# ATSE

**NAMA KELOMPOK :**

**NIA DINIYAH 1401129141**

**JOSEPHINE KARINA 1401127602**

**RAHMAN 1401127306**

**ALDOFA 1401127653**

**DAVID**

# Spring MVC

# Spring MVC

# Memakai Spring MVC di SpringSource Tool Suite

**MVC** adalah sebuah design pattern yang memisahkan logika aplikasi ke dalam tiga wilayah yang berbeda yaitu**Model**, **View**, dan **Controller**.  Hal ini berbeda dengan multi-tier (**N-tier**) dimana pada **N-tier** terdapat pemisahan tier pada mesin yang berbeda di komputer yang berbeda.

Salah satu alasan terbentuknya arsitektur N-tier adalah untuk menciptakan skalabilitas yang lebih baik.  Sebagai contoh, program database umumnya membutuhkan memori dalam jumlah besar sementara program logika membutuhkan prosesor yang kencang.  Dengan arsitektur N-tier, database berada di sebuah tier tersendiri yang diletakkan pada server dengan memori berjumlah besar dan prosesor sedang.

Sementara itu, business logic dapat berada di sebuah server dengan memori sedang tetapi memiliki prosesor berkecepatan tinggi.

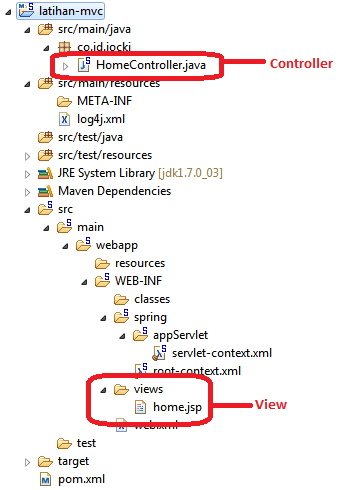
Bila **N-tier** tercipta karena alasan skalabilitas, maka **MVC** terlahir untuk memisahkan program kita kedalam tiga elemen yang terpisah dan sebisa mungkin tidak saling terkait.

Sebagai contoh, bagian **View** yang tadinya adalah HTML, dapat diganti menjadi PDF, Flash, ataupun WAP (mobile), tanpa harus memprogram ulang **Controller** dan**Model**.  Bila kita ingin meletakkan **MVC** ke dalam **N-tier**, maka **MVC** berada di **Presentation Tier**.

Pada artikel ini, saya akan membuat sebuah aplikasi **MVC** sederhana dengan memakai**Spring MVC** (bagian dari**Spring Framework**) dan IDE **SpringSource Tool Suite (STS)**.  Spring Framework memang lebih terkenal dengan**IoC** container-nya, yang banyak ditiru di JEE 6.  Memakai framework ini menimbulkan sebuah perasaan nostalgia tersendiri, karena Spring Framework adalah framework Java yang pertama kali saya pakai setelah lulus kuliah dan bekerja sebagai programmer junior.

Pada STS, saya mulai dengan memilih menu **File**, **New**, dan **Spring Template Project**.  Pada kotak dialog **New Template Project**, saya memilih **Spring MVC Project**.  Pada dialog **New Spring MVC Project** yang muncul, saya mengisi **Project Name** dengan **latihan-mvc**, mengisi **top-level package** dengan **co.id.jocki.latihan** dan kemudian men-klik tombol **Finish**.

STS akan membuat struktur proyek seperti pada gambar berikut ini:

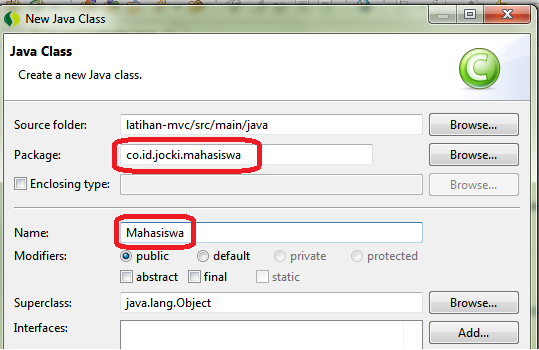
[](http://thesolidsnake.files.wordpress.com/2012/06/gambar1.png)

Struktur Proyek Spring MVC

Template **Spring MVC** secara otomatis membuat sebuah c**ontroller** bernama **HomeController** dan sebuah **view**dengan nama **home**.  Saya akan mengabaikan dua file ini.

## Membuat Model

Pada langkah berikutnya, saya membuat sebuah model dengan nama **Mahasiswa** di package**co.id.jocki.mahasiswa**.  Saya mulai dengan men-klik kanan pada **src/main/java** kemudian memilih **New**, **Class**.  Lalu saya mengisi dialog **New Java Class** dengan seperti yang terlihat di gambar:

[](http://thesolidsnake.files.wordpress.com/2012/06/gambar2.png)

Membuat Model Baru

Class Mahasiswa sebagai model memiliki isi seperti berikut ini:

package co.id.jocki.mahasiswa;

public class Mahasiswa {

private String nim;

private String nama;

private Date tanggalLahir;

public String getNim() {

return nim;

}

public void setNim(String nim) {

this.nim = nim;

}

public String getNama() {

return nama;

}

public void setNama(String nama) {

this.nama = nama;

}

public Date getTanggalLahir() {

return tanggalLahir;

}

public void setTanggalLahir(Date tanggalLahir) {

this.tanggalLahir = tanggalLahir;

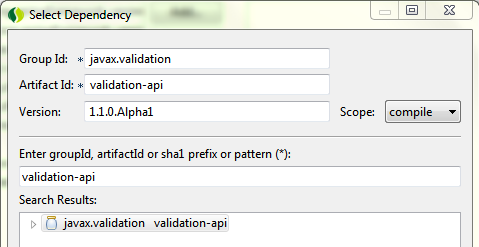
}

}

Sebelum beranjak meninggalkan **model** ini, saya ingin mencoba menggunakan fitur **Bean Validation** yang didukung oleh **Spring Framework**.  Dengan **Bean Validation**, saya tidak perlu report-repot melakukan validasi di sisi client maupun di sisi server, karena semuanya akan dilakukan otomatis oleh Spring Framework.

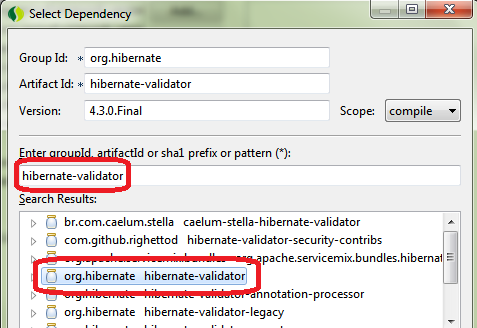
Untuk itu, saya perlu menambahkan dependency baru di **Maven** dengan melakukan langkah-langkah seperti berikut ini:

1. Buka file **pom.xml** di STS.  Akan muncul sebuah halaman editor untuk Maven.
2. Buka tab **Dependencies** (disebelah kanan tab **Overview**).
3. Klik tombol **Add…**
4. Pada bagian **Enter groupId, artifactId or sha1 prefix or pattern**, isi dengan **validation-api**.
5. Klik pada **javax.validation**seperti yang terlihat di gambar berikut ini:

[](http://thesolidsnake.files.wordpress.com/2012/06/gambar3.png)

