BAB 3
ANALISIS DAN PERANCANGAN ONTOLOGI *USER-MODEL*

Pada bab ini dijelaskan proses penelitian, analisis skenario personalisasi, ontologi-ontologi yang diperlukan, serta perancangan ontologi *user-model* yang paling sesuai dengan skenario personalisasi.

3.1 Proses Penelitian

Kegiatan penelitian yang dilakukan penulis dimulai dengan melakukan studi bersama pembimbing tugas akhir mengenai bahasa ontologi menggunakan buku "*Semantic Web for the Working Ontologist*." Kegiatan ini dilakukan setelah ditetapkannya teknologi *Semantic Web* untuk *personalized e-learning* sebagai topik besar untuk penelitian ini.

Selanjutnya penulis melakukan observasi dan menetapkan ontologi *user-model* untuk *personalized e-learning* sebagai topik penelitian ini. Penulis melanjutkan studi literatur mengenai ontologi *user-model* dan mendapatkan bahan-bahan pembelajaran mengenai ontologi *user-model*. Kemudian penulis menemukan suatu riset penelitian mengenai suatu sistem *personalized e-learning* bernama LOCO[5]. Dari Kemudian penulis melakukan diskusi dengan salah satu peneliti LOCO, yaitu Jelena Jovanović[6] mengenai penerapan ontologi *user-model* pada sistem LOCO. Kemudian beliau merekomendasikan suatu paper mengenai TANGRAM [9]. TANGRAM adalah proyek pendahulu dari LOCO dan juga menerapkan pendekatan berbasis ontologi seperti LOCO. Selanjutnya penulis mempelajari TANGRAM dan kemudian melakukan perancangan ontologi *user-model* berdasarkan rancangan ontologi *user-model* TANGRAM.

[5] http://iis.fon.rs/LOCO-Analyst
[6] http://jelenajovanovic.net/

Perancangan ontologi *user-model* pada penelitian ini disesuaikan dengan perkiraan kebutuhan *personalized e-learning* di Fasilkom UI. Penulis kemudian merancang perkiraan skenario personalisasi yang dapat terjadi pada sistem *personalized e-learning* yang dikembangkan. Skenario ini berguna sebagai referensi untuk perancangan ontologi *user-model* dan juga diharapkan berguna untuk referensi penelitian selanjutnya. Selanjutnya penulis merancang dan memodifikasi ontologi *user-model* berdasarkan skenario personalisasi. Ontologi ini dikembangkan dengan menggunakan bahasa ontologi OWL. Penulis menggunakan *tool* Protégé sebagai ontologi *editor*. Setelah melakukan perancangan ontologi *user-model* penulis membuat suatu prototipe untuk mensimulasikan ontologi *user-model* tersebut. Prototipe ini dikembangkan menggunakan aplikasi *web* PortalCore yang akan lebih dijelaskan selanjutnya.

3.2 Skenario Personalisasi

Skenario personalisasi pada *personalized e-learning* dapat dikelompokkan menjadi tiga skenario berdasarkan fungsi personalisasinya, yaitu:

- Perekomendasi Kuliah;
- Personalisasi Pembelajaran Berdasarkan *Learning Style*;
- Pembelajaran Berdasarkan *Performance.*

3.2.1 Perekomendasi Kuliah

Informasi mengenai performa kuliah mahasiswa disimpan pada ontologi *performance*. Informasi-informasi itu adalah Indeks Prestasi Kumulatif (IPK), Indeks Prestasi Semester (IPS), mata kuliah yang diambil beserta nilai tiap kuliah tersebut. Sistem kemudian mengolah informasi tersebut untuk merekomendasikan jumlah mata kuliah dan kuliah apa saja yang sebaiknya diambil mahasiswa. Jumlah mata kuliah yang dapat diambil disesuaikan dengan nilai IPS pada semester terakhirnya. Setiap mata kuliah dapat memiliki suatu prasyarat mata kuliah. Dengan informasi yang sudah terstrukstur dalam ontologi, sistem dapat mengerti hubungan informasi yang dibutuhkan dengan prasyarat jumlah mata

kuliah atau prasyarat mengambil mata kuliah tertentu, hubungan antar mata kuliah, dll. TANGRAM menggunakan teknik penggunaan *link* keterangan dan penyembunyian *link* (*annotated link and link hiding*) untuk perekomendasi. *Link annotation* yang digunakan pada sistem TANGRAM adalah sebagai berikut:

- *Link* berwarna biru adalah *link* kepada domain konsep yang sudah dimengerti oleh mahasiswa.
- *Link* berwarna hijau adalah *link* menuju domain konsep yang belum dipelajari mahasiswa namun mahasiswa tersebut sudah memiliki pengetahuan tentang topic-topik prasyaratnya.
- *Link* berwarna merah adalah *link* kepada domain konsep yang belum dipelajari mahasiswa dan mahasiswa tersebut belum memenuhi topic-topik prasyaratnya.

Link hiding digunakan untuk mencegah mahasiswa mengakses topik kuliah tingkatannya masih terlalu tinggi baginya, dengan kata lain *link annotation* yang berwarna merah dibuat tidak aktif. Mahasiswa boleh mengakses topik yang link annotatednya berwarna biru dan hijau.

Teknik *link annotated* dan *link hiding* ini dapat kita gunakan pada sistem *personalized e-learning* di Fasilkom UI. Teknik-teknik ini dapat digunakan pada proses pendaftaran mata kuliah. Link hijau untuk mata kuliah yang dapat diambil mahasiswa dan mahasiswa tersebut sudah memenuhi prasyarat mata kuliah tersebut. Link merah diberikan untuk mata kuliah yang belum boleh diambil oleh seorang mahasiswa karena dia belum memenuhi prasyarat mata kuliah. Prasyarat-prasyarat mata kuliah tersebut antara lain:

1. Prasyarat harus lulus atau sudah mengambil mata kuliah tertentu;
2. Prasyarat jumlah SKS yang sudah diambil mahasiswa. Contohnya untuk mata kuliah KP memiliki prasyarat sudah mengambil 100 SKS;
3. Prasyarat batasan jumlah SKS yang boleh diambil pada semester tersebut.

Sistem dapat mengambil informasi untuk rekomendasi kuliah ini dari ontologi *performance* mahasiswa. Dapat disimpulkan dari pembahasan ini untuk sistem perekomendasi kuliah, komponen yang perlu tersedia dalam ontologi *performance* adalah: IPK, IPS, prasyarat mata kuliah, dan nilai mata kuliah.

3.2.2. Personalisasi Pembelajaran Berdasarkan *Learning Style*

Sistem *personalized e-learning* diharapkan dapat menyajikan konten pembelajaran yang disesuaikan dengan gaya belajar tiap pembelajar. Oleh karena itu dalam ontologi *user model* diperlukan komponen *learning style*. Ontologi *user-model* ini memungkinkan kita untuk menambahkan kategori *learning style* dari sumber teori yang berbeda.

Ontologi *user model* yang dikembangkan pada tugas akhir ini menggunakan teori *learning style* Felder-Silverman. Seperti yang sudah dijelaskan pada Bab 2, Felder-Silverman mengelompokkan *learning style* dalam lima kategori yaitu: *active-reflective*, *visual-verbal*, *sensing-intuitive*, *sequential-global*, dan *inductive-deductive*.

Kita dapat memilih salah satu atau beberapa kategori yang dianggap paling relevan untuk kebutuhan personalisasi. Dengan kata lain kita boleh saja tidak menggunakan salah satu kategori *learning style* jika dianggap belum sesuai fitur personalisasi yang ingin lebih dahulu dikembangkan.

