# 1. tiles

참고사이트

:http://discafe.tistory.com/entry/egov-%ED%83%80%EC%9D%BC%EC%A6%88tiles-%EC%A0%81%EC%9A%A9-3

## 가. pom.xml 에 라이브러리 추가

```
<properties>
	...
	<org.apache.tiles-version>3.0.3</org.apache.tiles-version>
</properties>
```

- 스프링 프레임워크 인 경우

```
<!-- jsp tiles -->
	<dependency>
		<groupId>org.apache.tiles</groupId>
		<artifactId>tiles-core</artifactId>
		<version>${org.apache.tiles-version}</version>
	</dependency>
	<dependency>
		<groupId>org.apache.tiles</groupId>
		<artifactId>tiles-servlet</artifactId>
		<version>${org.apache.tiles-version}</version>
		</dependency>
	<dependency>
		<groupId>org.apache.tiles</groupId>
		<artifactId>tiles-extras</artifactId>
		<version>${org.apache.tiles-version}</version>
	</dependency>
```

- egov framework인 경우

타일즈 3버전 이상은 스프링 3.2 이상부터 지원한다고 한다.

스프링 3.2 이하일 경우 타일즈 2 버전으로 해야 한다.

tiles-core, tiles-servlet, tiles-extras 의 라이브러리를 추가한다.

여기서 exclusions는 해당 라이브러리를 현재 dependency에는 포함하지 않는다는 뜻이다.

이 exclusions를 안하면 egov 프레임워크와 타일즈의 다른 slf4j 버전으로 충돌이 발생 한다.

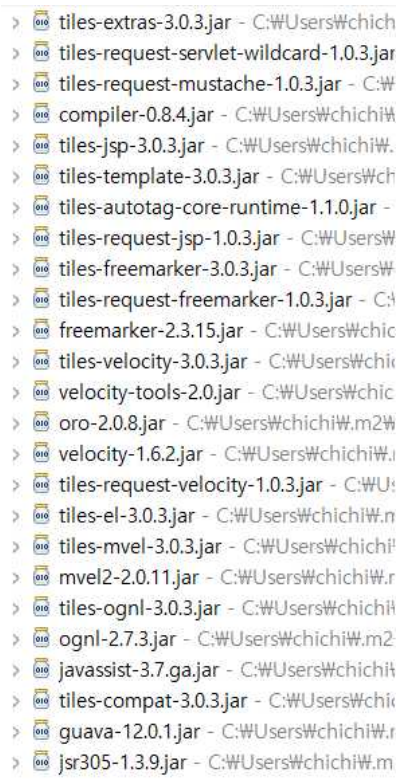
```
<!-- jsp tiles -->
	<dependency>
		<groupId>org.apache.tiles</groupId>
		<artifactId>tiles-core</artifactId>
		<version>${org.apache.tiles-version}</version>

		<!-- needed to exclude slf4j which causes incompatibilities -->
		<exclusions>
			<exclusion>
				<groupId>org.slf4j</groupId>
				<artifactId>slf4j-nop</artifactId>
			</exclusion>
			<exclusion>
				<groupId>org.slf4j</groupId>
				<artifactId>slf4j-api</artifactId>
			</exclusion>
			<exclusion>
				<groupId>org.slf4j</groupId>
				<artifactId>jcl-over-slf4j</artifactId>
			</exclusion>
		</exclusions>
	</dependency>
	<dependency>
		<groupId>org.apache.tiles</groupId>
		<artifactId>tiles-servlet</artifactId>
```

```
                <version>${org.apache.tiles-version}</version>

                <!-- needed to exclude slf4j which causes incompatibilities -->
                <exclusions>
                    <exclusion>
                        <groupId>org.slf4j</groupId>
                        <artifactId>slf4j-nop</artifactId>
                    </exclusion>
                    <exclusion>
                        <groupId>org.slf4j</groupId>
                        <artifactId>slf4j-api</artifactId>
                    </exclusion>
                    <exclusion>
                        <groupId>org.slf4j</groupId>
                        <artifactId>jcl-over-slf4j</artifactId>
                    </exclusion>
                </exclusions>
            </dependency>
            <dependency>
                <groupId>org.apache.tiles</groupId>
                <artifactId>tiles-extras</artifactId>
                <version>${org.apache.tiles-version}</version>

                <!-- needed to exclude slf4j which causes incompatibilities -->
                <exclusions>
                    <exclusion>
                        <groupId>org.slf4j</groupId>
                        <artifactId>slf4j-nop</artifactId>
                    </exclusion>
                    <exclusion>
                        <groupId>org.slf4j</groupId>
                        <artifactId>slf4j-api</artifactId>
                    </exclusion>
                    <exclusion>
                        <groupId>org.slf4j</groupId>
                        <artifactId>jcl-over-slf4j</artifactId>
                    </exclusion>
                </exclusions>
            </dependency>
```