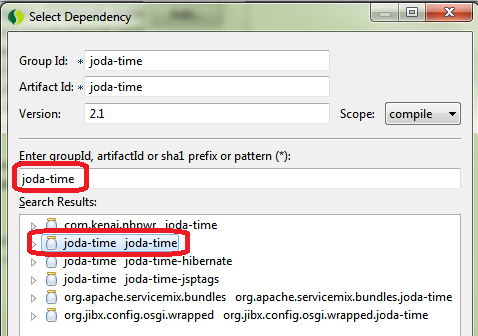
Dependency Maven untuk validation-api

1. Klik tombol **OK**.
2. Kembali Klik tombol **Add…**
3. Pada bagian **Enter groupId, artifactId or sha1 prefix or pattern**, isi dengan **hibernate-validator**.
4. Klik pada **org.hibernate**seperti yang terlihat di gambar berikut ini:

[](http://thesolidsnake.files.wordpress.com/2012/06/gambar3b.png)

Dependency Maven untuk hibernate-validator

1. Klik tombol **OK**.
2. Kembali klik tombol **Add…**
3. Pada bagian **Enter groupId, artifactId or sha1 prefix or pattern**, isi dengan **joda-time**.
4. Klik pada **joda-time**seperti yang terlihat di gambar berikut ini:

[](http://thesolidsnake.files.wordpress.com/2012/06/gambar3c.png)

Dependency Maven untuk joda-time

1. Klik tombol **OK**.
2. Klik kanan pada nama proyek, pilih **Maven**, **Update Dependencies…** dan klik pada tombol **OK**. Beri tanda centang pada **Force Update of Snapshots/Releases**.  Jangan lupa menghubungkan komputer ke internet agar Maven dapat men-download JAR yang dibutuhkan.  **Hibernate Validator** dan **Joda Time** adalah dua framework terpisah yang bukan merupakan bagian dari **Spring Framework**.

Sekarang, kembali ke class **Mahasiswa.java**, saya akan mengubah isi file tersebut menjadi:

package co.id.jocki.mahasiswa;

import java.util.Date;

import javax.validation.constraints.Past;

import javax.validation.constraints.Size;

import org.hibernate.validator.constraints.NotEmpty;

import org.springframework.format.annotation.DateTimeFormat;

public class Mahasiswa {

    @Size(min=8, max=8, message="NIM harus 8 digit!")

    private String nim;

    @NotEmpty(message="Nama harus di-isi!")

    private String nama;

    @Past(message="Tanggal harus berada di masa lalu!")

    @DateTimeFormat(pattern="dd/MM/YYYY")

    private Date tanggalLahir;

    public String getNim() {

        return nim;

    }

    public void setNim(String nim) {

        this.nim = nim;

    }

    public String getNama() {

        return nama;

    }

    public void setNama(String nama) {

        this.nama = nama;

    }

    public Date getTanggalLahir() {

        return tanggalLahir;

    }

    public void setTanggalLahir(Date tanggalLahir) {

        this.tanggalLahir = tanggalLahir;

    }

}

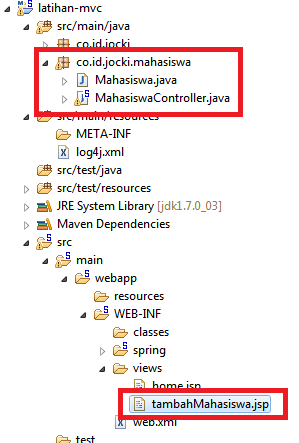
Pada kode program di atas, saya memberikan validasi **@Size** pada atribut **nim** dimana panjang **nim** harus 8 karakter.  Pesan yang ada di **message** akan ditampilkan di **view** bila nilai tidak memenuhi persyaratan.  Saya juga memberikan validasi **@NotEmpty** untuk atribute **nama**, sehingga nantinya **nama** wajib di-isi.

Saya memberikan validasi**@Past** di **tanggalLahir** untuk memastikan bahwa **tanggalLahir** yang dimasukkan oleh pengguna tidak melewati batas hari ini (karena tidak mungkin ada mahasiswa yang tanggal lahirnya tahun depan tetapi sudah duluan mendaftar).

## Membuat View

Sekarang, saya akan membuat **View**.  Pada **src/main/webapp/WEB-INF/views**, saya men-klik kanan folder **views**dan memilih **New**, **Others**.  Pada dialog yang muncul, saya memilih **Web**, **JSP File**, kemudian men-klik tombol**Next**.  Pada **File name**, saya mengisi dengan nama **tambahMahasiswa.jsp**.  Setelah itu, saya men-klik tombol**Finish**.

Struktur proyek akan terlihat seperti pada gambar berikut ini:

[](http://thesolidsnake.files.wordpress.com/2012/06/gambar4.png)

Menambah view baru

Berikut ini adalah isi file **tambahMahasiswa.jsp**:

<%@ page language="java" contentType="text/html; charset=ISO-8859-1"

    pageEncoding="ISO-8859-1"%>

<%@ taglib prefix="form" uri="http://www.springframework.org/tags/form" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">

<html>

<head>

<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">

<style type="text/css">

    .error {color: red; }

</style>

<title>Tambah Mahasiswa Baru</title>

</head>

<body>

    <h1>Tambah Mahasiswa Baru</h1>

    <form:form modelAttribute="mahasiswa" method="post">

        <fieldset>

        <p>

            <form:label path="nim" for="nim" cssErrorClass="error">NIM:</form:label>

            <form:input path="nim" />

            <form:errors path="nim" cssClass="error"/>

        </p>

        <p>

            <form:label path="nama" for="nama" cssErrorClass="error">Nama:</form:label>

            <form:input path="nama" />

            <form:errors path="nama" cssClass="error"/>

        </p>

        <p>

            <form:label path="tanggalLahir" for="tanggalLahir" cssErrorClass="error">Tanggal Lahir:</form:label>

            <form:input path="tanggalLahir"/>

            <form:errors path="tanggalLahir" cssClass="error"/>

        </p>

        <p>

            <input type="submit"/>

        </p>

        </fieldset>

    </form:form>

</body>

</html>

Pada halaman JSP tersebut, saya memakai tag library yang disediakan oleh Spring Framework dengan memakai directive **taglib**.  Spring MVC menyediakan tag seperti **<form:form>**, **<form:label>**, **<form:input>** dan **<form:errors>**untuk mempermudah data-binding dengan **model**.  Pada JSP tersebut, saya melakukan binding dengan model**Mahasiswa**.  Nilai **path** di **<form:label>**, **<form:input>** dan **<form:errors>** harus sesuai dengan namaatribut/variabel di **model**. Pada **<form:label>** saya menambahkan **cssErrorClass** sehingga bila terjadi kesalahan validasi, maka class tersebut akan dipakai (pada contoh ini, mengubah warna tulisan menjadi merah).