Untuk pembahasan skenario personalisasi pembelajaran ini penulis merekomendasikan personalisasi berdasarkan kategori *visual-verbal*. Penulis berpikir bahwa personalisasi berdasarkan kategori ini cocok untuk menyajikan jenis media dan materi yang paling sesuai dengan pembelajar. *Visual learner* akan mendapatkan materi pembelajaran yang lebih banyak mengandung unsur *image* seperti diagram dan gambar dibandingkan kata-kata. Jenis media video klip juga cocok untuk *visual learner*. Berkebalikan dengan itu *verbal learner* akan mendapatkan materi pembelajaran yang kata-kata tertulis. Jenis media audio klip juga cocok untuk *verbal learner* oleh karena *verbal learner* cepat menangkap informasi dari kalimat, baik yang tertulis maupun yang dibacakan.

Alur personalisasi pembelajaran berdasarkan *learning style* melewati beberapa tahapan. Pada awal menggunakan sistem, pembelajar perlu mengerjakan semacam

kuis psikologi untuk menentukan nilai dari setiap *learning style* kategori. Hasil dari kuis ini akan dimasukkan sebagai nilai-nilai *learning style* pembelajar pada setiap kategorinya. Kemudian sistem dapat mengolah informasi *learning style* untuk selanjutnya melakukan perangkaian konten pembelajaran yang paling cocok dengan *learning style* pembelajar ini.

Nilai dari setiap kategori *learning style* dibatasi dalam rentang nilai -1 hingga 1. Batas nilai -1 dan 1 menunjukkan kecenderungan paling tinggi dari setiap kategori. Misalkan nilai -1 pada kategori *visual-verbal* menunjukkan bahwa pembelajar ini berkarakteristik pembelajar yang sangat *visual*. Berkebalikan dengan itu nilai 1 menunjukkan pembelajar yang sangat *verbal*.

3.2.3. Personalisasi Pembelajaran Berdasarkan *Performance*

Personalisasi berdasarkan *Performance* ini didasarkan pada pemikiran bahwa sistem menyediakan materi pembelajaran sesuai dengan tingkat kemampuan pembelajar sehingga pembelajaran pada tingkat yang lebih lanjut akan lebih efektif. Berdasarkan pemikiran ini materi pembelajaran pada bab selanjutnya yang didapatkan oleh pembelajar yang nilai test bab sebelumnya kurang baik akan berbeda dengan pembelajar yang nilai testnya jauh lebih baik.

Pada ontologi *Performance* - yang menyimpan informasi performa kuliah seorang pembelajar - diperlukan komponen mengenai mata kuliah yang sedang diambil pembelajar tersebut. Pada ontologi ini juga diperlukan komponen yang menyimpan hasil setiap test pada kuliah tersebut dan juga hasil tugas kuliah. *Property* mata-kuliah prasyarat juga perlu ditambah pada komponen mata-kuliah agar sistem dapat mengerti konten pembelajaran dari kuliah lain yang berhubungan dengan mata kuliah yang sedang diambil. Pada ontologi *user model* kemudian diperlukan *property hasPerformance* antara *Learner* dengan ontologi *Performance* yang menunjukkan seorang pembelajar memiliki objek *Performance*.

Alur personalisasi berdasarkan *Performance* melalui beberapa tahapan. Pada awal sesi pembelajaran pembelajar perlu mengisi kuis pra-kuliah untuk mengetahui tingkat pengetahuan pembelajar berkaitan dengan mata kuliah yang diambil kemudian hasil ini disimpan sebagai hasil test kuis pra-kuliah pada ontologi *Performance* pembelajar. Kemudian sistem merangkai konten pembelajaran sesuai kuliah yang diambil, *learning style*, dan hasil kuis pra-kuliah. Pada sesi pembelajaran selanjutnya setiap hasil test, tugas, dan lain-lain disimpan pada *Performance* pembelajar dan kemudian sistem menganalisis hasil-hasil tersebut untuk merangkai konten pembelajaran pada bab selanjutnya yang paling cocok untuk pembelajar tersebut.

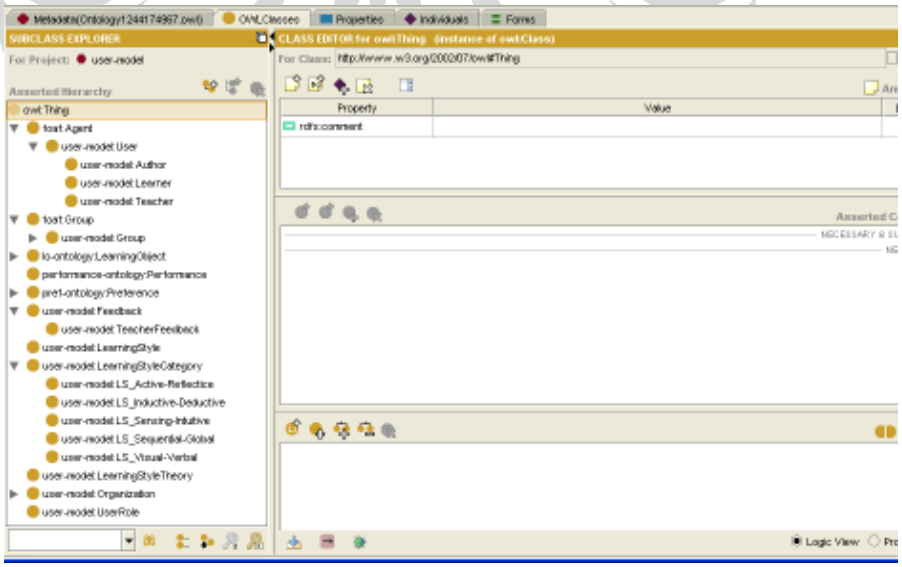
3.3 Perancangan Ontologi *User-Model*

Perancangan ontologi *user-model* pada tugas akhir ini penulis terinspirasi dari *user-model* yang dikembangkan untuk sistem *personalized e-learning* bernama TANGRAM [9]. Sebelumnya penulis melakukan studi literatur dengan mencari *paper* mengenai ontologi *user-model* untuk *personalized e-learning* dan pada akhirnya menemukan proyek riset TANGRAM. Penulis melakukan diskusi melalui *e-mail* dengan Jelena Jovanovic yang merupakan seorang peneliti dari proyek TANGRAM ini dan dari beliau penulis mendapatkan *paper* mengenai *personalised e-learning* yang dikembangkannya. Berdasarkan paper itu [9] penulis mendapat gambaran mengenai *personalised e-learning* berbasis ontologi untuk kemudian diadaptasi sesuai dengan kondisi perkuliahan di Fasilkom UI. Ontologi *user-model* pada tugas akhir ini merupakan penggunaan ulang dari rancangan ontologi *user-model* TANGRAM [9] yang diadaptasi sesuai perkiraan kebutuhan sistem *personalized e-learning* di Fasilkom UI. Gambar 2.3 menunjukkan representasi diagram dari ontologi *user-model* TANGRAM.

Penulis menggunakan *tool* Protege_3.4_rc2 untuk membuat ontologi *user-model*. Protégé adalah *tool ontology editor* yang dikembangkan oleh Universitas Stanford. Sebagai ontologi *editor,* Protégé ini merupakan *tool* yang *powerful*.

Penulis sangat merekomendasikan Protégé untuk digunakan sebagai ontologi *editor*.

