目标

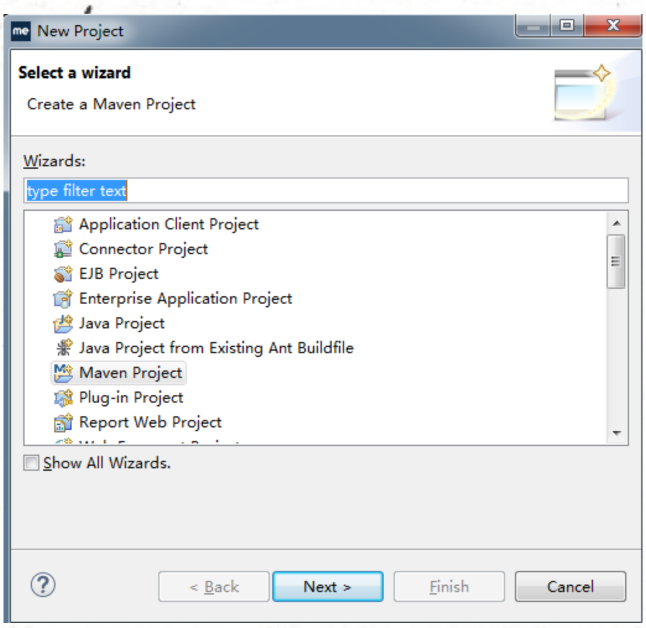
使用框架，完成注册一个用户，和登陆

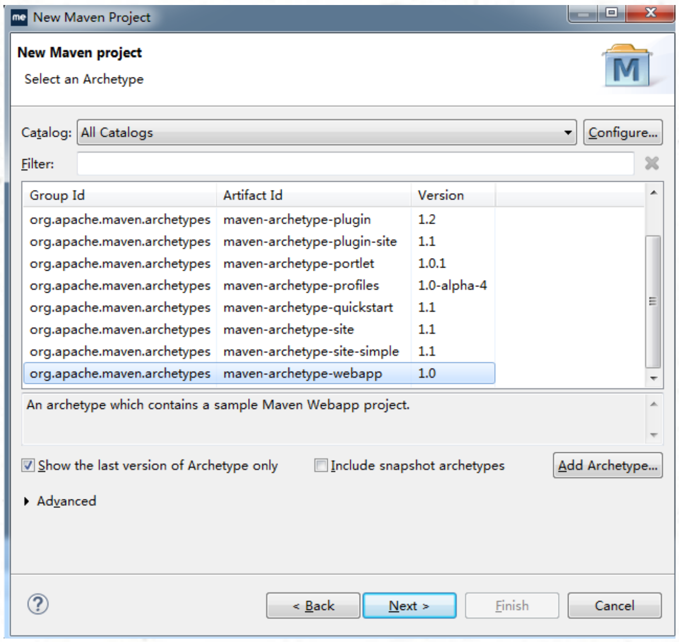
准备工作

创建一个用户表

```
CREATE TABLE `t_user` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `username` varchar(20) DEFAULT NULL,
  `nickname` varchar(20) DEFAULT NULL,
  `password` varchar(20) DEFAULT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

创建项目




New Maven Project

New Maven project

Select an Archetype

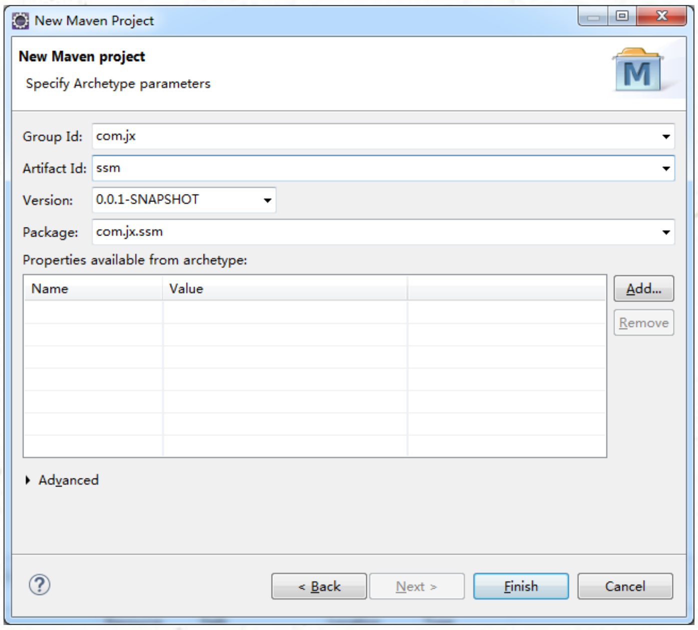
Catalog: All Catalogs

Filter:

Group Id	Artifact Id	Version
org.apache.maven.archetypes	maven-archetype-plugin	1.2
org.apache.maven.archetypes	maven-archetype-plugin-site	1.1
org.apache.maven.archetypes	maven-archetype-portlet	1.0.1
org.apache.maven.archetypes	maven-archetype-profiles	1.0-alpha-4
org.apache.maven.archetypes	maven-archetype-quickstart	1.1
org.apache.maven.archetypes	maven-archetype-site	1.1
org.apache.maven.archetypes	maven-archetype-site-simple	1.1
org.apache.maven.archetypes	maven-archetype-webapp	1.0

An archetype which contains a sample Maven Webapp project.