Kemudian, dengan mengikuti langkah yang sama dengan yang diatas, saya membuat sebuah **view** baru dengan nama **lihatMahasiswa.jsp**.  Isi dari file **lihatMahasiswa.jsp** adalah:

<%@ page language="java" contentType="text/html; charset=ISO-8859-1"

    pageEncoding="ISO-8859-1"%>

<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>

<%@ taglib prefix="fmt" uri="http://java.sun.com/jsp/jstl/fmt" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">

<html>

<head>

<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">

<style type="text/css">

    table {margin-bottom: 20px; border: solid 1px black;}

    th {text-align: left; padding-right: 20px; border-bottom: solid 2px black;}

    td {padding-right: 20px;}

</style>

<title>Lihat Daftar Mahasiswa</title>

</head>

<body>

<h1>Daftar Mahasiswa</h1>

<table>

<thead>

    <tr><th>NIM</th><th>Nama</th><th>Tanggal Lahir</th></tr>

</thead>

<c:forEach items="${lstMahasiswa}" var="mahasiswa">

    <tr>

    <td>${mahasiswa.nim}</td>

    <td>${mahasiswa.nama}</td>

    <td><fmt:formatDate value="${mahasiswa.tanggalLahir}" pattern="dd/MM/yyyy"/></td>

    </tr>

</c:forEach>

</table>

<a href="mahasiswa">Tambah Mahasiswa Baru</a>

</body>

</html>

Pada halaman JSP ini, saya mengharapkan **controller** untuk mengirimkan sebuah **List** dengan nama**lstMahasiswa**.  List tersebut harus merupakan kumpulan dari class **Mahasiswa** (model yang saya pakai).  Lalu dengan menggunakan JSTL **<c:forEach>**, saya melakukan perulangan untuk menampilkan setiap class**Mahasiswa** sebagai baris di dalam tabel.  Saya juga menggunakan JSTL **<fmt:formatDate>** untuk men-format atribut tanggalLahir yang bertipe **java.util.Date**.

## Membuat Controller

Berikutnya, saya membuat sebuah controller dengan nama **MahasiswaController.java** di package yang sama.  Caranya adalah dengan men-klik kanan di nama package **co.id.jocki.mahasiswa**, kemudian memilih menu **New**,**Class**.  Pada dialog **New Java Class** yang muncul, isi nama dengan **MahasiswaController**, kemudian klik tombol**Finish**.  Kemudian, saya membuat kode program seperti berikut ini untuk **MahasiswaController.java**:

package co.id.jocki.mahasiswa;

import java.util.ArrayList;

import java.util.List;

import javax.validation.Valid;

import org.springframework.stereotype.Controller;

import org.springframework.ui.Model;

import org.springframework.validation.BindingResult;

import org.springframework.web.bind.annotation.RequestMapping;

import org.springframework.web.bind.annotation.RequestMethod;

@Controller

@RequestMapping(value="/mahasiswa")

public class MahasiswaController {

    private List<Mahasiswa> lstMahasiswa = new ArrayList<Mahasiswa>();

    @RequestMapping(method=RequestMethod.GET)

    public String getFormTambah(Model model) {

        model.addAttribute(new Mahasiswa());

        return "tambahMahasiswa";

    }

    @RequestMapping(method=RequestMethod.POST)

    public String tambahMahasiswa(@Valid Mahasiswa mahasiswa, BindingResult result, Model model) {

        if (result.hasErrors()) {

            return "tambahMahasiswa";

        }

        lstMahasiswa.add(mahasiswa);

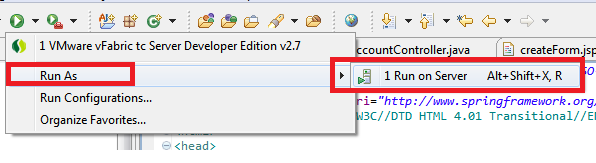
        model.addAttribute("lstMahasiswa", lstMahasiswa);

        return "lihatMahasiswa";

    }

}

Untuk menjalankan proyek ini, klik pada icon **Run As…** dan pilih **Run As**, **Run on Server**.  Secara default, STS akan memakai server **VMware vFabric tc Server Developer Edition v2.7** seperti yang terlihat digambar berikut ini:

[](http://thesolidsnake.files.wordpress.com/2012/06/gambar5.png)

Menjalankan Proyek

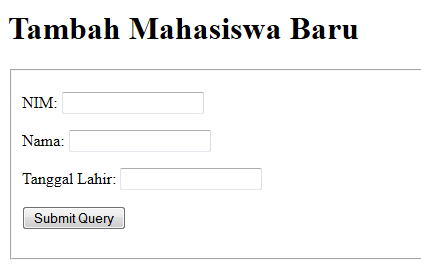
Pada dasarnya **tc Server** adalah sebuah server **Apache Tomcat** yang dilengkapi fitur tambahan (dan dukungan komersial).  Pada kotak dialog yang muncul, beri centang pada **Always use this server when running this project**.  Klik tombol **Next**.  Pastikan bahwa **latihan-mvc** berada di daftar **configured**, bukan di **available**.  Setelah itu klik tombol **Finish**.

STS akan berusaha menjalankan **tc Server** sembari menampilkan informasi di window **Console**.  Bila **tc Server**sudah selesai dijalankan, akan terdapat baris dengan tulisan seperti:

INFO: Server startup in 2789 ms

Saya akan mulai dengan membuka browser dan memasukkan URL [**http://localhost:8080/latihan-mvc/mahasiswa**](http://localhost:8080/latihan-mvc/mahasiswa).  Hal ini akan menyebabkan salah satu dari method di **MahasiswaController** dikerjakan, karena nilai value di **@RequestMapping** untuk **MahasiswaController** adalah **/mahasiswa**.

Karena saya memasukkan URL secara biasa dengan mengetik di browser, maka **request method** yang saya pakai adalah request method **GET**.  Dengan demikian, **getFormTambah()** di **MahasiswaController** akan dikerjakan.  **getFormTambah()** hanya membuat sebuah objek **Mahasiswa** baru, kemudian mengembalikan sebuah String **tambahMahasiswa**.  Hal ini akan menyebabkan view **tambahMahasiswa.jsp** ditampilkan di browser, seperti yang terlihat di gambar berikut ini:

[](http://thesolidsnake.files.wordpress.com/2012/06/gambar6.png)

View tambahMahasiswa

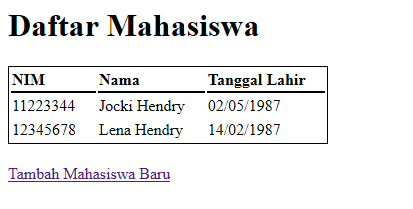
Pada saat saya menekan tombol **Submit Query**, maka URL **/mahasiswa** akan kembali dipanggil, tapi kali ini dengan request method **POST**.  Hal ini menyebabkan method **tambahMahasiswa()** di **MahasiswaController**dikerjakan.  Saya menambahkan annotation **@Valid** pada model **Mahasiswa** sehingga Spring MVC akan melakukan validasi model yang dikirim oleh view secara otomatis seperti yang terlihat pada gambar berikut ini:

[](http://thesolidsnake.files.wordpress.com/2012/06/gambar7.png)

Validasi secara otomatis

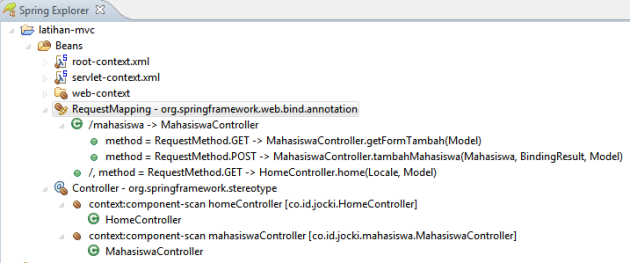
Bila tidak ada kesalahan, maka **model** yang dikirim oleh **view** akan disimpan ke dalam sebuah **List**.  Tentu saja, pada proyek nyata, **model** tidak disimpan ke dalam **list** melainkan kedalam sebuah database.