## 나. dispatcher-servlet.xml

하지만 타일즈를 적용할라면 다른 View Resolver를 사용해야 한다.

**UrlBasedViewResolver를** 주석 처리한 후 아래와 같이 타일즈를 사용 할 수 있도록 변경하자.

```
<bean class="org.springframework.web.servlet.view.UrlBasedViewResolver" p:order="1"
        p:viewClass="org.springframework.web.servlet.view.tiles3.TilesView" />
        <!--Don't add suffix or prefix like you do with .jsp files-->

<bean id="tilesConfigurer" class="org.springframework.web.servlet.view.tiles3.TilesConfigurer" >
        <property name="definitions">
            <value>/WEB-INF/tiles/tiles.xml</value>
        </property>
</bean>
```

## 다. tiles.xml

경로 : 컨텍스트₩src₩main₩webapp₩WEB-INF₩tiles₩tiles.xml

템플릿인 layout.jsp파일을 지정해준다.
그리고 layout 파일 안에 header, content, footer 파일을 주입시켜준다.

또한 URL 패턴이 /로 구분하여 2개, 3개일 경우(egov가 2개, 3개의 URL패턴을 가진다.) 패턴에 맞는/요청 URL에 맞는 jsp파일을 리턴하도록 할 수 있다.

또한 extends를 통해 레이아웃을 상속 받을 수 있다.
주석 처리된 것은 어떤 URL패턴이 요청하면 web/jsp에 구분자에 맞게 contents만 변경하여 페이지를 리턴한다.
이는 header와 footer에 중복되는 코드는 그대로 사용하고 컨텐츠만 변경하여 가지고 리턴할 수 있다.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE tiles-definitions PUBLIC "-//Apache Software Foundation//DTD Tiles Configuration
2.1//EN"
        "http://tiles.apache.org/dtds/tiles-config_2_1.dtd">

<tiles-definitions>
        <definition name="mainTemplate" template="/WEB-INF/jsp/tiles/layout.jsp">
                <put-attribute name="header" value="/WEB-INF/jsp/tiles/header.jsp" />
                <put-attribute name="content" value="/WEB-INF/jsp/tiles/content.jsp" />
                <put-attribute name="footer" value="/WEB-INF/jsp/tiles/footer.jsp" />
        </definition>
        <definition name="*/*" extends="mainTemplate">
                <put-attribute name="content" value="/WEB-INF/jsp/{1}/{2}.jsp" />
        </definition>
        <definition name="*/*/*" extends="mainTemplate">
                <put-attribute name="content" value="/WEB-INF/jsp/{1}/{2}/{3}.jsp" />
        </definition>
</tiles-definitions>
```

## 라. layout.jsp

참조: http://www.w3schools.com/html/html_layout.asp

```
<style>
div.container {
        width: 100%;
        border: 1px solid gray;
}

header, footer {
        padding: 1em;
        color: white;
        background-color: black;
        clear: left;
        text-align: center;
}
```

```
nav {
        float: left;
        max-width: 160px;
        margin: 0;
        padding: 1em;
}

nav ul {
        list-style-type: none;
        padding: 0;
}

nav ul a {
        text-decoration: none;
}

article {
        margin-left: 170px;
        border-left: 1px solid gray;
        padding: 1em;
        overflow: hidden;
}
</style>
```

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<%@ taglib prefix="tiles" uri="http://tiles.apache.org/tags-tiles"%>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>
<!DOCTYPE html>
<html>
<head></head>
<body>
        <div class="container">
                <header>
                        <h1>City Gallery</h1>
                </header>

                <nav>
                        <tiles:insertAttribute name="header" />
                </nav>

                <article>
                        <tiles:insertAttribute name="content" />
                </article>

                <footer>
                        <tiles:insertAttribute name="footer" />
                </footer>
        </div>
</body>
</html>
```