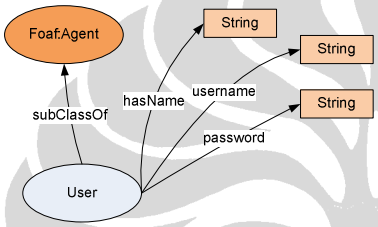
Protégé memiliki berbagai *Tab Widgets* yang memiliki fungsi masing-masing untuk pengembangan suatu proyek ontologi. *Tab Widgets* ini dapat ditambahkan dan dikurangi pada menu *configure*. Kita juga dapat mengunduh *plugin Tab Widgets* yang lain di internet. Lima *Tab Widgets* utama pada Protégé adalah *OWLMetadata Tab*, *OWLClasses Tab*, *OWLProperties Tab*, *OWLIndividual Tab*, dan *OWLForms Tab*. *OWLClasses Tab* berguna untuk menambah atau mengurangi *class* pada ontologi. Gambar 3.1 menunjukkan visualisasi dari *OWLClassesTAB*. Pada gambar tersebut ditampilkan *class-class* pada ontologi *user-model* di tugas akhir ini. *OWLProperties Tab* berguna untuk *editor property* dari ontologi. *OWLMetadata Tab* berguna untuk meng-*import* ontologi, membuat ontologi baru, menyiapkan *namespace* dan *prefix* yang digunakan pada ontologi. *OWLindividual Tab* untuk menambahkan atau mengurangi individual dari suatu *class* sedangkan *OWLForms Tab* berguna untuk mengatur tampilan *form instance* suatu *class*.



Gambar 3.1 *OWLClassesTAB*

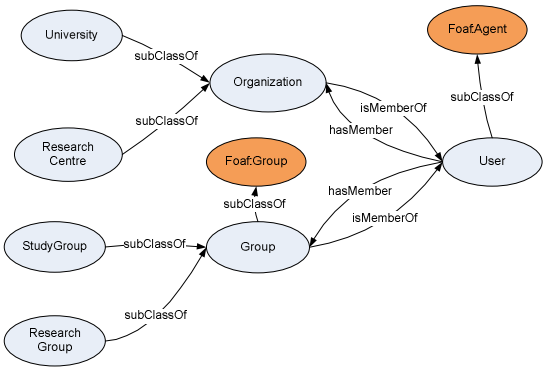
Diagram keseluruhan dari ontologi *user-model* dapat dilihat pada LAMPIRAN A. Sedangkan *source code* dari ontologi *user-model* yang dikembangkan dapat

dilihat pada LAMPIRAN B. Untuk memperjelas, penjelasan mengenai ontologi diagram ini akan dipecah menjadi bagian-bagian kecil. Pada ontologi *user-model* ini terdapat *class User* yang mendeskripsikan pengguna dari sistem. *User* didefinisikan sebagai *subclass* dari *Class Agent* pada ontologi FOAF (*Friend Of A Friend*). FOAF adalah format untuk mendukung deskrispsi terdistribusi tentang orang dan relasinya [1]. Pada class *User* ditambahkan *datatype property hasName,username,* dan *password*. *HasName* menunjukkan nama dari seorang *User* sedangkan *username* dan *password* menunjukkan *username* dan *password* dari User. Ketiga *property* ini bertipekan *String*. Diagramnya dapat kita lihat pada Gambar 3.2.



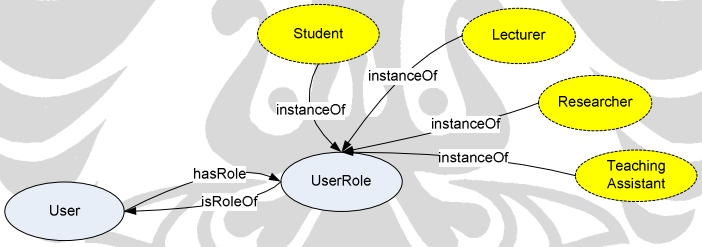
Gambar 3.2 *User*

Setiap *User* dapat merupakan anggota dari suatu Organisasi (*Class Organization*). Lebih spesifiknya seorang *User* dapat merupakan anggota dari suatu universitas (*Class University*) dan atau suatu pusat penelitian (*Class ResearchCentre*). Selain itu seorang *User* dapat juga merupakan anggota dari satu atau beberapa grup (*Class Group*) yaitu kelompok studi (*Class StudyGroup*) dan kelompok riset (*Class ResearchGroup*). *Class Group* didefinisikan sebagai *subclass* dari *class Group* pada ontologi FOAF. *Property isMemberOf* merepresentasikan *User* merupakan anggota dari suatu *Organization* atau *Group* sedangkan *property hasMember* merupakan invers dari *isMemberOf*. *Property hasMember* menunjukkan anggota-anggota dari suatu *Organization* atau *Group*. Diagramnya dapat dilihat pada Gambar 3.3.



Gambar 3.3 *hasMember*

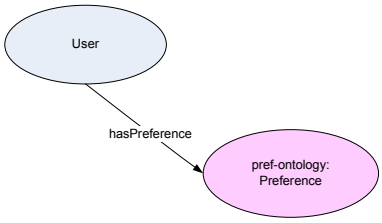
Class UserRole seperti dapat dilihat pada Gambar 3.4 menunjukkan peran atau posisi seorang *User* pada organisasinya. *Property hasRole* menunjukkan seorang *User* memiliki peran tertentu dan *property isRoleOf* merupakan invers dari *hasRole*. *Class UserRole* ini memiliki empat obyek yaitu *Student, Lecturer, Teaching_Assistant, dan Researcher*. Seorang *User* nantinya akan memiliki satu dari empat obyek *UserRole* ini.



Gambar 3.4 *UserRole*

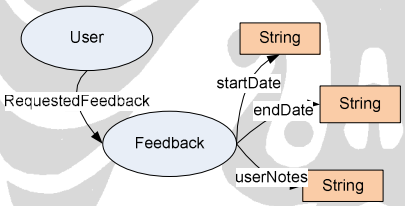
Seperti dapat dilihat pada Gambar 3.5, *User* memiliki *property hasPreference* yang menunjukkan preferensi *User*. *Class Preference* ini adalah *class* di luar ontologi *user-model*. Sistem TANGRAM menggunakan ontologi ims-lip-elements.owl yang sesuai dengan spesifikasi IMS LIP untuk merepresentasikan preferensi domain topik, bahasa, dan juga konten pembelajaran. Oleh karena ontologi yang berkaitan dengan *Preference* ini berada di luar ruang lingkup tugas akhir ini, penulis menyerahkan kepada peneliti-peneliti selanjutnya untuk mengobservasinya. Pada ontologi *user-model* ini penulis menggunakan *prefix* "*pref-ontology*:" untuk menunjukkan ontologi yang berhubungan dengan

Preference. Salah satu kegunaan *Preference* ini adalah agar sistem dapat mengerti preferensi konten pembelajaran yang paling sering digunakan *User*.



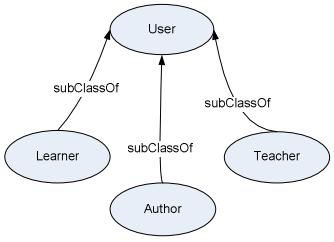
Gambar 3.5 *Preference*

User memiliki *property RequestedFeedback* yang menunjukkan permohonan *feedback*. Seperti dapat dilihat pada Gambar 3.6, *RequestedFeedback* menghubungkan *class User* dengan *class Feedback*. *Datatype property startDate* dan *EndDate* menunjukkan rentang waktu *feedback* dimohonkan hingga dibuat. *Property userNotes* menghubungkan *instance* dari *Feedback* dengan catatan yang dibuat oleh *User* saat atau setelah mereview *feedback* tersebut.