Setelah menyimpan model ke dalam **List**, **tambahMahasiswa()** akan memanggil view **lihatMahasiswa.jsp** sembari mengirimkan **List**berisi model **Mahasiswa** dengan nama atribut **lstMahasiswa**, sehingga tampilan yang diperoleh adalah seperti berikut ini:

[](http://thesolidsnake.files.wordpress.com/2012/06/gambar8.png)

Tampilan view lihatMahasiswa

Untuk melihat gambaran umum MVC di proyek secara garis besar, saya dapat membuka window **Spring Explorer**yang terlihat seperti pada gambar berikut ini:

[](http://thesolidsnake.files.wordpress.com/2012/06/gambar9.png)

Tampilan Spring Explorer

**Membuat MVC proyek Spring dengan SpringSource Tool Suite Home**

Ini adalah tutorial kedua bekerja dengan SpringSource Tool Suite, saat ini saya akan mengajarkan Anda bagaimana membuat aplikasi Spring MVC sederhana. Jika Anda seorang pemula dengan SpringSource Tool Suite, maka saya sarankan Anda membaca sebelum tutorial pertama tentang Memulai dengan SpringSource Tool Suite. Jika Anda seorang pemula dengan Spring Web MVC, maka pastikan untuk membaca pertama 16 ini. Web MVC framework.

Tim Pengembang SpringSource yang memberi kita pilihan yang kuat dalam SpringSource Tool Suite untuk membuat siklus pengembangan kita lebih mudah, opsi ini disebut Proyek Template Musim Semi di mana dalam opsi ini kami memiliki beberapa template yang ditetapkan di mana ini didasarkan pada Proyek musim semi tertentu.

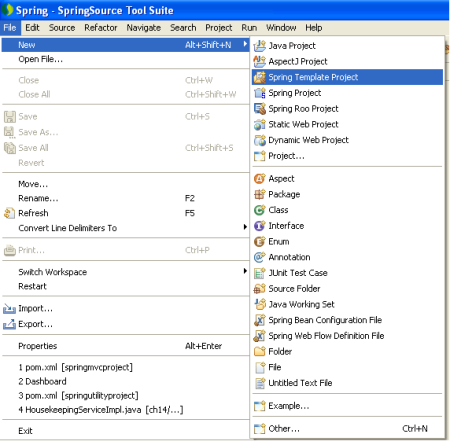
Tujuan dari setiap template sesuai tipenya sendiri adalah membuat sebuah proyek dengan beberapa pengaturan konfigurasi default dalam mendukung Anda untuk menghindari mengulang selalu konfigurasi proses pengaturan yang sama ketika Anda membuat jenis proyek yang sama itu setiap kali. Sebagai contoh, proyek ini akan dibuat termasuk semua wajib Maven dependensi, tentu saja, itu tergantung dari jenis proyek musim semi yang dipilih, selanjutnya, termasuk beberapa pengaturan konfigurasi dasar dan kode Java.

Bahkan lebih, misalnya, template berdasarkan MVC (dibahas nanti dalam tutorial ini), sudah dibuat dengan struktur direktori klasik untuk Aplikasi Java Web (WEB-INF dll).Catatan: Dalam banyak gambar berikut Anda akan menyadari bahwa banyak dari mereka telah ditangkap pada Windows. Hal ini karena bahwa Linux tidak bisa menangkap beberapa layar yang diinginkan dan futhermore saya termasuk beberapa petunjuk rinci untuk mendukung Anda.

**Startup SpringSource Tool Suite**

Langkah 01: Membuat Proyek Template Musim Semi

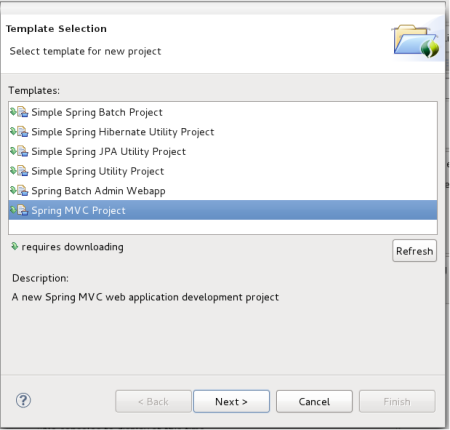
Setelah memulai IDE hanya melakukan tindak. File-> New-> Project Template Spring.



Gambar 01: Membuat Proyek Template Musim Semi

Langkah 02: Membuat MVC Proyek Musim Semi

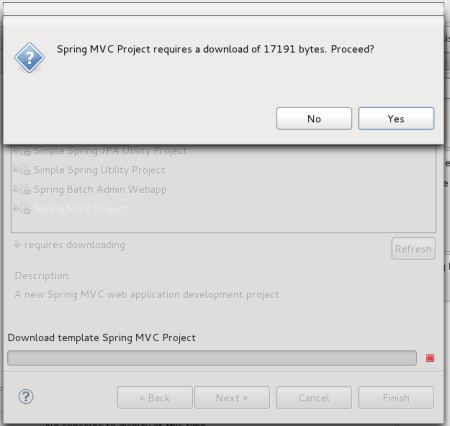
Setelah telah memilih opsi Template Proyek musim semi maka Anda sekarang harus memilih Template. Karena tutorial ini didasarkan pada MVC, karena itu pilih item musim semi Proyek MVC. Di bawah gambar yang menunjukkan semua sisa Template yang tersedia.



Gambar 02: Membuat MVC Proyek Musim Semi

Apakah menyadari diri sendiri bahwa setiap item memiliki di bagian kiri baris hijau. Dalam gambar yang sama Anda dapat melihat deskripsi dari baris hijau, yang berarti memerlukan download. Hanya saja terjadi hanya satu kali ketika Anda membuat sebuah proyek yang pertama kalinya berdasarkan template tertentu, dan itu adalah untuk mendapatkan template yang diinginkan.

Dari gambar sebelumnya kami melanjutkan untuk melakukan klik pada tombol Next>. Karena akan menjadi pertama kalinya kami bahwa kami membuat sebuah proyek yang berbasis di jenis template kita akan menerima pesan berikut. Lihat di bawah masing-masing gambar.

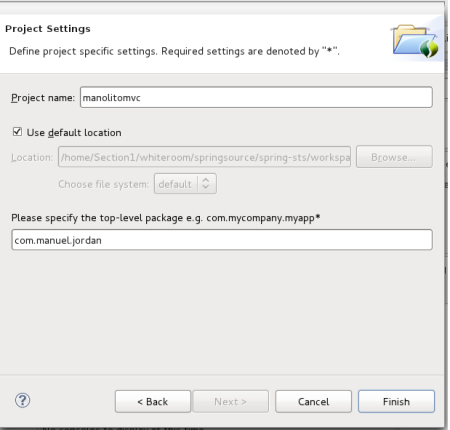


Gambar 03: Men-download Spring MVC Proyek Template

Lanjutkan untuk melakukan klik di tombol Yes. Proses download harus sangat cepat.

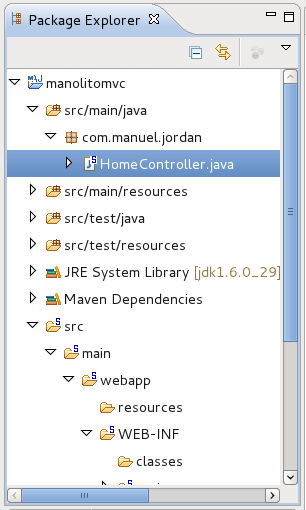
Langkah 03: Mendefinisikan Pengaturan Proyek

Langkah berikutnya adalah hanya menulis nama proyek dan menentukan paket top-level. Untuk masing-masing masing-masing saya akan menggunakan manolitomvc dan com.manuel.jordan. Di bawah gambar untuk pemahaman yang lebih baik.