마. content.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
내용
```

바. footer.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
Copyright
```

사. header.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
  <ul>
    <li><a href="#">London</a></li>
    <li><a href="#">Paris</a></li>
    <li><a href="#">Tokyo</a></li>
  </ul>
```

# 2. ViewResolver 설정

## 가. viewResolver 구현 클래스

| ViewResolver 구현 클래스 | 설 명 |
|---|---|
| InternalResourceViewResolver | 뷰이름으로부터 JSP나 Tiles 연동을 위한 View 객체를 리턴 |
| VelocityViewResolver | 뷰 이름으로부터 Velocity 연동을 위한 View 객체를 리턴 |
| VelocityLayoutResolver | VelocityViewResolver와 동일한 기능을 제공하며, 추가로 Velocity의 레이아웃 기능을 제고 |
| BeanNameViewResolver | 뷰 이름과 동일한 이름을 갖는 빈 책체를 View 객체로 사용 |
| ResourceBundleViewResolver | 뷰 이름과 View 객체간의 매핑 정보를 저장하기 위해 자원 파일을 사용 |
| XmlViewResolver | 뷰 이름과 View 객체간의 매핑 정보를 저장하기 위해 XML 파일을 사용 |

## 나. InternalResourceViewResolver

(1) Jsp나 Html 파일과 같이 웹 어플리케이션의 내부 자원을 이용하여 뷰를 생성하는 AbstractUrlBasedView 타입의 뷰 객체를 리턴

(2) 기본적으로 사용하는 View 클래스임

(3) prefix, suffix 프로퍼티를 사용

## 다. BeanNameViewResolver

(1) 뷰 이름과 동일한 이름을 갖는 빈을 뷰 객체로 사용

(2) 주로 커스텀 View 클래스를 뷰로 사용해야 하는 경우에 사용

```
<bean id="viewResolver" class="org.springframework.web.servlet.view.BeanNameViewResolver"/>
<bean id="download" class="sp.mvc.file.download.ExcelDown"/>
```

특정 Controller에서 뷰의 이름을 "download"라고 지정 시 위의 sp.mvc.file.download.ExcelDown 클래스가 처리

## 라. XmlViewResolver

(1) 뷰 이름과 동일한 이름을 갖는 빈을 뷰 객체로 사용

(2) 별도의 XML 설정 파일로부터 빈 객체를 검색

(3) Location 프로퍼티의 값을 지정하지 않을 경우 기본값은 "/WEB-INF/views.xml"임

```
<bean id="viewResolver" class="org.springframework.web.servlet.view.XmlViewResolver">
  <property name="location" value="/WEB-INF/xml_views.xml"></property>
</bean>
```

## 마. 다수의 ViewResolver 설정

(1) 하나의 DispatcherServlet은 한 개 이상의 ViewResolver를 설정할 수 있음

(2) "order" 프로퍼티를 이용하여 뷰 이름을 검사할 ViewResolver의 순서를 결정

(3) "order" 프로퍼티의 값이 작을수록 우선 순위가 높으며 우선순위가 높은 ViewResolver가 null을 리턴하면, 다음 우선순위를 갖는 ViewResolver에 뷰을 요청

(4) 주의할 점은 InternalResourceViewResolver는 마지막 우선 순위를 갖도록 지정해야 함 <= InternalResourceViewResolver는 항상 뷰 이름에 매핑되는 뷰 객체를 리턴하기 때문에 null을 리턴하지 않음. 따라서 InternalResourceViewResolver의 우선순위가 높을 경우 우선순위가 낮은 ViewResolver는 사용되지 않게 됨.

```
<bean id="beanNameViewResolver" class="org.springframework.web.servlet.view.BeanNameViewResolver" p:order="1"/>
<bean id="internalResourceViewResolver" class="org.springframework.web.servlet.view.InternalResourceViewResolver">
<property name="order" value="2"></property>
```