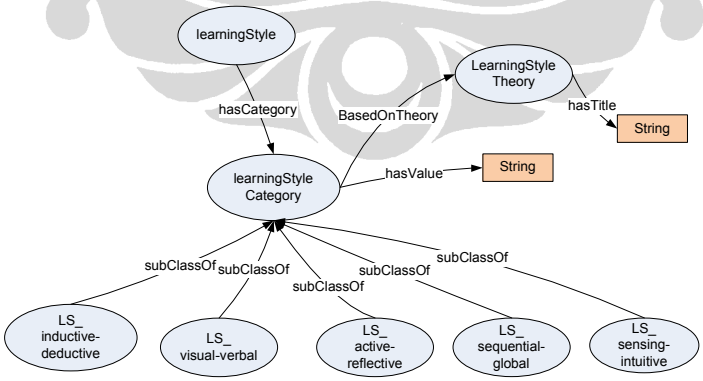
Gambar 3.6 *Feedback*

Class User terbagi menjadi tiga *subclass* yaitu *Learner*, *Teacher*, dan *Author*. Diagramnya dapat dilihat pada Gambar 3.7. *Learner* merupakan representasi dari pembelajar, *Teacher* merupakan pengajar atau dosen, sedangkan *Author* adalah seorang pembuat atau penulis dari suatu konten pembelajaran.



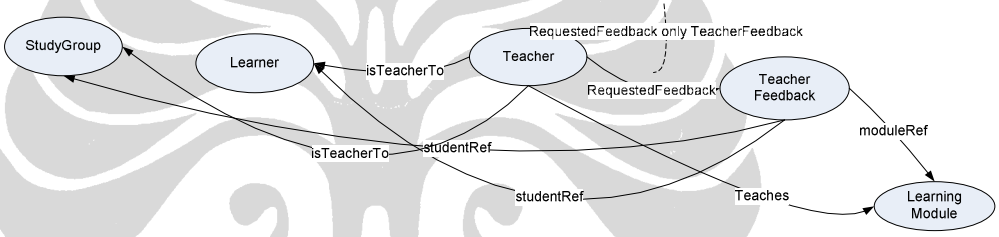
Gambar 3.7 *Subclasses of User*

Seorang *Learner* memiliki *property hasPerformance* dan *hasLearningStyle*. *Property hasPerformance* menghubungkan ontologi *user-model* dengan ontologi *performance* yang menyimpan data performa perkuliahan pembelajar. Untuk itu diperlukan suatu ontologi *performance* yang cocok dengan proses perkuliahan di Fasilkom UI. Pembuatan ontologi *performance* ini di luar ruang lingkup tugas akhir ini. TANGRAM sendiri menggunakan ontologi PAPI *Learner* untuk merepresentasikan performa pembelajar.*Property hasLearningStyle* berasosiasi dengan *LearningStyle* seorang *Learner*. *Class LearningStyle* memiliki *property hasCategory* untuk menghubungkannya kepada *class LearningStyleCategory*. Seperti sudah dijelaskan pada pembahasan skenario personalisasi, ontologi *user model* ini menggunakan teori *Learning Style* Felder-Silverman. *Property BasedOnTheory* ditambahkan pada *class LearningStyleCategory* agar memungkinkan untuk menambahkan kategori *learning style* yang berdasarkan teori yang berbeda. *Class LearningStyleTheory* memiliki *datatype property hasTitle* yang obyeknya bertipe *String* untuk memberikan judul dari teori yang digunakan. *Class LearningStyleCategory* memiliki *datatype property hasValue* yang bertipe *Float* untuk memasukkan nilai dari setiap kategori *learning style. Class LearningStyleCategory* terbagi menjadi lima *subclass* yaitu *LS_Active-Reflective*, *LS_Inductive-Deductive*, *LS_Sensing-Intuitive*, *LS_Sequential-Global*, dan *LS_Visual-Verbal*. Kelima *class* ini *BasedOnTheory* Felder-Silverman. Diagramnya dapat dilihat pada Gambar 3.8.



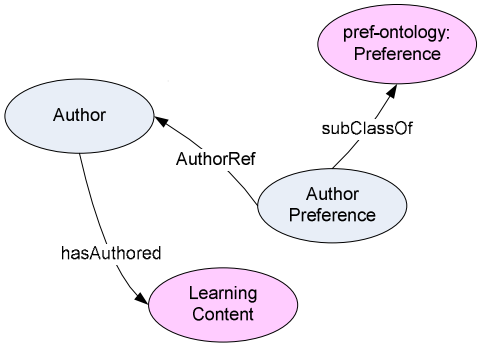
Gambar 3.8 *LearningStyle*

Seorang *Teacher* memiliki *property isTeacherTo* untuk menunjukkan dia mengajar siapa. Seperti dapat dilihat pada Gambar 3.9, *range* dari *isTeacherTo* ini adalah *Learner* dan *StudyGroup* yang berarti seorang *Teacher* dapat menjadi pengajar bagi seorang *Learner* atau bagi satu *Study Group*. *Property Teaches* menunjukkan modul pembelajaran apa yang dia ajarkan (*class LearningModule*). *Class LearningModule* ini merupakan *subclass* dari *class LearningObjective* pada ontologi *Learning Object*. Khusus untuk *Teacher*, *property RequestedFeedback* akan menunjuk kepada *TeacherFeedback* yang merupakan *subclass* dari *Feedback*. *Class TeacherFeedback* memiliki *property StudentRef* yang menunjukkan kepada siapa atau studi grup apa *feedback* ini direferensikan. *Property moduleRef* pada *TeacherFeedback* menunjukkan berkaitan dengan *LearningModule* apakah *feedback* ini.



Gambar 3.9 *TeacherFeedback*

Class Author merepresentasikan pembuat atau penulis dari suatu konten pembelajaran. Seperti dapat dilihat pada Gambar 3.10, *Author* memiliki *property hasAuthored* yang menunjukkan konten pembelajaran apa saja yang sudah dia buat. *Class AuthorPreference* yang merupakan *subclass* dari *class Preference* memiliki *property AuthorRef* untuk menunjukkan *author* yang menjadi preferensi seorang *User*.



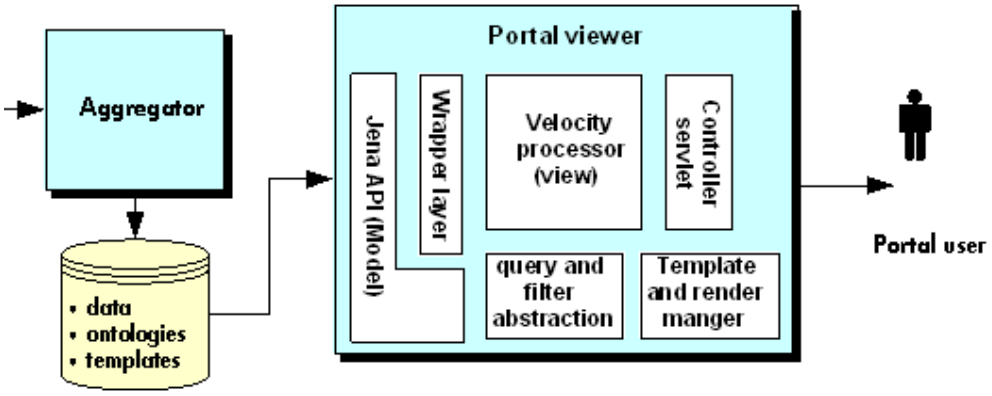
Gambar 3.10 *Author*

BAB 4
IMPLEMENTASI ONTOLOGI *USER-MODEL*

Pada bab ini akan dijelaskan implementasi ontologi *user-model* pada PortalCore dan hasil pengembangan prototipe *user-model* menggunakan PortalCore.