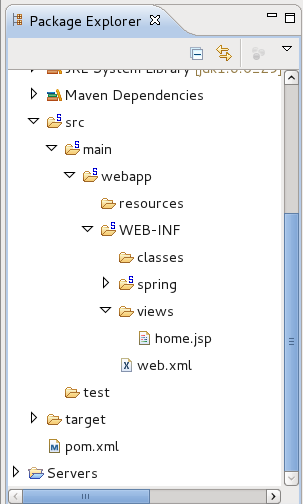
Gambar 04: Pengaturan Proyek untuk MVC Proyek Musim Semi

Setelah Anda telah menulis nilai-nilai, lanjutkan untuk melakukan klik pada tombol Finish. Dengan semua petunjuk ini, kami telah menciptakan Proyek MVC musim semi kami. Di bawah gambar menunjukkan bagian dari struktur keseluruhan proyek.



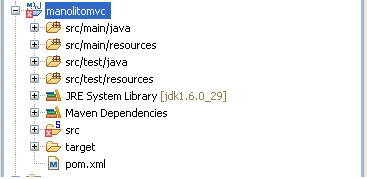
Gambar 05: Spring MVC Proyek dibuat

Apakah menyadari diri sendiri bahwa struktur direktori proyek didasarkan pada Apache Maven dan Anda dapat melihat bagian dari struktur direktori Web juga. Pada gambar berikutnya Anda dapat melihat lebih diperluas struktur ini sedikit



Gambar 06: MVC Proyek struktur direktori Web Musim Semi

Catatan:! Beberapa kali, tidak selalu, karena dependensi Inti Musim Semi dikelola oleh Maven, lebih detail segera, Anda bisa mendapatkan kesalahan dalam direktori proyek Anda. Lihat gambar berikutnya untuk pemahaman yang lebih baik.

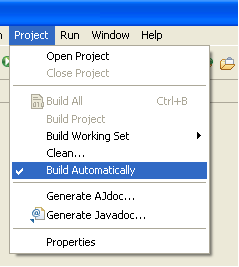


Gambar 07: Spring MVC Proyek dengan kesalahan

Jika Anda punya kesalahan ini, jangan khawatir, abaikan saja dengan saat ini, nanti Langkah 05, Anda akan belajar bagaimana untuk menghilangkan kesalahan ini.

Langkah 04: Menonaktifkan opsi Build Otomatis (Opsional)

Ini adalah pendapat saya sendiri, tapi saya pikir adalah bijaksana menonaktifkan opsi Build otomatis untuk IDE kami untuk menghindari beberapa masalah ketika Anda mengedit kode Anda. Untuk menonaktifkan fitur ini, hanya melakukan tindak Proyek-> Build secara otomatis. Di bawah gambar menunjukkan belum opsi ini diaktifkan. bila Anda telah selesai melakukan instruksi saya, baris hitam harus menghilang.



Gambar 08: Option Membangun otomatis diaktifkan

Langkah 05: Konfigurasi pom.xml

Ini merupakan langkah penting untuk pertimbangan kami. Buka file bernama pom.xml, harus terletak di bagian bawah struktur direktori, lihat Gambar 07. Sementara saya menulis tutorial ini rilis saat terbaru tetapi tidak stabil dari Spring Core 3.1.0.RC2.

Setelah dibuka file ini, praktis di atas Anda akan menemukan baris berikut:

<spring.framework.version>3.0.6.RELEASE</spring.framework.version>

Semi Inti 3.0.6.RELEASE adalah rilis stabil terbaru saat ini. Untuk tutorial ini saya akan bekerja dengan 3.1.0.RC2 Inti Spring. Oleh karena itu memperbarui lini ditunjukkan di atas dengan cara ini:

<spring.framework.version>3.1.0.RC2</spring.framework.version>

Karena pembaruan ini sekarang kita harus menyertakan repositori dalam ke file pom.xml kami untuk membiarkan proyek kami mendapatkan dependensi Inti Spring, kalau tidak kita akan mendapatkan masalah.

<repositories>

<repository>

<snapshots>

<enabled>false</enabled>

</snapshots>

<id>org.springframework.maven.milestone</id>

<name>Spring Maven Milestone Repository</name>

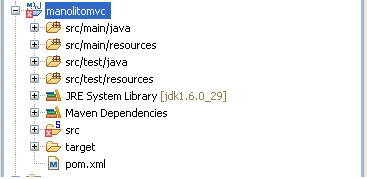
<url>http://maven.springframework.org/milestone</url>

</repository>

</repositories>

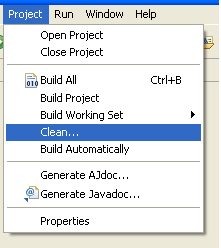
Sampai titik ini Anda dapat menyimpan file pom.xml. Dalam titik ini IDE harus mencoba untuk mendapatkan dependensi Inti Spring untuk 3.1.0.RC2 versi.

Catatan: Seperti saya katakan sebelumnya, mungkin Anda mendapatkan error dalam direktori proyek Anda. Saya pikir karena dependensi Inti Musim Semi dikelola oleh Maven. Lihat gambar berikutnya untuk pemahaman yang lebih baik.



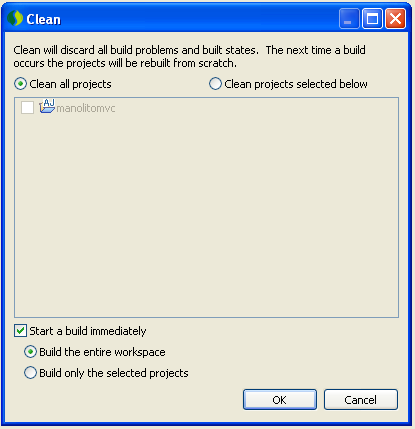
Gambar 09: Spring MVC Proyek dengan kesalahan

Untuk memperbaiki hal ini, hanya melakukan tindak. Pilih direktori proyek manolitomvc, lihat gambar sebelumnya, dan kemudian melakukan Proyek-> Clean ....



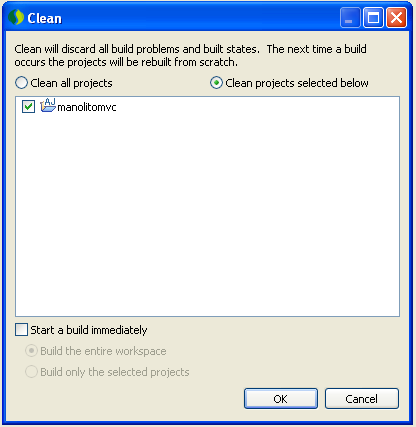
Gambar 10: Membersihkan Proyek (Part 01)

Setelah melakukan ini, Anda harus mendapatkan layar berikut.