[✓] Show the last version of Archetype only [ ] Include snapshot archetypes


New Maven Project

New Maven project

Specify Archetype parameters

Group Id: com.jx

Artifact Id: ssm

Version: 0.0.1-SNAPSHOT

Package: com.jx.ssm

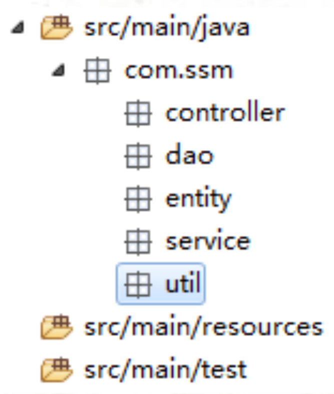
Properties available from archetype:

Name	Value

创建目录结构

1 创建 java和test目录。eclipse下不好创建，那么就到操作系统中src/main文件夹中创建两个文件夹。

2 创建package包：在eclipse创建paceage包



src/main/java
com.ssm
controller
dao
entity
service
util
src/main/resources
src/main/test

配置pom.xml文件

1 修改junit版本为4.12

2 加入plugins配置 引入编译插件 （这个时候项目可能报错,那么选中项目 右键，maven -> update一下就好了.）

```
<plugins>
    <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-compiler-plugin</artifactId>
        <version>2.3.2</version>
        <configuration>
            <source>1.8</source>
            <target>1.8</target>
        </configuration>
    </plugin>
</plugins>
```

测试maven web项目是否已经建立好了

新建一个 index.jsp 发布到tomcat如果正常启动，并且可以访问，那么maven web 项目就建立好了，准备工作就做完了

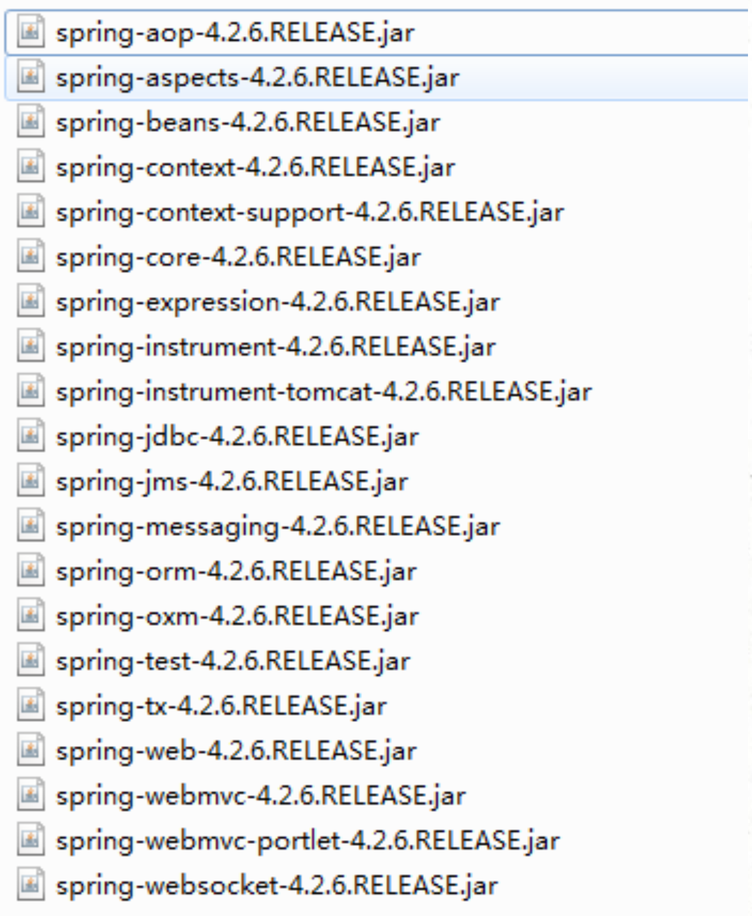
引入spring+springmvc

修改pom文件引入jar

加入properties节点,这里加入全局的来控制spring版本和项目构建编码