4.1 Implementasi Ontologi *User-Model* Pada Aplikasi PortalCore

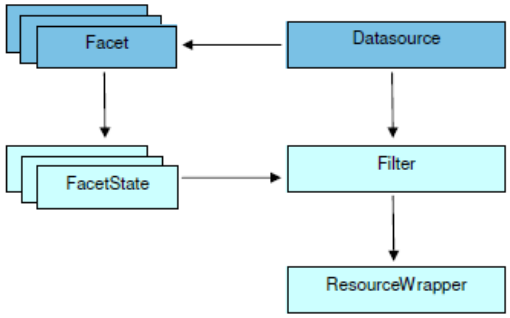
Setelah selesai membuat ontologi *user-model,* penulis melakukan implementasi ontologi tersebut dengan membuat prototipe menggunakan aplikasi PortalCore. Prototipe ini bertujuan untuk mensimulasikan ontologi *user-model* pada suatu aplikasi *web*. PortalCore sendiri merupakan aplikasi *open-source semantic portal* dari proyek SWAD-E (*Semantic Web Advanced Development for Europe*) yang mengembangkan portal SWED (*Semantic Web Environmental Directory*). Portal SWED ini bertujuan sebagai *directory* bagi lintas komunitas yang terdiri atas organisasi lingkungan (*wildlife, environmental and biodiversity*). Dengan menggunakan Jena *framework, portal* SWED telah dijadikan *portal generator tool* yang bersifat *open-source*. *Tool* itulah yang disebut PortalCore yang digunakan pada penelitian ini. Jena sendiri adalah *framework* berupa API (*Application Programming Interface*) untuk *Semantic Web* yang berbasis Java dan sudah banyak digunakan dalam pengembangan aplikasi berbasis ontologi. Gambar 4.1 menunjukkan struktur PortalCore. Terlihat komponen utama dari PortalCore adalah *Portal viewer* yang menerima *input* data, ontologi, dan *templates*.



Gambar 4.1 Struktur PortalCore [17]

Struktur PortalCore mengadopsi rancangan MVC (*Model-View-Controller*). *Model* berupa *Class* Java yang menggunakan *library Jena* untuk membungkus data (ontologi dan *instance* data) yang dapat berasal dari *multiples files* atau *database*. *View* berupa *Velocity template engine* untuk *generate* tampilan halaman *portal*. Sedangkan *Controller* berupa *java servlets* dengan sejumlah *built in actions*.

Model objects yang tertutama pada *portal viewer* terdiri dari *datasource, facet, filter,* dan *resource wrappers*. *Datasource* adalah suatu definisi sumber data dan konfigurasi *portal*. *Facets* adalah atribut-atibut yang digunakan dalam pencarian informasi pada *portal*. *FacetState* atau nilai atribut menjadi *filter* untuk menampilkan hasil pencarian. Templates berguna untuk tampilan data dengan menggunakan *resource wrappers*. Hubungan antara *model objects* tersebut dapat dilihat pada Gambar 4.2.



Gambar 4.2 Hubungan Antara *Model Objects*

Pada tugas akhir ini PortalCore dijalankan pada *web server Apache Tomcat 5.5*. Alur pengerjaan prototipe ontologi *user-model* ini terbagi menjadi tiga kegiatan, yaitu:

1. *Input* data PortalCore
   Terdiri atas persiapan data dan pendefinisan *rules*.
2. Konfigurasi PortalCore
   Terdiri atas pendefinisian *datasource*, *facet*, dan *template*.
3. Tampilan PortalCore
   Terdiri atas pembuatan *template* untuk menampilkan data.

4.2 *Input* PortalCore

Data pada PortalCore terdiri dari dua jenis yaitu ontologi berupa *files* ontologi OWL dan *instances* data dalam format .n3. Selain kedua jenis data tersebut, kita dapat melakukan *inference* untuk mendapatkan data atau relasi baru dengan cara membuat beberapa *rules*. Data dan *rules* ini yang akan diproses sebagai *input* PortalCore.

4.2.1 Persiapan Data

PortalCore membutuhkan data ontologi dan *instances* datanya. Data ontologi menggunakan ontologi *user-model.* Selanjutnya penulis menyiapkan *instances* data dari ontologi *user-model* ini. Penulis membuat data-data yang merepresentasikan individu-individu dari *classes* pada ontologi *user-model* ini. PortalCore menggunakan data dalam format RDF, khususnya sintaks N3.

Persiapan pertama adalah mempersiapkan data mengenai pengguna sistem atau *instance* dari class *User. Instance Class User* memiliki 3 *subclass* yaitu *Learner*, *Teacher*, dan *Author*. *Instance* dari tiap *User* berdasarkan tiap jenis *subclass*-nya. Penulis membuat delapan *instance User* yang terdiri dari enam *Learner,* satu *Teacher*, dan satu *Author*. Data *user* ini disimpan pada *file* datatestuser.n3.

Instance dari *class Learner* merepresentasikan pembelajar dalam sistem *personalized e-learning*. *Object* dari *Learner* ini berkaitan dengan informasi personal pembelajar, *learning style*-nya, anggota apakah dia, dan apakah perannya di organisasinya. Berikut ini contoh *code instance data* dari *Learner.*

```
user-model:User2
 a              user-model:Learner;
 user-model:hasName "Yohanes Immanuel";
 user-model:username "yoim50";
 user-model:password "test123";
 user-model:hasLearningStyle user-model:LS_2;
 user-model:hasRole user-model:user_role_Student;
 user-model:isMemberOf user-model:um_org_04;
 user-model:isMemberOf user-model:um_org_01.
```

Object User2 ini merupakan merupakan *instance* dari *class Learner*. Semua *subclass* dari *User* memiliki *property hasName*, *username*, dan *password* yang merupakan informasi personal dari seorang *user*. *Property hasLearningStyle* merujuk kepada *LS_2* yang merupakan *instance* dari *class LearningStyle*. *Property hasLearningStyle* merupakan *property* yang *functional* yang menunjukkan setiap *Learner* memiliki satu *Learning Style*. Setiap *object LearningStyle* merujuk khusus kepada seorang *Learner* sehingga pada saat pendefinisian suatu *object Learner* penulis sekaligus mendefinisikan *object LearningStyle* milik *Learner* tersebut. Berikut ini *code* dari *object LS_2.*

```
user-model:LS_2
 a user-model:LearningStyle;
 user-model:hasCategory user-model:LS_Act-Ref_2;
 user-model:hasCategory user-model:LS_Ind-Dec_2;
 user-model:hasCategory user-model:LS_Sen-Int_2;
 user-model:hasCategory user-model:LS_Seq-Glo_2;
 user-model:hasCategory user-model:LS_Vis-Ver_2;.

 user-model:LS_Act-Ref_2
 a user-model:LS_Active-Reflective;
 user-model:BasedOnTheory user-model:felder-silverman;
 user-model:hasValue "0.7".

user-model:LS_Ind-Dec_2
 a user-model:LS_Inductive-Deductive;
 user-model:BasedOnTheory user-model:felder-silverman;
 user-model:hasValue "-0.4".
```

```
user-model:LS_Sen-Int_2
 a user-model:LS_Sensing-Intuitive;
 user-model:BasedOnTheory user-model:felder-silverman;
 user-model:hasValue "-0.5".

user-model:LS_Seq-Glo_2
 a user-model:LS_Sequential-Global;
 user-model:BasedOnTheory user-model:felder-silverman;
 user-model:hasValue "0.3".

user-model:LS_Vis-Ver_2
 a user-model:LS_Visual-Verbal;
 user-model:BasedOnTheory user-model:felder-silverman;
 user-model:hasValue "-0.3".
```

Property hasCategory merujuk kepada kelima kategori *Learning Style* Felder. Setiap *object* kategori ini adalah khusus untuk *User2.* Setiap kategori ini memiliki nilai *hasValue* yang menunjukkan karakteristik *Learning Style User2*. *User2* ini merupakan *member* dari organisasi *um_org_01* dan um_org_04 yang ditunjukkan oleh *property isMemberOf*.