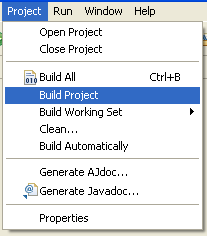
Gambar 11: Membersihkan Proyek (Part 02)

Karena IDE kami telah diinstal dan itu adalah proyek pertama kami, karena itu memiliki arti hanya melihat satu proyek. Cukup pilih proyek kami melalui kotak centang, untuk melakukan ini, sebelum Anda harus melakukan klik pada tombol radio bernama proyek Clean dipilih di bawah ini dan kemudian pastikan untuk hapus centang kotak centang bernama Mulai membangun segera. Untuk memahami petunjuk ini melihat gambar berikut.



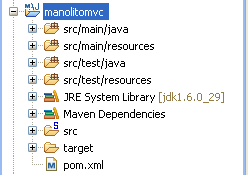
Gambar 12: Membersihkan Proyek (Part 03)

Setelah Anda mendapatkan layar sebelumnya di IDE Anda, melanjutkan untuk melakukan klik pada tombol OK. Kesalahan masih belum, maka pastikan Anda telah memilih direktori proyek kami dan kemudian melakukan tindak Proyek-> Build.



Gambar 13: Membangun Proyek

Setelah melakukan hal ini, kesalahan telah pergi. Berikut hasil akhir.



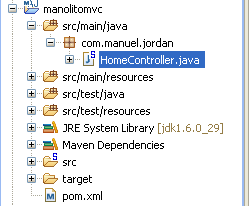
Gambar 14: Spring MVC Proyek tanpa kesalahan

Langkah 06: Mengedit Pengendali @ kami (Opsional)

Langkah ini adalah opsional, bebas untuk melewati itu.

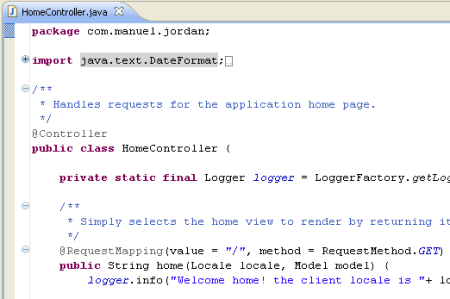
Catatan: Sekali lagi, jika Anda seorang pemula dengan Spring Web MVC, maka pastikan untuk membaca pertama 16 ini. Web MVC framework.

Pergi ke src / main / java dan memperluas com.manuel.jordan paket dan membuka file HomeController.java.



Gambar 15: Membuka file HomeController.java

Setelah membuka kelas ini Anda harus melihat kode sumber lengkap. Di bawah ini saya menunjukkan bagian dari kode sumber HomeController.



Gambar 16: kode sumber HomeController ini (Partial view)

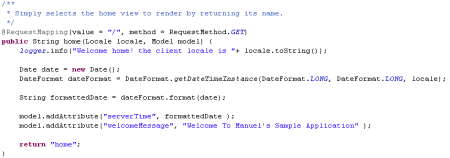
Di kelas ini, kita memiliki metode unik bernama rumah. Dalam metode ini kita memiliki baris kode ini:

model.addAttribute(“serverTime”, formattedDate );

Tepat di bawah garis seperti menulis ini yang baru.

model.addAttribute(“welcomeMessage”, “Welcome To Manuel’s Sample Application” );

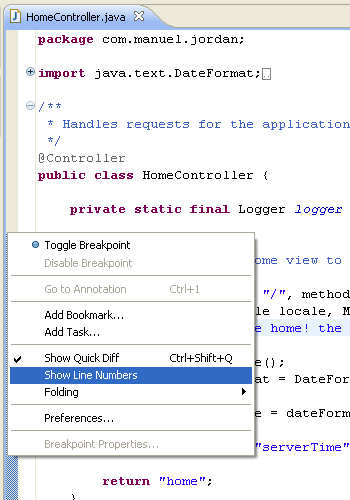
Hasil akhir harus seperti gambar berikut



Gambar 17: metode rumah HomeController 's diedit

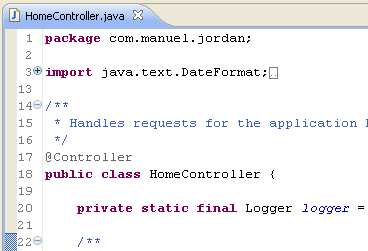
Langkah 07: Menampilkan Bilangan Baris (Opsional)

Jika Anda lakukan menyadari diri Anda sendiri, kita memiliki tidak nomor baris dalam kode sumber HomeController kami. Oleh karena itu untuk membuat pembangunan kita bekerja lebih mudah, mari kita mengaktifkannya. Hanya jangan klik kanan di bagian kiri jendela di mana muncul kode sumber HomeController dan pilih Tampilkan Numbers Line. Di bawah gambar untuk pemahaman yang lebih baik.



Gambar 18: Bilangan Baris Aktivasi dalam HomeController

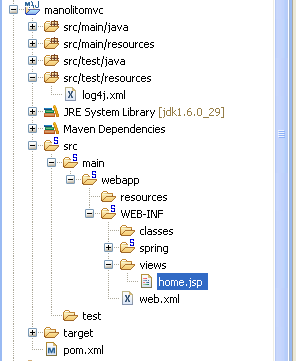
Setelah diaktifkan opsi ini Anda akan melihat nomor baris dalam kelas HomeController, bahkan lebih, Anda akan melihat fitur ini untuk semua file dalam IDE (praktis untuk semua proyek Anda), seperti. File java,. File jsp , file. xml,. properti file. Lihat gambar berikut di mana HomeController terlihat sekarang dengan nomor baris di bagian kiri.



Gambar 19: Bilangan Baris Termasuk dalam HomeController

Langkah 08: Edit file JSP kami (Opsional)

Setelah diedit kelas HomeController kami, sekarang kami harus mengedit file home.jsp. Untuk melakukan hal ini, pergi ke src / main / webapp / WEB-INF / pandangan dan membuka file home.jsp.



Gambar 20: Membuka file home.jsp

Setelah membuka file ini, Anda harus melihat kode sumber lengkap.



Gambar 21: kode sumber rumah itu

Dalam file JSP ini, kita memiliki baris kode ini:

<P> Waktu pada server adalah $ {} Servertime. </ P>

Tepat di bawah garis seperti menulis ini yang baru.

<P> $ {WelcomeMessage} </ P>

Hasil akhir harus seperti gambar berikut

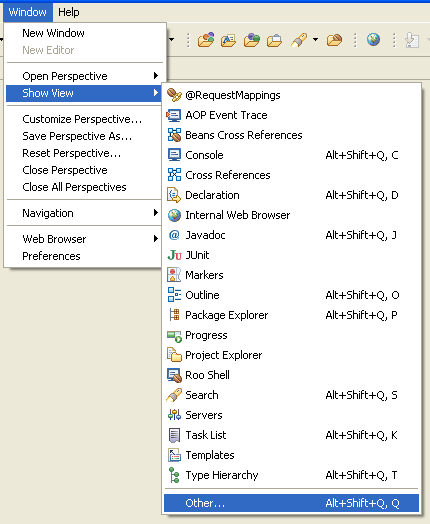


Gambar 22: kode sumber rumah itu diedit

Langkah 09: Membuka Server Lihat mengkonfigurasi VMware vFabric tc Server Developer Edition

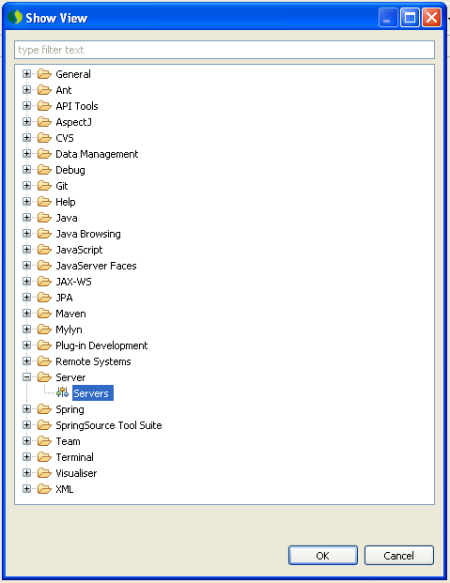
Setelah telah mengedit kode sumber kami, kami mampu melaksanakannya. Sebelum melakukan ini, pastikan untuk Membangun aplikasi kita untuk mencerminkan dua edisi ini. Pilih direktori proyek dan kemudian melakukan tindak Proyek-> Build. Lihat Gambar 13 terletak di Langkah 05 untuk pemahaman yang lebih baik.

