor,17</src>
                  <des>actor,21</des>
            </edge>
            <edge id="22" type="0">
                  <src>actor,18</src>
                  <des>actor,22</des>
            </edge>
            <edge id="23" type="0">
                  <src>actor,19</src>
                  <des>actor,23</des>
            </edge>
            <edge id="24" type="0">
                  <src>actor,20</src>
                  <des>actor,23</des>
            </edge>
            <edge id="25" type="0">
                  <src>actor,21</src>
                  <des>actor,24</des>
            </edge>
            <edge id="26" type="0">
                  <src>actor,22</src>
                  <des>actor,24</des>
            </edge>
            <edge id="27" type="0">
                  <src>actor,23</src>
                  <des>actor,25</des>
            </edge>
            <edge id="28" type="0">
                  <src>actor,24</src>
                  <des>actor,25</des>
            </edge>
      </edges>
</model>
```

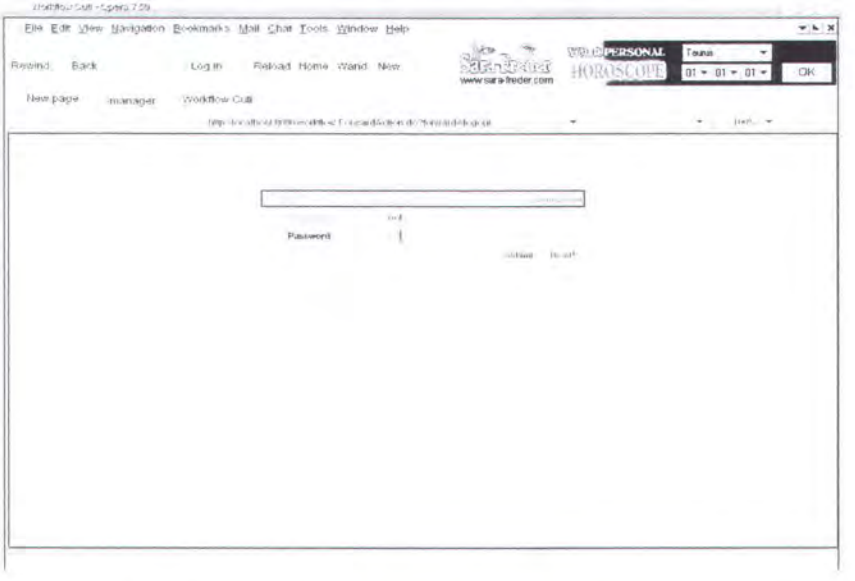
Gambar 5.20 Dokumen XML yang terbentuk dari scenario kedua

Sedangkan pengisian untuk property-property database ditunjukkan dalam tabel 5.4.

Tabel 5.4 Daftar pengisian property untuk object database skenario kedua

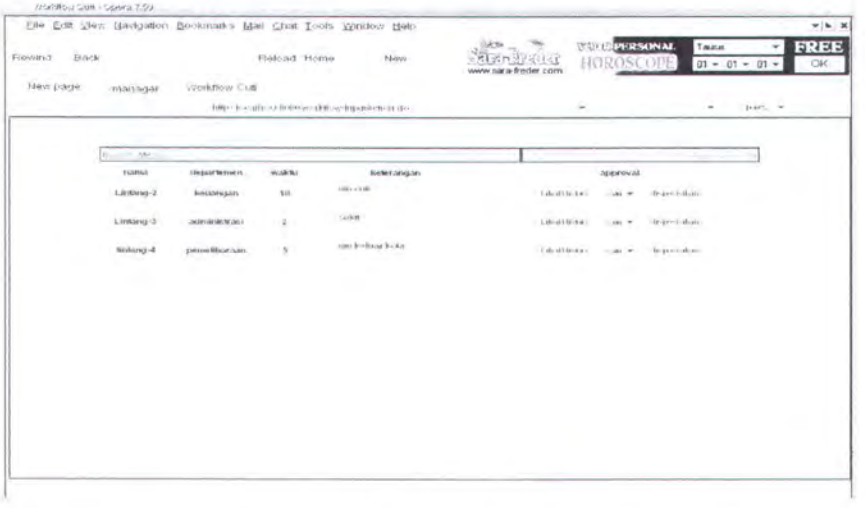
ID	Name	Type
1	Nama	String
2	Departemen	String
3	Waktu	Int

Untuk isi dari setting database dilakukan sama seperti pada skenario pertama. Setelah dilakukan proses generate, maka langkah selanjutnya adalah proses kompilasi beberapa file .java yang digenerate oleh aplikasi, serta proses reload web container yang bersangkutan agar aplikasi yang telah terletak di tempat yang semestinya tersebut dapat terbaca oleh web container. Hasil jadi dari proses generate model ini ditunjukkan oleh gambar 5.21.



