Jurnal Politeknik Caltex Riau
http://jurnal.pcr.ac.id

Rancang Bangun Sistem Informasi Pengembangan Sumber Daya Manusia menggunakan Teknologi Java pada Level Control (Studi Kasus PT. Chevron Pacific Indonesia)

Oggsa Sukardi[1], Juni Nurma Sari[2], dan Ardianto Wibowo[3]

[1]Politeknik Caltex Riau, E-mail : oggsa.os@gmail.com
[2]Politeknik Caltex Riau, E-mail : juni@pcr.ac.id
[3]Politeknik Caltex Riau, E-mail : ardie@pcr.ac.id

Abstrak

PT. Chevron Pacific Indonesia memiliki suatu program pelatihan untuk mengembangkan sumber daya manusia. Program pelatihan diadakan oleh pihak HR(Human Resource)dan ditujukan kepada pegawai yang baru bergabung atau memiliki pengalaman kerja kurang dari lima tahun di PT. Chevron Pacific Indonesia. Program ini membutuhkan sebuah sistem informasi berbasis web yang bisa diakses oleh peserta pelatihan . Sistem yang sudah ada masih menggunakan metode manual dalam mengolah data, sehingga membutuhkan waktu yang lama untuk memperoleh informasi dan menghasilkan laporan. Penelitian ini membahas bagaimana merancangan dan membangun sistem informasi yang dikembangkan menggunakan teknologi Java dengan menggunakan Spring Framework.Tujuan dari penelitian ini adalah merancang dan membangun sistem informasi berbasis web pada sisi controller yang dapat menghubungkan anatara sisi view dan sisi model. Sistem ini menerapkan konsep MVC, yaitu konsep yang memisahkan antara tiga lapisan diantaranya lapisan pemodelan data(Model), lapisan desain tampilan sistem(View), dan lapisan pengelola alur sistem(controller). Penelitian ini akan menitik beratkan pada pengembangan dan perancangan sistem dalam sisi controller yaitu membahas bagaimana alur sistem bekerja dan bagaimana peran controller dalam sistem ini. Hasil dari penelitian ini adalah sebuah sistem informasi Pengembangan Sumber Daya Manusia yang memepermudah pihak HR dalam memperoleh informasi atau laporan dan pembahasan bagaimana kinerja controller dalam menangani permintaan data dari level view.

Kata kunci : *PT.Chevron Pacific Indonesia, sistem informasi, web, MVC, controller*

Abstract

PT. Chevron Pacific Indonesia has a training program to develop human resources. The training program conducted by the HR (Human Resource) and addressed to new employees joining or have work experience of less than five years in PT. Chevron Pacific Indonesia. This program requires a web-based information system that can be accessed by a trainee. Existing systems still use manual methods in processing the data, so it takes a long time to obtain the information and generate reports. This study discusses how to design and build information systems that are developed using Java technology and uses the concept of MVC, the concept of separating the three layers including data modeling layer (Model), system interface design layer (View), and the management layer flow system (controller) . Penielitian will focus on the

development and design of systems in the controller is discussing how the system works and how the flow of the controller's role in this system. The results of this study is an information system Human Resources Development that facilitate the HR in obtaining information or a report and discussion of how the performance of the controller in handling requests for data from a view. The conclusion is the view controller can meet the needs of the display data or reports and the controller is running correctly in the connection between the view and the model.

***Keywords**: PT.Chevron Pacific Indonesia, information systems, Web, MVC, controller*

1. Pendahuluan

PT. Chevron Pacific Indonesia membutuhkan sebuah sistem informasi berbasis web yang dapat digunakan untuk mendapatkan informasi secara cepat dan akurat sehingga dapat memudahkan dalam mengambil keputusan. Dalam mengembangkan sistem informasi berbasis web ada salah satu konsep yang memiliki keunggulan dalam segi maintenance dan reusability yaitu konsep MVC (Model, View, dan Controller).