SpringSource Tool Suite 2.8.1 telah disertakan secara default VMware vFabric tc Server Developer Edition v2.6. Mungkin Anda tidak dapat melihat server ini di IDE Anda, karena pandangan Server belum terlihat. Jika Anda memiliki situasi ini, hanya melakukan tindak Jendela-> Show View-> Other ...



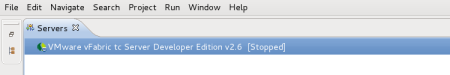
Gambar 23: Membuka Server View (Part 01)

Lalu pergi ke direktori Server dan pilih item yang unik bernama Server dan akhirnya melakukan klik pada tombol OK.



Gambar 24: Membuka Server View (Part 02)

Setelah melakukan ini, Anda harus melihat VMware vFabric tc Server Developer Edition v2.6 dalam tampilan Server seperti gambar berikut.

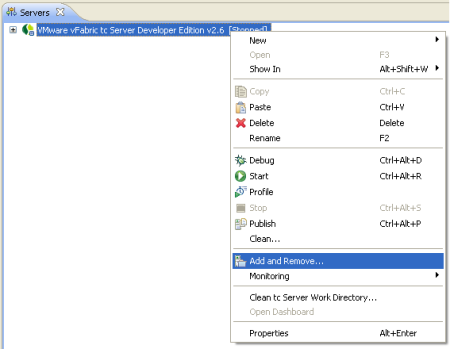


Gambar 25: Server View

Jika Anda melihat kembali Gambar 23, perhatikan bahwa opsi Server sudah muncul, baik beberapa kali tidak muncul, itu sebabnya saya mengajar Anda dengan cara ini.

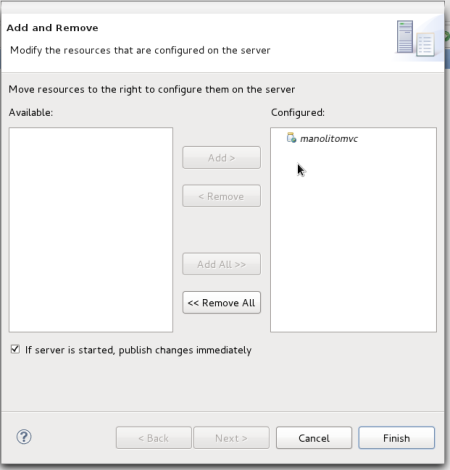
Langkah 10: Konfigurasi VMware vFabric tc Server Developer Edition

Setelah kami dapat melihat server melihat kita dapat mengkonfigurasi VMware vFabric tc Server kami Developer Edition v2.6 untuk bekerja dengan Spring MVC aplikasi kami. Untuk melakukan hal ini, di VMware vFabric tc Server Developer Edition v2.6 melanjutkan untuk melakukan klik kanan dan pilih item yang bernama Add dan Remove ....



Gambar 26: Memilih Add dan Remove ... option

Setelah melakukan ini, mungkin Anda akan melihat manolitomvc proyek di bagian kiri bernama Tersedia. Jika ini terjadi hanya memilih proyek dan melakukan klik di tombol Add>. Sekarang manolitomvc proyek kami harus berada di bagian kanan bernama dikonfigurasi. Untuk saat ini terus memeriksa kotak centang bernama Jika server dimulai, mempublikasikan perubahan segera. Lihat gambar berikut untuk memahami konfigurasi akhir. Setelah melakukan ini, melanjutkan untuk melakukan klik pada tombol Finish

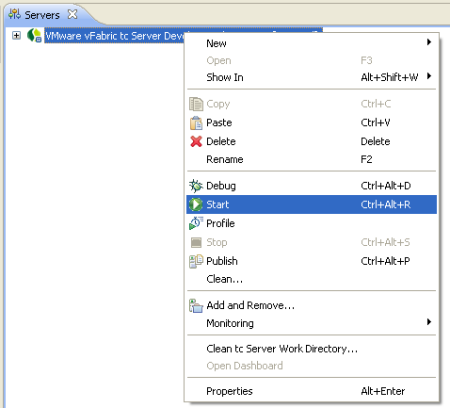


Gambar 27: Memindahkan proyek manolitomvc ke bagian Configured

Langkah 11: Menjalankan VMware vFabric tc Server Developer Edition

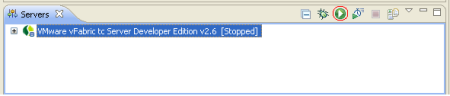
Dengan konfigurasi sebelumnya, proyek manolitomvc kami mampu dipertimbangkan untuk dikembangkan oleh VMware vFabric tc Server Developer Edition v2.6.

Oleh karena itu kita sekarang dapat startup server. Untuk melakukan hal ini, sekali lagi dalam VMware vFabric tc Server Developer Edition v2.6 melanjutkan untuk melakukan klik kanan dan pilih item yang bernama Start.



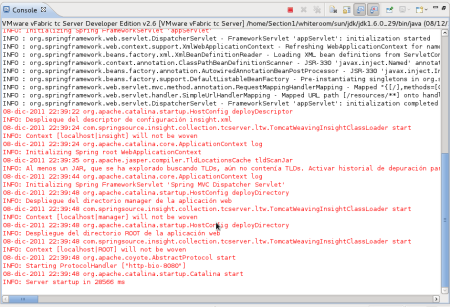
Gambar 28: Mulai VMware vFabric tc Server Developer Edition v2.6 (Jalan 01)

Cara lain untuk startup server adalah melakukan klik pada ikon lingkaran hijau yang terletak di bagian kanan, lihat gambar berikut lingkaran merah yang berisi ikon ini.



Gambar 29: Mulai VMware vFabric tc Server Developer Edition v2.6 (Jalan 02)

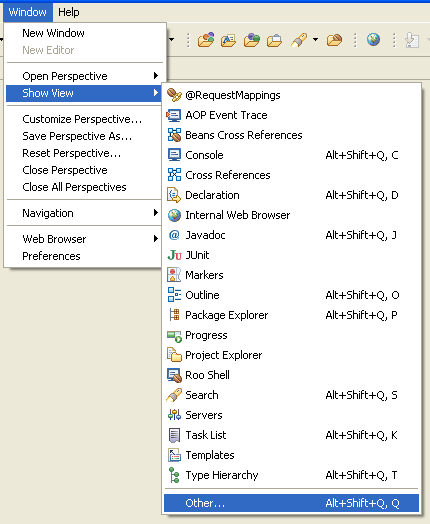
Setelah diberi perintah untuk startup server di salah satu dari dua bentuk yang tersedia, Anda akan melihat output tindak.