```
  <properties>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
    <spring-version>4.3.6.RELEASE</spring-version>
  </properties>
```

下面图示是spring下载下来的所有jar包



```
spring-aop-4.2.6.RELEASE.jar
spring-aspects-4.2.6.RELEASE.jar
spring-beans-4.2.6.RELEASE.jar
spring-context-4.2.6.RELEASE.jar
spring-context-support-4.2.6.RELEASE.jar
spring-core-4.2.6.RELEASE.jar
spring-expression-4.2.6.RELEASE.jar
spring-instrument-4.2.6.RELEASE.jar
spring-instrument-tomcat-4.2.6.RELEASE.jar
spring-jdbc-4.2.6.RELEASE.jar
spring-jms-4.2.6.RELEASE.jar
spring-messaging-4.2.6.RELEASE.jar
spring-orm-4.2.6.RELEASE.jar
spring-oxm-4.2.6.RELEASE.jar
spring-test-4.2.6.RELEASE.jar
spring-tx-4.2.6.RELEASE.jar
spring-web-4.2.6.RELEASE.jar
spring-webmvc-4.2.6.RELEASE.jar
spring-webmvc-portlet-4.2.6.RELEASE.jar
spring-websocket-4.2.6.RELEASE.jar
```

http://repo.spring.io/release/org/springframework/spring/ 这里是spring的下载地址

这里不引入这么多，先引入最基本的jar包

配置web.xml

web3.0之后开始支持不需要web.xml，直接在Java类中通过继承AbstractAnnotationConfigDispatcherServletInitializer 覆盖几个方

法来进行配置web项目这里不做说明

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xmlns="http://java.sun.com/xml/ns/javaee"
         xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd"
        version="3.0">


  <display-name>ssm</display-name>


  <!-- 配置spring -->
  <listener>
      <listener-
class>org.springframework.web.context.ContextLoaderListener</listener-
class>
  </listener>
  <context-param>
      <param-name>contextConfigLocation</param-name>
      <!-- 指定spring配置文件 -->
      <param-value>classpath:applicationContext.xml</param-value>
  </context-param>

  <!-- 配置spring mvc -->
  <servlet>
      <servlet-name>springmvc</servlet-name>
      <servlet-class>org.springframework.web.servlet.DispatcherServlet
      </servlet-class>
      <init-param>
          <param-name>contextConfigLocation</param-name>
          <!-- 指定spring mvc配置文件 -->
          <param-value>classpath:spring-mvc.xml</param-value>
      </init-param>
      <load-on-startup>1</load-on-startup>
   </servlet>
   <servlet-mapping>
        <servlet-name>springmvc</servlet-name>
        <url-pattern>/</url-pattern>
   </servlet-mapping>



   <!-- 设置编码filter -->
   <filter>
       <filter-name>CharEncoding</filter-name>
       <filter-
class>org.springframework.web.filter.CharacterEncodingFilter</filter-class>
       <async-supported>true</async-supported>
       <init-param>
           <param-name>encoding</param-name>
           <param-value>UTF-8</param-value>
       </init-param>
       <init-param>
           <param-name>forceEncoding</param-name>
           <param-value>true</param-value>
       </init-param>
   </filter>
   <filter-mapping>
       <filter-name>CharEncoding</filter-name>
       <url-pattern>/*</url-pattern>
   </filter-mapping>

</web-app>
```