Oleh karena itu, dibuatlah sebuah sistem informasi berbasis web yang menerapkan konsep MVC, sehingga dapat memudahkan developer lain dalam mengembangkan sistem informasi ini. Sistem informasi ini dikembangkan menggunakan teknologi java dengan menggunakan Spring framework. Dalam penelitian ini akan membahas tentang penerapan konsep MVC dalam suatu sistem informasi berbasis web dengan menggunakan Spring framework. Sistem informasi ini menggunakan teknologi java dalam bahasa pemrograman. Penelitian ini hanya membahas konsep MVC dari sisi Controller yaitu bagian MVC yang bertugas mengatur alur berjalanya program dan pengolahan data.

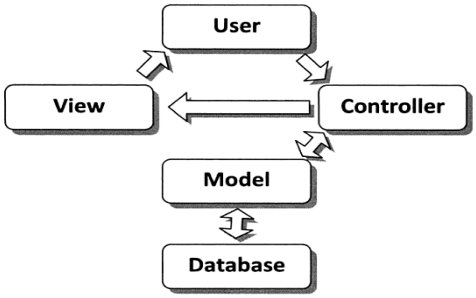
2. Tinjauan Pustaka

2.1 Teknologi Java

Java menurut definisi dari Sun Microsystem [1] adalah nama untuk sekumpulan teknologi yang membuat perangkat lunak dan menjalankan perangkat lunak pada komputer stand-alone ataupun pada lingkungan jaringan. Java adalah suatu teknologi di dunia software komputer, yang merupakan suatu bahasa pemrograman, dan sekaligus suatu platform (RP-Harahap, 2011)[2]. Teknologi Java dapat dikategorikan berdasarkan jenis perangkat lunak yang dikembangkan dan dipaketkan dalam tiga edisi yaitu, J2SE, J2EE,dan J2ME (Rr. Artiana Krestianti, 2010)[3].

2.2 Konsep MVC

MVC Model terdiri dari tiga komponen yaitu Model, View, dan Controller. MVC Model merupakan sebuah pola atau arsitektur dalam pengembangan perangkat lunak berbasis web yang memilah tiga komponen utama yang terpisah yaitu, komponen basis data (Model), komponen tampilan (View), dan komponen proses data (Controller) (Arief Hidayat,2012)[4]. Arief Hidayat[4] juga menambahkan bahwa dengan menggunakan konsep MVC, suatu aplikasi dapat dikembangkan sesuai dengan kemampuan pengembang preangkat lunak, yaitu programmer yang menangani bagian model dan controller, sedangkan designer yang menangani bagian view, sehingga penggunaan arsitektur MVC dapat meningkatkan maintanability dan organisasi kode



Gambar 1. Arsitektur MVC

2.3 Spring Framework

Menurut Toha[5] Spring merupakan open source framework yang diciptakan oleh Rod Johnshon dan pada akhir tahun 1996 Sun Microsystem menerbitkan spesifikasi Java Beans 1.00A. Spesifikasi ini menjelaskan tentang peraturan – peraturan koding java yang memperbolehkan suatu objek yang bisa menjadi komponen dan dapat dipakai berulang – ulang ke dalam suatu aplikasi java yang lebih kompleks. Toha (2010)[5] juga berpendapat bahwa Spring merupakan sebuah framework (kerangka kerja) yang digunakan untuk membangun sebuah aplikasi Enterprise. Spring termasuk framework yang ringan untuk mendukung secara penuh dalam pengembangan aplikasi Enterprise siap pakai.

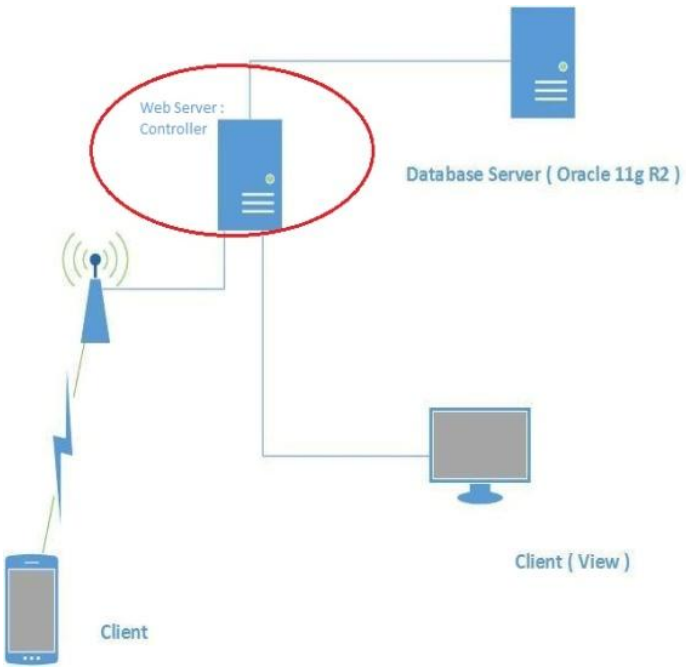
Toha[5] menambahkan bahwa didalam Spring Framework ada beberapa fitur-fitur yang baru yaitu :