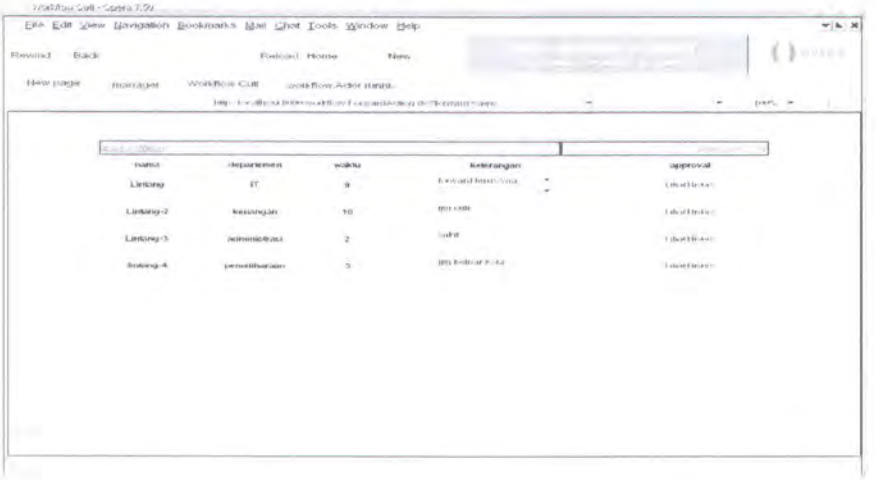
Gambar 5.21 Tampilan layar login model diagram skenario ketiga

Dalam skenario kedua ini hasil workflow yang diharapkan adalah sampainya informasi dari aktor "nol" menuju aktor "dualima" melalui beberapa jalur aktor yang berbeda. Gambar 5.23 menunjukkan informasi yang akan dicoba untuk didisposisikan melalui workflow ini.



Gambar 5.22 Informasi yang akan diujikan untuk skenario kedua

Setelah melalui proses disposisi, maka seluruh informasi tersebut sampai kepada aktor "dualima". Gambar 5.23 menunjukkan hasil disposisi yang sampai pada aktor "dualima".

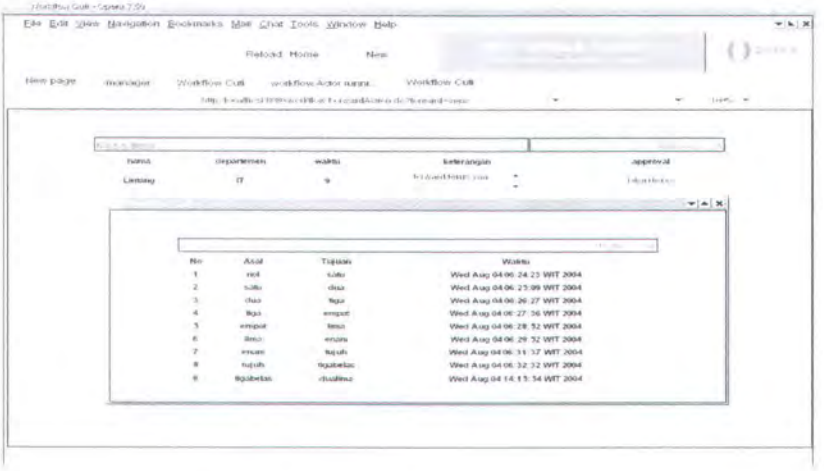


Gambar 5.23 Informasi yang sampai pada aktor "dualima"

Pada gambar 5.24 ditunjukkan alur pertama disposisi informasi, yaitu melalui aktor "nol" – "satu" – "dua" – "delapan" – "sembilan" – "sepuluh" – "sebelas" – "duabelas" – "tigabelas" – "dualima".