Instance dari *class Teacher* merepresentasikan seorang pengajar atau dosen suatu kuliah. Berikut ini adalah contoh *code instance Teacher*.

```
user-model:User7
 a              user-model:Teacher;
 user-model:hasName "Siti Aminah";
 user-model:username "ami47";
 user-model:password "test123";
 user-model:hasRole user-model:user_role_Lecturer;
 user-model:isTeacherTo user-model:User1;
 user-model:isTeacherTo user-model:User2;
 user-model:isTeacherTo user-model:User3;
 user-model:isTeacherTo user-model:User4;
 user-model:isTeacherTo user-model:User5;
 user-model:isTeacherTo user-model:User6;
 user-model:isTeacherTo user-model:Personalized_Elearning_Group;
 user-model:isTeacherTo user-model:Database_Group;
 user-model:Teaches user-model:tugas_akhir_1;
 user-model:RequestedFeedback user-model:teacher_feedback_1;
 user-model:isMemberOf user-model:um_org_01.
```

Object User7 pada contoh ini memiliki peran (*property hasRole*) sebagai *Lecturer*. *Property isTeacherTo* merujuk kepada pembelajar dan atau *study* group yang dia ajar. *Property Teaches* menunjukkan kuliah atau modul pembelajaran yang dia ajarkan. *Property RequestedFeedback* merujuk kepada *TeacherFeedback.*Berikut ini salah satu contoh *code instance TeacherFeedback*.

```
user-model:teacher_feedback_1
 a user-model:TeacherFeedback;
 user-model:startDate "22 Juni 2009";
 user-model:endDate "3 Juli 2009";
 user-model:moduleRef user-model:  TBA _1;
 user-model:studentRef user-model:User1;
 user-model:userNotes "kumpulkan laporan yaaa ^^".
```

Suatu *instance* dari *class TeacherFeedback* memiliki property *datatype startDate* dan *endDate* yang menunjukkan rentang waktu *feedback* ini. *Property moduleRef* merujuk kepada modul pembelajaran apakah *feedback* ini. *Property studentRef* merujuk kepada kepada pengguna yang manakah *feedback* ini. *Property userNotes* catatan *user* saat atau setelah mereview *feedback*.

Instance dari *Class Author* merepresentasikan seorang pembuat/penulis dari suatu konten pembelajaran. *Author* memiliki *property* hasAuthored untuk menunjukkan konten pembelajaran yang dia buat. Berikut ini adalah contoh *code instance Author*.

```
user-model:User8
 a user-model:Author;
 user-model:hasName "Eko S";
 user-model:username "Eko17";
 user-model:password "test123";
 user-model:hasRole user-model:user_role_Researcher;
 user-model:hasAuthored "LC_PBK_Chap_3";
 user-model:isMemberOf user-model:um_org_01.
```

Pada *file datatestother.n3* penulis memasukkan *instances* class yang lainnya. Salah satu alasan penulis menyimpan *instances data* ini pada *file* berbeda adalah karena umumnya *instances data classes* ini dapat berelasi dengan lebih dari satu *user*. *Classes* itu adalah *class University, ResearchCentre, UserRole, StudyGroup, ResearchGroup,* dan *LearningStyleTheory*.

Penulis membuat tiga *instances University*. *Object um_org_01* merepresentasikan Universitas Indonesia, *um_org_02* adalah ITB, sedangkan *um_org_03* adalah ITS. Berikut ini *code* dari ketiga *instance* tersebut.

```
user-model:um_org_01
 a user-model:University;
 user-model:hasTitle "Universitas Indonesia";
 .
user-model:um_org_02
 a user-model:University;
 user-model:hasTitle "ITB";
 .
user-model:um_org_03
 a user-model:University;
 user-model:hasTitle "ITS";
```

Penulis membuat tiga *instances ResearchCentre*. Berikut ini adalah *code* dari ketiga *instances* tersebut.

```
user-model:um_org_04
 a user-model:Research_Centre;
 user-model:hasTitle "MIC Lab";
 .
user-model:um_org_05
 a user-model:Research_Centre;
 user-model:hasTitle "Distance Learning Lab";
 .
user-model:um_org_06
 a user-model:Research_Centre;
 user-model:hasTitle "RISTEK BEM FASILKOM";
 .
```

Instances class UserRole pada ontologi *user-model* ini ditentukan menjadi empat *object* yaitu user_role_Lecturer, user_role_Student, user_role_Researcher, user_role_Teaching_Assistant. Pada waktu kemudian boleh saja untuk menambah atau mengganti *object UserRole* ini sesuai kebutuhan. Berikut ini *code* dari *instances UserRole*.

```
user-model:user_role_Lecturer
 a user-model:UserRole;
 rdfs:label "Lecturer";
 .
user-model:user_role_Researcher
 a user-model:UserRole;
 rdfs:label "Researcher";
 .
```

```
user-model:user_role_Student
 a user-model:UserRole;
 rdfs:label "Student";
 .
user-model:user_role_Teaching_Assistant
 a user-model:UserRole;
 rdfs:label "Teaching Assistant";
 .
```

Selain *instances* yang sudah dibahas sebelumnya, penulis juga membuat empat *instance* lainnya yang masing-masing merupakan *instance* dari *class LearningModule*, *LearningStyleTheory*, *ResearchGroup*, dan *StudyGroup*. Berikut ini adalah ke-empat *instance* tersebut.

```
user-model:tugas_akhir_1
 a user-model:LearningModule;
 user-model:hasTitle "Tugas Akhir Personalized Elearning";
 .
user-model:felder-silverman
 a user-model:LearningStyleTheory;
 user-model:hasTitle "Felder and Silverman Learning Style Theory";
 .
user-model:Personalized_Elearning_Group
 a user-model:ResearchGroup;
 rdfs:label "Personalized Elearning Research Group";
 .
user-model:Database_Group
 a user-model:StudyGroup;
 rdfs:label "Database Study Group";
 .
```

4.2.2 Pendefinisian *Rules*

Rules dibuat untuk melakukan proses *inference* terhadap ontologi. Proses *inference* ini sendiri dapat berguna untuk mendapatkan informasi baru dari data yang ada. Proses *inference* pada PortalCore menggunakan GenericRuleReasoner Jena sehingga struktur dan sintaks *rules* mengikuti *engine* tersebut. *Rules* pada prototipe ini disimpan pada *file testrules.rules.*

Ada beberapa RDFS *closure rules* yang sudah didefinisikan pada PortalCore, antara lain:

```
[rdfs7: (?a rdf:type rdfs:Class) -> (?a rdfs:subClassOf ?a)]
[rdfs8: (?a rdfs:subClassOf ?b), (?b rdfs:subClassOf ?c) -> (?a rdfs:subClassOf ?c)]
[rdfs9: (?x rdfs:subClassOf ?y), (?a rdf:type ?x) -> (?a rdf:type ?y)]
```

Rule rdfs7 menyatakan suatu *class* adalah *subclass* dari dirinya sendiri.
Rule rdfs8 menyatakan jika a *subclass* b, dan b *subclass* c, maka a juga subclass c.
Rule rdfs9 menyatakan jika x *subclass* y, a *instance* x, maka a juga *instances* y.