1. Transaction Management: Spring framework menyediakan sebuah layer abstrak yang generik untuk manajemen transaksi, sehingga memudahkan para developer dalam melakukan manajemen transaksi.
2. JDBC Exception Handling: layer abstrak JDBC menawarkan exception yang bersifat hierarki sehingga memudahkan penanganan error.
3. Integration with Hibernate, JDO, and iBatis : Spring menawarkan layanan integrasi terbaik dengan Hibernate, JDO dan lainya.
4. AOP Framework: Spring merupakan framework AOP terbaik yang pernah ada.
5. MVC Framework: Spring hadir dengan framework aplikasi web MVC, yang dibangun di atas inti Spring. Spring merupakan framework yang sangat fleksibel dalam pengaturan strategi interface, dan mengakomodasi beberapa teknologi view seperti JSP, Velocity, Tiles, iText, dan POI.

3. Perancangan

3.1 Arsitektur Sistem

Berikut ini merupakan arsitektur sistem yang akan dikembangkan menggunakan teknologi Java dan menerapkan konsep MVC.

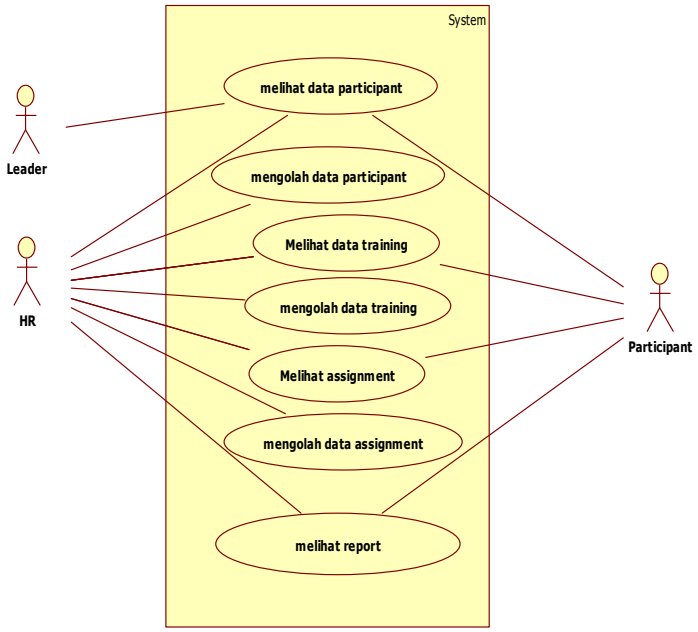


Gambar 2. Arsitektur Sistem

Pada penelitian ini akan membahas pada level controller, level ini bertugas mengelola data-data apa saja yang dibutuhkan oleh level View terhadap sistem. Level ini akan mengambil dan mengolah data sesuai dengan kebutuhan user yang diminta melalui View dan data tersebut dikirim ke View untuk ditampilkan sesuai dengan kebutuhan tersebut

3.2 Use Case Diagram

Pada perancangan usecase diagram aplikasi ini memiliki tiga aktor yaitu Partcipant, Leader, dan HR sebagai admin.



Gambar 3. Usecase Diagram Aplikasi

Pada usecase diagram diatas menunjukan bahwa setiap aktor dapat melihat data dan laporan terkait proses pelatihan pengembangan sumber daya manusia, namun tiap aktor dapat melihat data tersebut sesuai dengan hak akses.