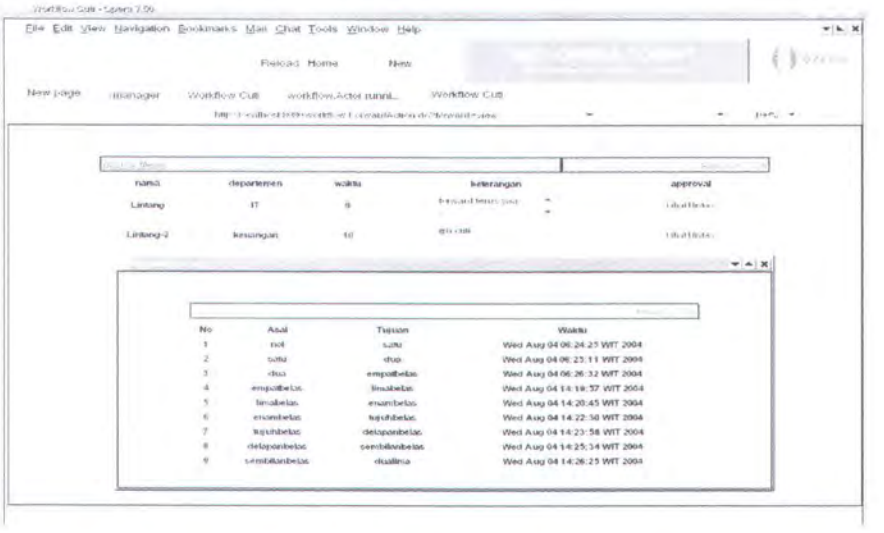
Gambar 5.24 Tracking alur pertama disposisi informasi skenario kedua

Pada gambar 5.25 ditunjukkan alur kedua disposisi informasi, yaitu melalui aktor "nol" – "satu" – "dua" – "tiga" – "empat" – "lima" – "enam" – "tujuh" – "tigabelas" – "dualima".



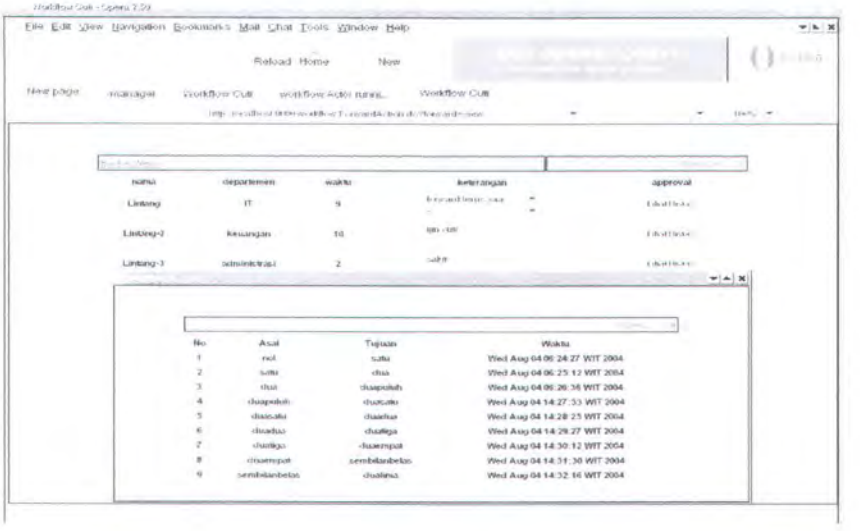
Gambar 5.25 Tracking alur kedua disposisi informasi skenario kedua

Pada gambar 5.26 ditunjukkan alur ketiga disposisi informasi, yaitu melalui aktor "nol" – "satu" – "dua" – "empatbelas" – "limabelas" – "enambelas" – "tujuhbelas" – "delapanbelas" – "sembilanbelas" – "dualima".