Berikut ini adalah *rules* yang didefinisikan oleh penulis:

- Untuk mendapatkan *member* suatu organisasi.
  Jika A adalah *member* (*isMemberOf*) dari B, maka B memiliki *member* (*hasMember*) A. (*inverse*)

```
(?A user-model:isMemberOf ?B) -> (?B user-model:hasMember ?A) .
```

- Untuk mendapatkan orang-orang yang memiliki satu *role* tertentu.
  Jika A memiliki *role* (*hasRole*) B, maka B adalah *role* (*isRoleOf*) dari A.

```
(?A user-model:hasRole ?B) -> (?B user-model:is_role_of ?A) .
```

- Untuk membantu visualisasi di PortalCore.
- *Rules* ini untuk menampilkan *property hasName* atau *hasTitle* sebagai *label* suatu *object*.

```
(?A user-model:hasName ?B) -> (?A rdfs:label ?B) .
```

4.3 Konfigurasi Portal

Proses konfigurasi *portal* dilakukan melalui satu *file* RDF (sources.n3) yang mendefinisikan *datasource, facets* dan *templates. File* RDF tersebut menggunakan *vocabulary* untuk PortalCore dengan *prefix* pcv: dan *namespace* http://jena.hpl.hp.com/2003/04/portal-config-vocab#.

4.3.1 Pendefinisian *Datasource*

Datasource menentukan *input* ontologi dan *instances data* yang digunakan, *rules*, *facets* dan *templates* yang digunakan, serta *stylesheets* dan beberapa *property* yang digunakan untuk mengatur konfigurasi PortalCore.

Pendefinisian *datasource* berupa pembuatan *instance* pcv:DataSource beserta pengaturan *property* yang dimiliki. Ada empat *datasource* yang didefinisikan yaitu Pengguna, Universitas, *Research Center*, dan *roles*. Berikut ini contoh definisi untuk *datasource* yang menampilkan Universitas.

```
#--------- Objek: Universitas ----------
[] rdf:type pcv:DataSource ;
 rdfs:label "Universitas" ;
 pcv:encoding "uni";
 pcv:order "3"^^xsd:integer ;
 dc:description "Prototipe User Model Ontology" ;

pcv:sourceURL <portal://data/datatestuser.n3> ;
pcv:sourceURL <portal://data/datatestother.n3> ;

pcv:ontologySourceURL <portal://data/user-model.owl> ;
pcv:closureRulesURL <portal://data/testrules.rules> ;

pcv:styleSheet "site.css" ;
```

Properties untuk mengatur *input portal* (ontologi, *instance, rule*):

- *pcv:ontologySourceURL.*

  Property ini untuk mengatur lokasi *files* ontologi yang digunakan.
- *pcv:sourceURL.*

  Property ini untuk mengatur lokasi *files instances data* yang digunakan.
- *pcv:closureRulesURL.*

  Property ini untuk mengatur lokasi *files rules* yang digunakan.

4.3.2 Pendefinisian *Facets*

Facet didefinisikan dalam *datasource* melalui *property* pcv:facet. Kemudian dibuat *instance facet* tersebut yang mendefinisikan *pcv:linkProp* untuk menampilkan objek yang dicari berdasarkan *property*. Berikut ini contoh definisi *facet* untuk Universitas.

```
pcv:filterOnType user-model:University;
. . . .
 pcv:facet user-model:universityFacet;
. . . .
user-model:universityFacet a pcv:Facet;
  rdfs:label "University";
              pcv:linkProp user-model:hasTitle;
              pcv:order "1"^^xsd:integer;
              rdfs:comment "The name of the University.";
  .
```

Facet Universitas ini akan menampilkan *instances* dari *class University*. *Property pcv:filterOnType* untuk membatasi *instance* dari *class* tertentu saja yang ditampilkan. *Property pcv:linkProp user-model:hasTitle*; akan menampilkan

instance dari *class* yang memiliki *property hasTitle*. Dengan kata lain *facet* ini akan menampilkan *instance* dari *University* berdasarkan *property hasTitle*. Pada Gambar 4.3 ditunjukkan tampilan *facet* Universitas.

Gambar 4.3 *Facet* Universitas

Oleh karena tujuan dari pembuatan prototipe ini adalah untuk mensimulasikan data pada ontologi *user-model* maka penulis tidak memfokuskan pekerjaan pada pengaturan tampilan. Pada PortalCore, *Instances* akan ditampilkan dalam bentuk tabel *raw data* jika tidak ada *template* untuk *instance* tersebut. Oleh karena fokus pengerjaan prototipe ini adalah menampilkan data maka penulis tidak membuat *template* tiap *class*.

4.4 Hasil Prototipe

Pengembangan prototipe ini bertujuan untuk menampilkan *instance data* dari *classes* yang ada pada ontologi *user-model*. Untuk melihat *instances data* tersebut penulis memanfaatkan fitur *faceted browse* dari PortalCore. *Faceted browse* berarti *browse* dengan melihat *object* yang ingin dicari dari berbagai dimensi. Kumpulan *object* yang ingin dilihat didefinisikan dalam *facet*. Penulis membuat 4 menu untuk melihat *object* melalui *facet*, yaitu Pengguna, Universitas, *Research Center*, dan *User Role*. Gambar 4.4 menunjukkan halaman *home* dari prototipe *user-model* yang menunjukkan 4 menu utama.

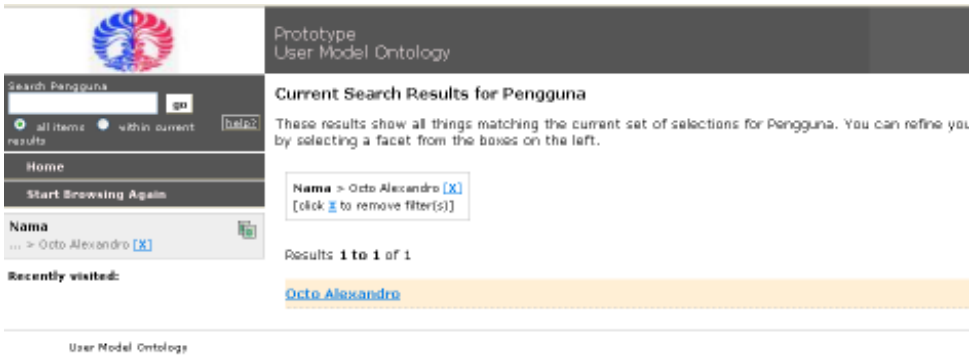
Gambar 4.4 Halaman Home Prototipe *user-model* dengan 4 menu utama *browse*

4.4.1 Pengguna

Pengguna adalah menu untuk melihat data seluruh pengguna sistem. Bila kita memilih menu ini akan ditampilkan *facet* yang menunjukkan nama-nama dari seluruh pengguna sistem. Gambar 4.5 menunjukkan halaman dengan *facet* pengguna. Yang ditampilkan oleh *facet* ini adalah *property hasName* dari setiap *instance User*.