4. Pembahasan dan Analisa

4.1 Pembahasan Sistem

Sistem informasi ini dikembangkan menerapkan konsep MVC dan menggunakan Spring framework. Didalam Spring framework pengembangan pada lapisan Controller menggunakan beberapa notasi atau code yang berbeda terhadap Class control biasa pada java.

4.1.1 Notasi @Controller

Untuk memulai sebuah class control dalam Spring framework harus dimulai dengan notasi @Controller pada awal class nya, ini digunakan agar memberikan tanda bahwa class tersebut merupakan class control. Notasi yang lainya juga digunakan sesuai dengan kebutuhan data yang diminta sesuai dengan proses bisnis.

```
package com.pcr.pdis.controller;

import java.util.List;

@Controller
public class TrainingController {
```

Gambar 4. Notasi @Controller

Dengan menggunakan notasi @Controller, class yang dibangun akan dianggap sebagai class controller.

4.1.2 Notasi @AutoWired

Notasi @Autowired merupakan notasi yang mendeklarasikan atau menginisialisasikan sebuah class service yang akan digunakan dalam mengambil data dari lapisan Model yang nantinya akan diteruskan ke lapisan View.

```
@Autowired
AreaService areaService;

@Autowired
EmployeeService employeeService;
```

Gambar 5. Notasi @AutoWired

Berdasarkan gambar 6 class Model AreaService dapat digunakan didalam class Controller, sehingga method-method yang dimiliki Class AreaService dapat digunakan didalam Class Controller.

4.1.3 Notasi @RequestMapping

Notasi ini merupakan notasi yang digunakan untuk menginisialisasikan sebuah request yang dipanggil dengan url, sehingga saat user mengakses sebuah request berbentuk url di browser maka, browser tersebut akan memanggil nama dari RequestMapping ini.

```
@RequestMapping("/getAreaList")
public ModelAndView getAreaList() {

    List<Area> AreaList = this.areaService.getAreaList();
    return new ModelAndView("list_Area", "AreaList", AreaList);

}
```

Gambar 7. Contoh @RequestMapping

Dari gambar diatas terlihat nama dari RequestMapping tersebut adalah /getAreaList, maka method yang dimilikinya akan menangani request dari url yang diakses user yang memanggil /getAreaList.

4.1.4 Notasi @RequestParam

Notasi @RequestParam merupakan notasi yang menginisialisasikan sebuah parameter yang dibutuhkan sebuah method yang nantinya akan ditangkap ketika user mengakses sebuah url yang ditangani method tersebut.

```
@RequestMapping("/deleteArea")
public String deleteArea(@RequestParam String badge_number,
        @RequestParam String start_date) {
    areaService.deleteData(badge_number, start_date);
    return "redirect:getAreaList";
}
```

Gambar 8. Notasi RequestParam

Berdasarkan gambar diatas sebuah RequestMapping yang bernama /deleteArea memiliki method yang membutuhkan dua parameter. Dua parameter itu akan dikirimkan oleh lapisan view disaat user mengkases /deleteArea.

4.1.5 Notasi @ResponseBody

Notasi ini merupakan notasi yang menangani format data Json. Didalam Spring Framework penanganan format data Json dapat dilakukan dengan mudah jika menggunakan notasi ini.

```
@RequestMapping(value = "/getAreaListjson", method = RequestMethod.GET)
public @ResponseBody
List<Area> area() {
    List<Area> are = this.areaService.getAreaList();
    return are;
}
```

Gambar 9. Penggunaan Notasi @ResponseBody

Dari gambar 9 menjelaskan bahwa sebuah method yang memiliki tipe data List yang menampung data yang dihasilkan dari pemanggilan method dari class Service. Karena method ini menggunkan notasi @ResponseBody maka List yang dikembalikan dari method ini akan berupa List dengan format data Json.

4.1.6 Notasi @RequestBody

Notasi ini merupakan notasi yang hampir sama dengan notasi @RequestParam dalam penggunaanya. Notasi ini merupakan notasi yang menunjukan adanya parameter yang dibutuhkan dalam sebuah method. Variabel yang diinisialisasikan oleh notasi ini akan menampung nilai parameter yang dikirmkan saat request dipanggil.

```
@RequestMapping(value="/insertAreajson", method= RequestMethod.POST)
@ResponseBody
public String empjson(@RequestBody Area are){
this.areaService.insertData(are);
return "Success";
        }
```

Gambar 10. Penggunakan Notasi @RequsetBody

Gambar 10 menjelaskan penggunaan notasi RequestBody dalam sebuah method yang membuthkan parameter. Notasi ini merupakan notasi yang menangani parameter format data Json.