Gambar 5.26 Tracking alur ketiga disposisi informasi skenario kedua

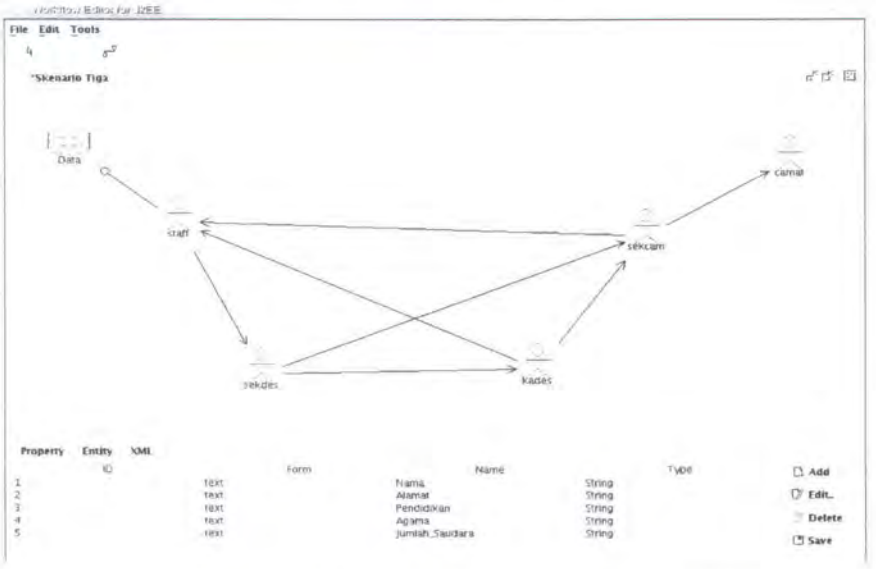
Pada gambar 5.27 ditunjukkan alur keempat disposisi informasi, yaitu melalui aktor "nol" – "satu" – "dua" – "duapuluh" – "duasatu" – "duadua" – "duatiga" – "duaempat" – "sembilanbelas" – "dualima".



Gambar 5.27 Tracking alur keempat disposisi informasi skenario kedua

5.2.3 Skenario Ketiga

Dalam skenario ketiga ini akan dicoba diagram yang lebih kompleks daripada skenario pertama dan kedua. Gambar 5.28 menunjukkan model diagram yang akan digunakan di dalam skenario ketiga ini.



Gambar 5.28 Model diagram skenario ketiga

Diagram dalam gambar 5.28 menunjukkan percabangan yang lebih kompleks daripada skenario sebelumnya. Dalam model diagram skenario ketiga ini terdapat 5 buah aktor dengan beberapa jalur yang berbeda untuk menuju titik akhir. Diagram ini dibentuk dengan cara *drag and drop* komponen yang bersangkutan ke dalam layer panel yang berada di tengah. Setelah semua aktor diletakkan pada tempatnya, maka langkah selanjutnya adalah mengisi property-property yang dimiliki oleh tiap-tiap object yang bersangkutan. Langkah-langkahnya sama seperti pada skenario pertama dan kedua. Gambar 5.29 menunjukkan dokumen XML yang terbentuk dari diagram scenario ketiga ini.

```
<?xml version="1.0" encoding="utf-8"?>
<model>
```

```
<project>
	<title>Skenario Tiga</title>
	<author>JePe</author>
	<created>7/25/2004 5:18:36 PM</created>
</project>
<aktors>
	<aktor id="0"  name="staff"  rect="173,131,50,75" >
		<entities>
			<entity>
				<name>Username</name>
				<type>String</type>
			</entity>
			<entity>
				<name>Password</name>
				<type>String</type>
			</entity>
			<entity>
				<name>Nama</name>
				<type>String</type>
			</entity>
			<entity>
				<name>Jabatan</name>
				<type>String</type>
			</entity>
		</entities>
	</aktor>
	<aktor id="1"  name="sekdes"  rect="270,324,50,75" >
		<entities>
			<entity>
				<name>Username</name>
				<type>String</type>
			</entity>
			<entity>
				<name>Password</name>
				<type>String</type>
			</entity>
		</entities>
	</aktor>
	<aktor id="2"  name="kades"  rect="596,318,50,75" >
		<entities>
			<entity>
				<name>Username</name>
				<type>String</type>
			</entity>
			<entity>
				<name>Password</name>
				<type>String</type>
			</entity>
		</entities>
	</aktor>
	<aktor id="3"  name="sekcam"  rect="724,147,50,75" >
		<entities>
			<entity>
				<name>Username</name>
				<type>String</type>
			</entity>
```

```
                        <entity>
                              <name>Password</name>
                              <type>String</type>
                        </entity>
                  </entities>
            </aktor>
            <aktor id="12"  name="camat"  rect="894,53,50,75" >
                  <entities>
                        <entity>
                              <name>Username</name>
                              <type>String</type>
                        </entity>
                        <entity>
                              <name>Password</name>
                              <type>String</type>
                        </entity>
                  </entities>
            </aktor>
      </aktors>
      <activities>
            <activity id="4"  name="Data"  rect="32,37,75,75" >
                  <datafields>