Gambar 30: VMware vFabric tc Server Developer Edition v2.6 Dimulai

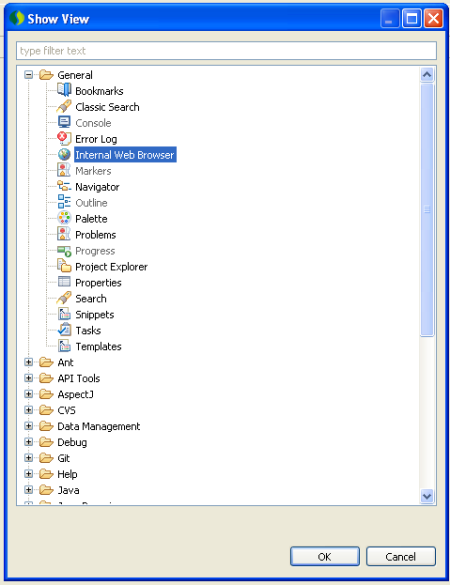
Langkah 12: Membuka Web Browser internal

Setelah mulai server, IDE kita secara otomatis harus membuka tampilan Web Browser internal. Jika tidak terjadi, lakukan tindak Jendela-> Show View-> Lainnya ....



Gambar 31: Membuka internal Web Browser View (Part 01)

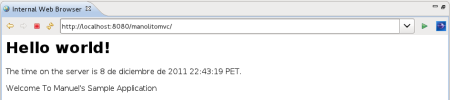
Lalu pergi ke direktori Umum dan pilih item yang unik bernama Web Browser internal dan akhirnya melakukan klik pada tombol OK.



Gambar 32: Membuka internal Web Browser View (Part 02)

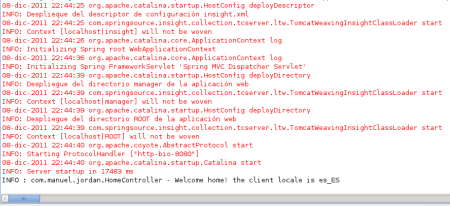
Langkah 13: Menjalankan kami Spring MVC aplikasi

Setelah Anda punya pandangan Web Browser internal, menulis URL http://localhost:8080/manolitomvc/ tindak dan tekan enter atau melakukan klik pada segitiga hijau yang terletak di bagian kanan. Bila Anda telah dilakukan baik dua pilihan untuk mengeksekusi URL. Anda harus mendapatkan output tindak.



Gambar 33: Menjalankan aplikasi manolitomvc

Perhatikan View Konsol output, kita memiliki pesan INFO log. Hal ini dalam warna hitam.



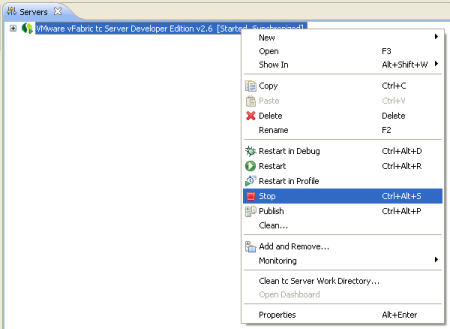
Gambar 34: Menjalankan aplikasi manolitomvc

Setiap kali Anda mengeksekusi URL, pesan INFO log baru akan muncul.

Langkah 14: Shutdown VMware vFabric tc Server Developer Edition

Jika Anda ingat, di Langkah 11, kami dapat startup server kami dalam dua cara, dengan cara yang sama, kita bisa shutdown server kami dalam dua cara.

Cara pertama adalah tindak, lagi di VMware vFabric tc Server Developer Edition v2.6 melanjutkan untuk melakukan klik kanan dan pilih item yang bernama Berhenti.



Gambar 35: Shutdown VMware vFabric tc Server Developer Edition v2.6 (Jalan 01)

Cara kedua untuk melakukan klik pada ikon merah persegi yang terletak di bagian kanan.

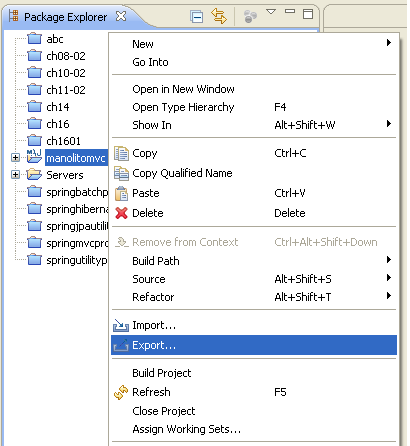
http://manueljordan.files.wordpress.com/2011/12/stsvfabrictcservershutdown02.png?w=450

Gambar 36: Shutdown VMware vFabric tc Server Developer Edition v2.6 (Jalan 02)

Setelah berhenti server, dalam tampilan Console Anda harus melihat beberapa pesan log yang menunjukkan bahwa server shutdown berhasil (saya tidak posting gambar tentang hal ini).

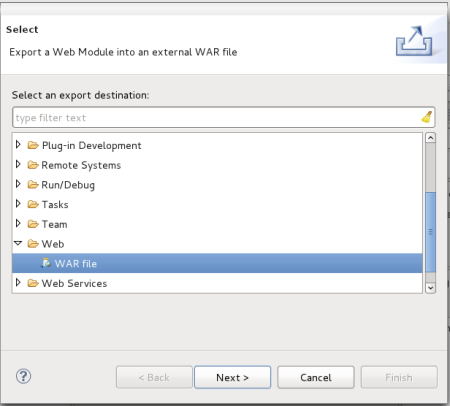
Langkah 15: Mengekspor File manolitomvc.war

Langkah terakhir dari tutorial ini adalah mengekspor file perang. Dari proyek kami. Saya akan menggunakan ini perang tutorial lainnya.. Untuk mengekspor proyek kami lakukan klik kanan di direktori proyek dan pilihan item Ekspor ....



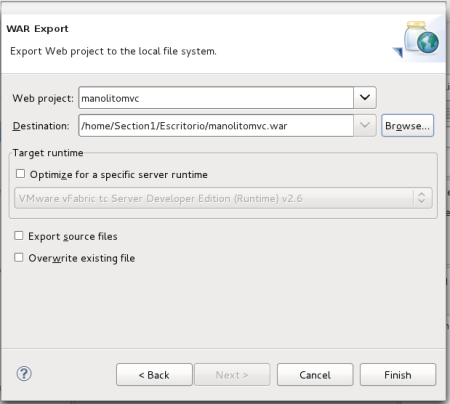
Gambar 37: Mengekspor File manolitomvc.war (Part 01)

Setelah melakukan ini, pergi ke direktori web dan pilih file WAR barang dan melakukan klik pada tombol Next>.



Gambar 38: Mengekspor File manolitomvc.war (Part 02)

Sekarang melakukan klik pada tombol Browse ... dan pilih tujuan, saya akan menggunakan Desktop saya. Biarkan sisa opsi tanpa perubahan. Di bawah gambar dengan hasil akhir.



Gambar 39: Mengekspor File manolitomvc.war (Part 03)

Setelah melakukan ini, hanya klik di tombol Finish

Catatan Akhir: Selamat!. Anda telah belajar bagaimana membuat aplikasi MVC Spring, mengkonfigurasi IDE kami, mengedit proyek kode sumber, startup dan shutdown VMware vFabric tc Server Developer Edition v2.6 dan akhirnya bagaimana untuk mengekspor file perang..