4.2 Pembahasan Proses Bisnis

4.2.1 Proses View Data

Didalam view data peran controller adalah menerima request dari view dan meneruskan request tersebut ke model, dan mengambil data tersebut menggunakan service yang disediakan oleh model. Kemudian data tersebut diteruskan sesuai dengan format data yang dibutuhkan oleh view. Berikut salah satu contoh view data :

```
@RequestMapping("/getEmployeeList")
public ModelAndView getEmployeeList() {
    List<Employee> employeeList = this.employeeService.getEmployeesList();
            return new ModelAndView("list_Employee", "employeeList", employeeList);
}
```

Gambar 11. Contoh view data

Gambar 11 adalah contoh potongan program yang melakukan view data. Notasi requestmapping yang diberi nama /getEmployeeList akan dipanggil view untuk melihat data. Ketika view memanggil /getEmployeeList, controller akan mengeksekusi method yang ada dibagian requestmapping. Method yang ada pada gambar 11, melakukan pemanggilan data menggunakan class service yang disediakan model. Hasil pemanggilan data tersebut akan ditampung dalam sebuah list dan dikirm ke halaman list_employee, halaman list_employee akan menangkap data tersebut dan menampilkanya.

4.2.2 Proses Insert Data

Dalam menangani proses insert data, controller melakukan dua tahapan yaitu proses mengirim data dan menampung data untuk form insert data dan proses untuk melakukan insert ke database menggunkan fasilitas dari model.

```
@RequestMapping("/addData")
public ModelAndView addEmpData(@RequestParam String badge_number,
        ModelMap map) {
    List<Department> dept = this.departmentService.getDepartmentList();

    map.addAttribute("department", dept);
    map.addAttribute("location", new EmpLocation());
    map.addAttribute("area", new Area());
    map.addAttribute("opg", new OPG());
    map.addAttribute("badge", badge_number);

    List<String> Status = new ArrayList<String>();
    Status.add("Active");
    Status.add("Inactive");

    map.addAttribute("status", Status);

    return new ModelAndView("EmpData");
}
```

Gambar 12. Proses mengirim data

Gambar 12 merupakan potongan program yang melakukan pengiriman data untuk form insert dan menampung data yang dihasilkan form insert untuk dikirm ke proses insert data ke database.

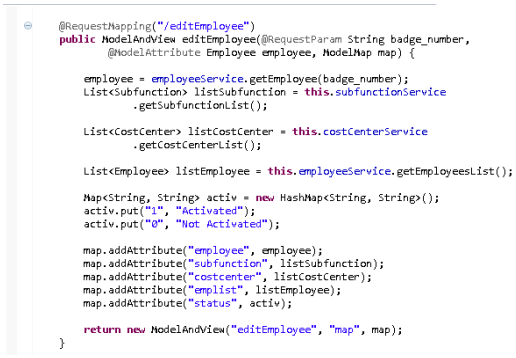
```
@RequestMapping(value = "/insertEmployee", method = RequestMethod.POST)
public String insertEmployee(
        @Valid @ModelAttribute("employee") Employee employee,
        BindingResult result, RedirectAttributes redirectAttributes) {

    if (result.hasErrors()) {
        System.out.println(result.hasErrors());
        redirectAttributes.addFlashAttribute(
                "org.springframework.validation.BindingResult.employee",
                result);
        redirectAttributes.addFlashAttribute("employee", employee);
        return "redirect:addEmployee";
    }

    employeeService.insertData(employee);
    return "redirect:getEmployeeList";

}
```

Gambar 13. Proses insert data

Gambar 13 merupakan proses melakukan insert data ke database menggunakan service yang disediakan oleh model. Data yang ditampung dari form insert akan diteruskan ke method ini, control akan melakukan validasi terlebih dahulu untuk memastikan data tersebut valid untuk dilakukan insert. Control menggunakan class EmployeeService agar method yang dimiliki dapat digunakan. Method yang digunakan untuk insert adalah insertData dengan parameter objek employee. Setelah dilakukan proses insert, control akan mengarahkan request ke getEmployeeList.