                        <field>
                              <id>1</id>
                              <form>text</form>
                              <name>Nama</name>
                              <type>String</type>
                        </field>
                        <field>
                              <id>2</id>
                              <form>text</form>
                              <name>Alamat</name>
                              <type>String</type>
                        </field>
                        <field>
                              <id>3</id>
                              <form>text</form>
                              <name>Pendidikan</name>
                              <type>String</type>
                        </field>
                        <field>
                              <id>4</id>
                              <form>text</form>
                              <name>Agama</name>
                              <type>String</type>
                        </field>
                        <field>
                              <id>5</id>
                              <form>text</form>
                              <name>Jumlah_Saudara</name>
                              <type>String</type>
                        </field>
                  </datafields>
            </activity>
      </activities>
      <edges>
```

```
            <edge id="5" type="1">
                  <src>aktor,0</src>
                  <des>activity,4</des>
            </edge>
            <edge id="6" type="0">
                  <src>aktor,0</src>
                  <des>aktor,1</des>
            </edge>
            <edge id="7" type="0">
                  <src>aktor,1</src>
                  <des>aktor,2</des>
            </edge>
            <edge id="8" type="0">
                  <src>aktor,2</src>
                  <des>aktor,3</des>
            </edge>
            <edge id="9" type="0">
                  <src>aktor,3</src>
                  <des>aktor,0</des>
            </edge>
            <edge id="10" type="0">
                  <src>aktor,2</src>
                  <des>aktor,0</des>
            </edge>
            <edge id="11" type="0">
                  <src>aktor,1</src>
                  <des>aktor,3</des>
            </edge>
            <edge id="13" type="0">
                  <src>aktor,3</src>
                  <des>aktor,4</des>
            </edge>
      </edges>
</model>
```

Gambar 5.29 Dokumen XML yang terbentuk dari diagram skenario ketiga

Setelah dilakukan proses generate, maka langkah selanjutnya adalah proses kompilasi beberapa file .java yang digenerate oleh aplikasi, serta proses reload web container yang bersangkutan agar aplikasi yang telah terletak di tempat yang semestinya tersebut dapat terbaca oleh web container. Hasil jadi dari proses generate model ini ditunjukkan oleh gambar 5.30.

Gambar 5.30 Tampilan layar login model diagram skenario ketiga

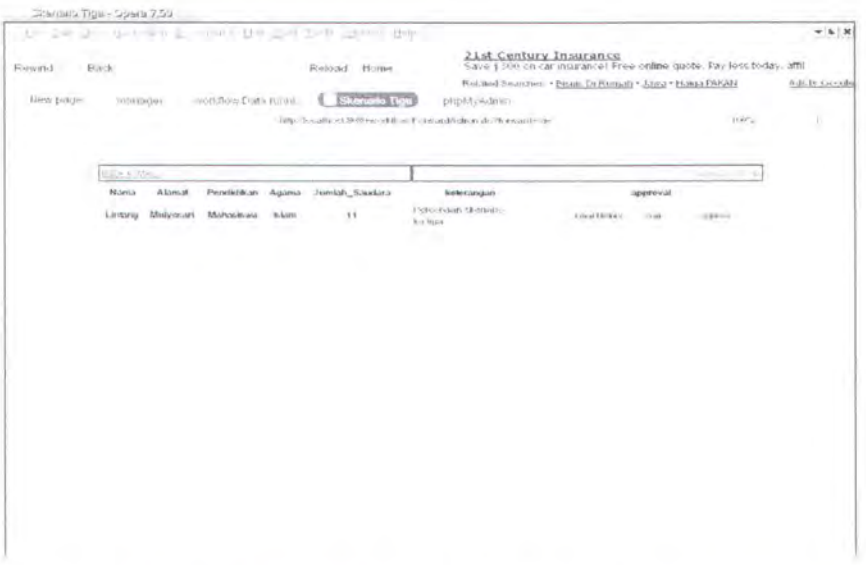
Setelah melakukan proses login, aktor "staff" akan diarahkan menuju halaman berisi pilihan menu. Setelah melakukan proses input data, maka sesuai dengan model diagram yang ada, aktor "staff" hanya dapat melakukan proses disposisi menuju aktor "sekdes". Gambar 5.31 menunjukkan tampilan layar disposisi oleh aktor "staff" menuju "sekdes" tersebut.