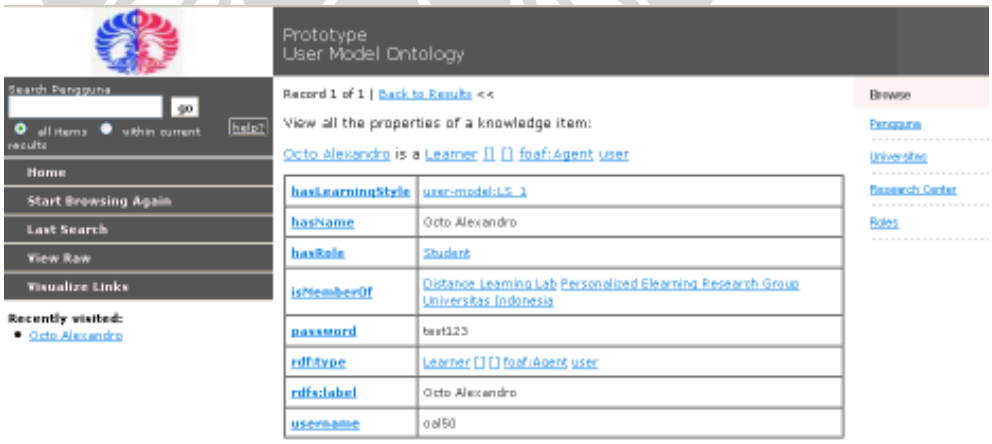
Gambar 4.5 Halaman Pengguna

Untuk melihat data seorang pengguna kita klik salah satu nama dan kemudian sistem akan otomatis melakukan *search* berdasarkan salah satu nama yang kita klik. Misalkan kita klik Octo Alexandro akan tampil hasil pencarian suatu *object User1* yang diberi *label* Octo Alexandro.Gambar 4.6 menunjukkan halaman *search result* tersebut.

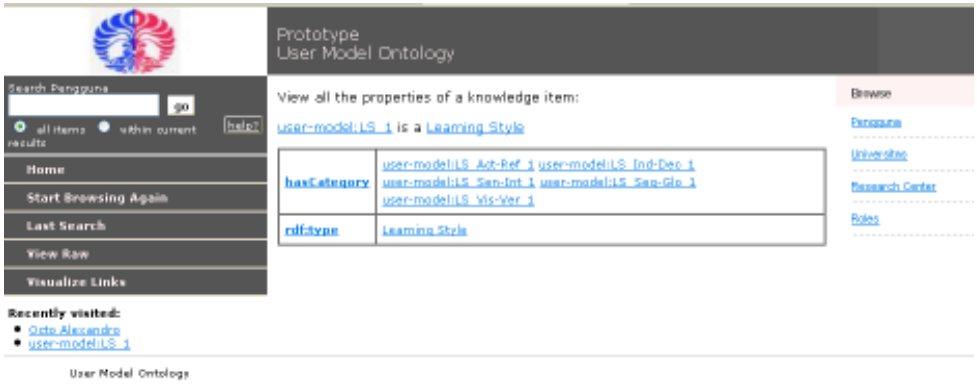


Gambar 4.6 *Search Result* Pengguna

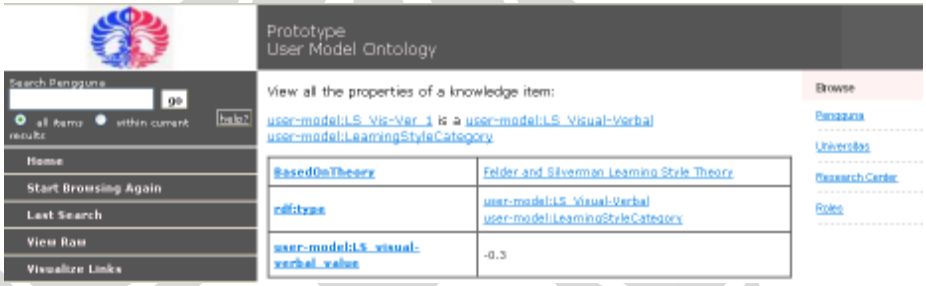
Kemudian kita klik *result* tersebut dan akan ditampilkan *instance data* dari pengguna tersebut. Gambar 4.7 menunjukkan *instance data* pengguna Octo Alexandro. Halaman itu menampilkan *Object* berlabel Octo Alexandro yang merupakan *instance* dari *Learner*.Dia memiliki nama (*hasName*) Octo Alexandro, *username* oal50, *role*-nya sebagai *Student*. Dia merupakan *member* (*isMemberOf*) dari Universitas Indonesia, *Distance Learning Lab*, dan *Personalized Learning Research Group*. Dia juga memiliki *learning style LS_1*. Untuk melihat detail mengenai *learning style* Octo Alexandro kita dapat mengklik *user-model:LS_1*. Gambar 4.8 menunjukkan *instance LS_1* sedangkan Gambar 4.9 menunjukkan detail nilai *learning style category visual-verbal*.



Gambar 4.7 Data Octo Alexandro



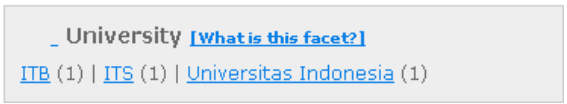
Gambar 4.8 Data *user-model:LS_1*



Gambar 4.9 Data *LS_Visual-Verbal* milik Octo Alexandro

4.4.2 Universitas

Universitas adalah menu untuk melihat *instances* dari *class University*. Pada suatu *object University* kita dapat melihat data tentang universitas dan pengguna yang termasuk anggota universitas tersebut. Langkah untuk melihat data universitas relatif sama dengan cara melihat pengguna. Gambar 4.10 menunjukkan *facet* universitas dan Gambar 4.11 menunjukkan *instance data* salah satu universitas, yaitu Universitas Indonesia.



Gambar 4.10 *Facet* Universitas

Record 1 of 1 | Back to Results <<

View all the properties of a knowledge item:

Universitas Indonesia is a University [] [] [] user-model:Organization

hasMember	Bambang Adhi Darwin Cuputra Leonny Meirna Octo Alexandro Siti Aminah Yohanes Immanuel
hasTitle	Universitas Indonesia
rdf:type	University [] [] [] user-model:Organization
rdfs:label	Universitas Indonesia

Gambar 4.11 Data Universitas Indonesia

4.4.3 *Research Center*

Research Center adalah menu untuk melihat *instances* dari *class ResearchCentre*. Pada suatu *object ResearchCentre* kita dapat melihat data tentang *research center* dan pengguna yang termasuk anggota r*esearch center* tersebut. Langkah untuk melihat data r*esearch center* relatif sama dengan cara melihat pengguna. Gambar 4.12 menunjukkan *facet Research Center* dan Gambar 4.13 menunjukkan *instance data* salah satu *research center*, yaitu Distance Learning Lab.

Research Center **[What is this facet?]**

Distance Learning Lab (1) | MIC Lab (1) | RISTEK BEM FASILKOM (1)

Gambar 4.12 *Facet Research Center*

Record 1 of 1 | Back to Results <<

View all the properties of a knowledge item:

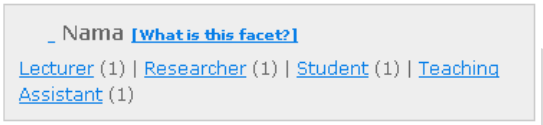
Distance Learning Lab is a Research Centre [] [] [] user-model:Organization

hasMember	Meirna Octo Alexandro
hasTitle	Distance Learning Lab
rdf:type	Research Centre [] [] [] user-model:Organization
rdfs:label	Distance Learning Lab

Gambar 4.13 *Distance Learning Lab*

4.4.4 *Roles*

Roles merupakan menu untuk melihat *instances* dari *class UserRole.* Pada suatu *object UserRole* kita dapat melihat orang-orang yang memiliki *role* tersebut. Langkah untuk melihat data *Roles* relatif sama dengan cara melihat pengguna. Gambar 4.14 menunjukkan *facet Roles* dan Gambar 4.15 menunjukkan *instance data* salah satu *role*, yaitu *Student*. Pada *instance Student* ditunjukkan pengguna-pengguna yang memiliki *role* sebagai *Student*.



Gambar 4.14 *Facet Roles*

Gambar 4.15 Data mengenai *role Student*