4.2.3 Proses Update Data

Proses update data hampir sama dengan proses insert. Porses ini memiliki dua proses yaitu mengirimkan data untuk form update dan proses melakukan update menggunakan class service yang disediakan oleh model.



```
@RequestMapping("/editEmployee")
public ModelAndView editEmployee(@RequestParam String badge_number,
        @ModelAttribute Employee employee, ModelMap map) {

    employee = employeeService.getEmployee(badge_number);
    List<Subfunction> listSubfunction = this.subfunctionService
            .getSubfunctionList();

    List<CostCenter> listCostCenter = this.costCenterService
            .getCostCenterList();

    List<Employee> listEmployee = this.employeeService.getEmployeesList();

    Map<String, String> activ = new HashMap<String, String>();
    activ.put("1", "Activated");
    activ.put("0", "Not Activated");

    map.addAttribute("employee", employee);
    map.addAttribute("subfunction", listSubfunction);
    map.addAttribute("costcenter", listCostCenter);
    map.addAttribute("emplist", listEmployee);
    map.addAttribute("status", activ);

    return new ModelAndView("editEmployee", "map", map);
}
```

Gambar 14. Proses mengirim data form

Gambar 14 merupakan proses mengirim data untuk form update dan menampung hasil inputan dari form update yang nantinya akan dikirim ke proses update data.

```
@RequestMapping("/updateEmployee")
public String updateEmployee(@ModelAttribute Employee employee) {
    employeeService.updateEmployee(employee);
    return "redirect:getEmployeeList";
}
```

Gambar 15. Proses insert data

Gambar 15 merupakan proses update data menggunakan parameter yang dikirimkan dari proses input sebelumnya. Controller akan menggunakan method updateEmployee yang disediakan oleh class employeeService dari model. Setelah proses update dilakukan controller akan mengarahkan request ke getEmployeelist yaitu menampilkan data employee.

4.2.4 Proses Delete Data

Dalam menangani proses delete data, controller menyediakan proses delete data yang membuthkan parameter, karena dalam melakukan delete data, controller menggunakan method yang disediakan class service dari model yang memiliki parameter sebagai acuan untuk melakukan delete data pada database.

```
@RequestMapping("/deleteEmployee")
public String deleteEmployee(@RequestParam String badge_number) {
    employeeService.deleteData(badge_number);

    return "redirect:getEmployeeList";

}
```

Gambar 16. Proses delete data

Ketika melakukan delete data, view mengirimkan badge_number ke controller, kemudian controller melakukan proses delete data menggunakan data yang dikirim dari view tersebut.

4.2.5 Proses Menampilkan Data Report

Dalam memenuhi kebutuhan reporting. Lapisan controller bertugas menyediakan data-data yang dibutuhkan view untuk menampilkan data berbentuk laporan.

```
@RequestMapping("/get_r_EnteringProgram")// Report Base on Entering Program per Function
public @ResponseBody
List<r_entering_program> getEntring(){
    List<r_entering_program> getf=this.r_byfunctionService.getEnteringProgram();
    return getf;
}

@RequestMapping("/get_r_GradbyFunc")// Report Base on Graduated per Function every Year
public @ResponseBody List<r_graduated_by_function> getgbf(){
    List<r_graduated_by_function> getgbf=this.r_byfunctionService.getGradByFunc();
    return getgbf;
}
```

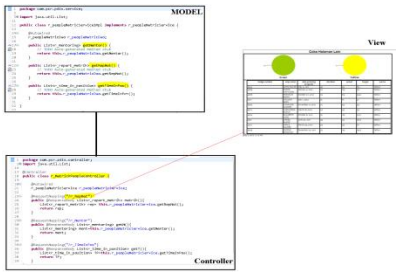
Gambar 17. Control Report

Gambar 17 merupakan proses menyediakan data untuk kebutuhan report pada view, method tersebut mengambil data dan menampung dalam bentuk list. Control ini mengirimkan data dalam bentuk format Json, karena untuk menampilkan report, view membutuhkan format data Json.