Gambar 5.31 Tampilan layer disposisi aktor "staff"

Sedangkan di bagian aktor "sekdes", terdapat dua buah jalur disposisi, yaitu menuju "kades", dan langsung menuju "sekcam". Disposisi data yang berasal dari aktor "staff" kemudian memiliki dua buah alternatif untuk di disposisikan ke aktor selanjutnya. Gambar 5.32 menunjukkan alternative disposisi data untuk dua aktor tersebut.

Gambar 5.32 Disposisi menuju aktor "kades" dan "sekcam".

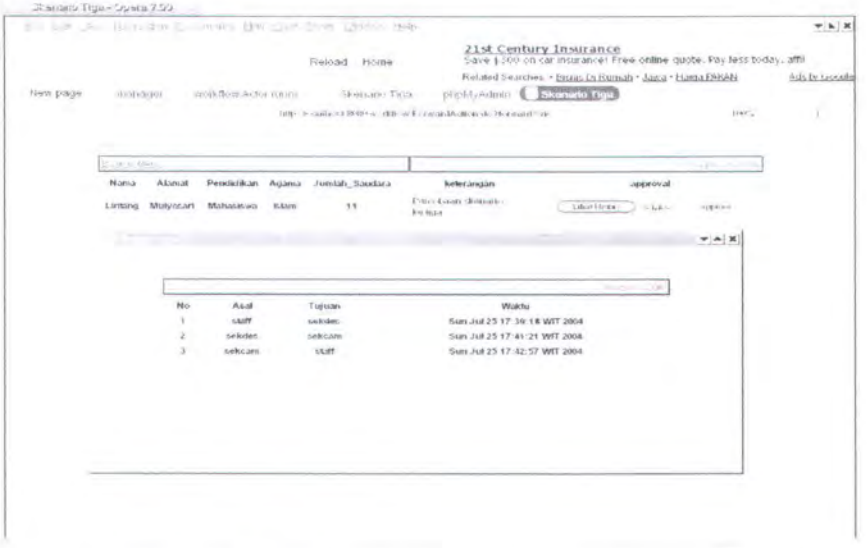
Setelah disposisi data masing-masing sampai pada aktor tujuan, yakni aktor "kades" dan "sekcam". Untuk proses ujicoba, akan dipilih disposisi menuju aktor "staff" yang berasal dari aktor "sekcam" - ditunjukkan oleh gambar 5.33, dan disposisi menuju aktor "sekcam" yang berasal dari aktor "kades" – ditunjukkan oleh gambar 5.34. Hasil akhir dari percobaan disposisi data ini memiliki dua buah muara, yaitu pada aktor "staff" dan aktor "camat".



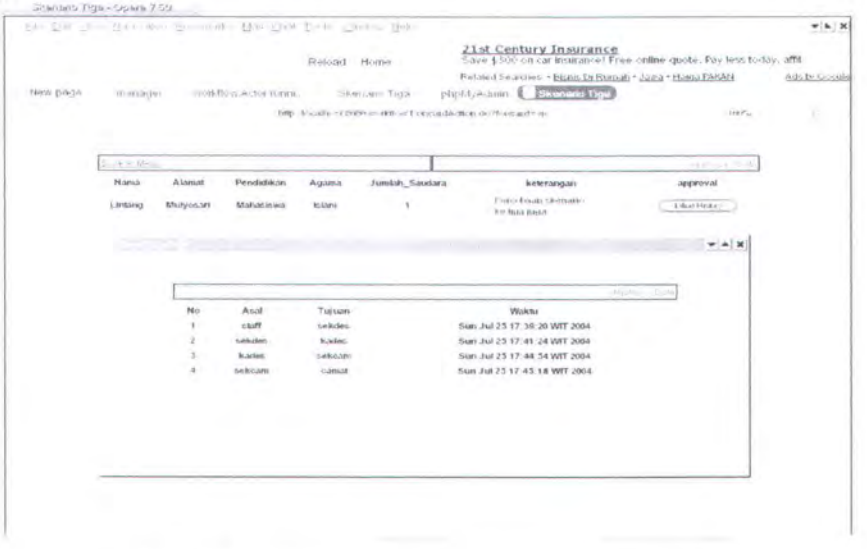
Gambar 5.33 Disposisi dari aktor "sekcam" menuju aktor "staff"

Gambar 5.34 Disposisi dari aktor "kades" menuju "sekcam"

Hasil akhir dari proses disposisi ini dapat dilihat di layer view milik login "staff" pada gambar 5.35 dan layer view milik login "camat" pada gambar 5.36. Di kedua layer view tersebut dapat pula ditampilkan hasil penelusuran history disposisi yang dilalui oleh masing-masing data yang dijadikan percobaan.



Gambar 5.35 Hasil penelusuran history data pada aktor "staff"



Gambar 5.36 Hasil penelusuran history data pada aktor "camat"

5.3 Evaluasi Hasil Uji Coba

Dari hasil ujicoba yang telah dijalankan, aplikasi workflow modeler ini telah dapat mengimplementasikan alur sesuai dengan model yang diinginkan. Data-data

yang didisposisikan bisa sampai dengan tepat menuju aktor yang diinginkan. Selain itu juga dapat dilakukan penelusuran untuk melakukan validasi kiriman data antara aktor satu dengan yang lain.

Beberapa kelemahan terletak pada proses ketika membuka atau melakukan submit pada halaman JSP untuk pertama kali. Hal ini dikarenakan web container yang terlebih dahulu akan mengkompilasi file-file JSP menjadi servlet, untuk kemudian memasuki dapat digunakan untuk melayani request yang masuk, sehingga response yang diharapkan agak lama. Untuk proses yang sama, selanjutnya tidak terdapat delay yang signifikan.

BAB VI
PENUTUP

BAB VI

PENUTUP

Bab ini berisi kesimpulan dan saran bagi pihak-pihak yang akan mengembangkan aplikasi ini lebih lanjut kedepannya

6.1 Kesimpulan

Dari pengerjaan tugas akhir ini didapatkan beberapa kesimpulan sebagai berikut :

1. Implementasi Struts framework dalam aplikasi berbasis web menggunakan teknologi J2EE dapat mempercepat pembangunan suatu halaman web dinamis karena mengimplementasikan MVC. Ini didukung dengan tersedianya fasilitas-fasilitas dari struts yang memungkinkan pembagian tugas yang jelas antara programmer dan desainer aplikasi, seperti struts-tag library yang digunakan dalam aplikasi web yang digenerate.
2. Proses akses data dari bahasa pemrograman ke database menjadi lebih mudah dan cepat menggunakan teknologi *hibernate* yang mengimplementasikan object relational mapping dari object javabean menuju database. Selain itu, hibernate juga menawarkan kemudahan library untuk akses data menggunakan DBMS yang berbeda-beda tanpa harus merubah isi program, hanya dengan setting file *hibernate.properties*.
3. Adanya kemudahan yang didapatkan dalam pembuatan suatu alur data menggunakan workflow editor yang berbasis desktop ini.

6.2 Saran

1. Pengembangan aplikasi workflow ini tidak hanya dapat melakukan disposisi data biasa, namun juga dapat melakukan disposisi file biner.
2. Banyak framework lain untuk diimplementasikan dalam proses pembuatan workflow application, seperti *spring framework*.
3. Penggunaan library lain semacam JDO (Java Data Object) yang lebih ringan dari EJB (Enterprise Java Beans) untuk menangani proses dengan database.
4. Penggunaan teknologi aplikasi web di java lainnya, seperti JSF dan JSTL untuk diimplementasikan dalam kasus workflow application.

DAFTAR PUSTAKA

DAFTAR PUSTAKA

[1] Apache-Struts Jakarta Project, *http://jakarta.apache.org/struts*.

[2] Baeyens, Tom, *The State of Workflow*. , " *http://www.theserverside.com/articles/article.tss?l=Workflow* "

[3] HighTower, Richard, "*Jakarta Struts-Live*", SourceBeat.LLC, 2004.

[4] Hibernate Project, *http://www.hibernate.org*

[5] JDK-1.4. Java API

[6] Rickyanto,Isak, *"Pemrograman Web Dengan Java Servlet"*, Penerbit Andi, 2004.

[8] Struts Community Resources, *http://struts.sourceforge.net*

[9] Workflow Management Coalition, "*http://www.wmfc.org*.

[10] Glosarry, *http://edocs.bea.com/wle/tuxedo/glossary/glossary.htm*

[11] Glosarry, http://*www.movesinstitute.org/~npsnet/v/glossary.html*

[12] Roman, Ed, *Mastering Enterprise JavaBeans 2nd Edition*, Wiley Computer Publishing, 2002.