4.3 Pengujian Controller

4.3.1 Pengujian *Reusability*

Pengujian reusability dilakukan pada salah satu controller untuk membuktikan bahwa satu controller dapat digunakan lebih dari satu view dalam menampilkan data. Pada pengujian ini dilakukan pada class r_matricPeopleController. Class controller ini digunakan untuk menyediakan data kepada view untuk diolah dan ditampilkan menjadi People Matrics Report. Class controller ini menyediakan data format json sehingga menggunakan notasi @ResponseBody.



Gambar 18. Skema pengujian

Berdasarkan gambar 18 merupakan tampilan data laporan yang menggunakan data requestmapping /"r_getRepMat". Controller akan mengirimkan data berbentuk json ke birt-Reporting untuk merepresentasikan data dalam bentuk laporan. Terlihat bahwa view menggunakan controller yang sama dalam menampilkan report, sehingga dapat disimpulkan bahwa controller dapat digunakan oleh lebih dari satu view secara berulang kali.

4.3.2 Pengujuan menggunakan *Whitebox*

Dalam pengujian menggunakan metode whitebox, pengujian controller dilakukan pada dua method. Yaitu method Login dan Forecasting, pengujian ini menghitung nilai cyclomatic complexity yang merupakan perhitungan jumlah jalur indipenden dalam sebuah program.

$$V(G) = E - N + 2$$

Dimana:
E = jumlah edge pada grafik alir
N = jumlah node pada grafik alir

Gambar 19. Rumus perhitungan nilai CC

Berdasarkan gambar 19 terlihat bahwa untuk mendapatkan nilai E dan N, dibutuhkan perhitungan berdasarkan flowgraph sebuah method.

Berdasarkan pengujian menggunakan metode Whitebox dengan menghitung nilai cyclomatic complexity, didapatkan bahwa nilai cyclomatic complexity pada method login lebih besar yaitu V(G)=10 dibandingkan nilai cyclomatic complexity pada method forecasting yaitu V(G)=2. Sehingga dapat disimpulkan bahwa method login memiliki tingkat kompleksitas yang lebih tinggi dibandingkan method forecasting.

5. KESIMPULAN DAN SARAN

5.1 Kesimpulan

Berdasarkan hasil pengujian dan analisa sistem dapat ditarik kesimpulan sebagai berikut:

1. Data yang dikirimkan oleh controller dapat digunakan oleh view dalam menampilkan data dalam bentuk lain, sehingga ketika tampilan view dirubah dan menggunakan data yang sama, view hanya menggunakan control yang sama tanpa harus menambah control yang baru.
2. Berdasarkan analisa, controller dapat digunakan oleh beberapa view dalam menampilkan data. Sehingga tingkat reusability pada sistem ini dapat terpenuhi
3. Berdasarkan hasil pengujian Whitebox, method Forecasting meiliki tingkat komplesitas yang lebih kecil dibandingkan pada method Login

5.2 Saran

Saran yang didapat dalam penelitan ini untuk penelitian selanjutnya adalah menggunakan framework teknologi java lainnya seperti Struts atau Hibernate, sehingga dapat membandingkan teknologi *framework* yang lebih efisien dari segi pengolahan data, penulisan koding, dan lainya pada level controller.

Daftar Pustaka

[1] Hall, Mark: (2012). Java Introduction. Di ambil 26 Januari 2014 dari http://www.britannica.com/EBchecked/topic/573631/Sun-Microsystems-Inc/288357/Java

[2] Harahap, R. P. (2008). Teori Teknologi Java. Universitas Sumatra Utara.

[3] H.Arief, S. Bayu. (2012). "AMP C (Pengenalan Java) Bab 4". Laporan Tugas Akhir. Program Studi Ilmu Komputer Universitas Diponegoro.

[4] Jeni. (2013). Diambil 23 November 2013 dari todaypdf.org/jeni-web-programming-bab-7-mvc.html

[5] Toha, Mudzakkir. (2010). "Implementasi Framework Spring MVC untuk Pembuatan Sistem Informasi Managemen E Commerce". Laporan Tugas Akhir. Program Diploma III Ilmu Komputer Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Sebelas Maret.