配置spring核心配置文件

在src/main/resources下创建applicationContext.xml文件，内容如下

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:context="http://www.springframework.org/schema/context"
       xsi:schemaLocation="
         http://www.springframework.org/schema/beans
         http://www.springframework.org/schema/beans/spring-beans-4.2.xsd
         http://www.springframework.org/schema/context
      http://www.springframework.org/schema/context/spring-context-4.2.xsd
         ">

</beans>
```

配置spring mvc配置文件

在src/main/resources下创建spring-mvc.xml文件，内容如下

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:p="http://www.springframework.org/schema/p"
       xmlns:context="http://www.springframework.org/schema/context"
       xmlns:mvc="http://www.springframework.org/schema/mvc"
       xsi:schemaLocation="
       http://www.springframework.org/schema/beans
       http://www.springframework.org/schema/beans/spring-beans-4.2.xsd
       http://www.springframework.org/schema/context
       http://www.springframework.org/schema/context/spring-context-4.2.xsd
       http://www.springframework.org/schema/mvc
       http://www.springframework.org/schema/mvc/spring-mvc-4.2.xsd">

   <!-- 开启mvc注解 -->
```

```
    <mvc:annotation-driven/>

    <!-- 注解扫描controller -->
    <context:component-scan base-package="com.ssm.controller">
        <context:include-filter type="annotation"
expression="org.springframework.stereotype.Controller"/>
    </context:component-scan>

    <!-- 配置jsp视图 -->
    <bean id="viewResolver"

class="org.springframework.web.servlet.view.InternalResourceViewResolver"
>
        <property name="prefix" value="/WEB-INF/views/"></property>
        <property name="suffix" value=".jsp"></property>
    </bean>

</beans>
```

测试spring spring mvc整合情况

1 在WEB-INF新建文件views用来存放jsp文件
在views下新建一个home.jsp 内容如下

```
<%@ page language="java" import="java.util.*" pageEncoding="UTF-8"%>
<%
String path = request.getContextPath();
String basePath =
request.getScheme()+"://"+request.getServerName()+":"+request.getServerP
ort()+path+"/";
%>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
```

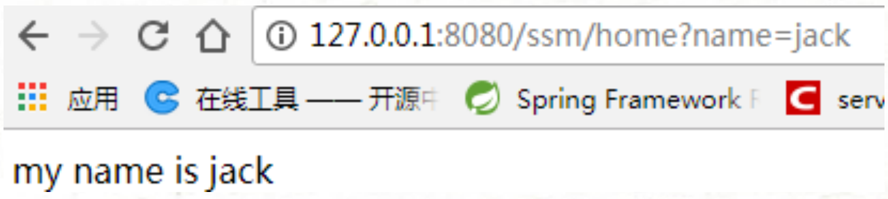
```
  <head>
    <base href="<%=basePath%>">
    
    <title>My JSP 'home.jsp' starting page</title>
    
    <meta http-equiv="pragma" content="no-cache">
    <meta http-equiv="cache-control" content="no-cache">
    <meta http-equiv="expires" content="0">
    <meta http-equiv="keywords" content="keyword1,keyword2,keyword3">
    <meta http-equiv="description" content="This is my page">
    <!--
    <link rel="stylesheet" type="text/css" href="styles.css">
    -->

  </head>

  <body>
    my name is ${name}
  </body>
</html>
```

2 新建一个Controller

```
package com.ssm.controller;
import javax.servlet.http.HttpServletRequest;

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

@Controller
@RequestMapping("/")
public class HomeController {

    @RequestMapping("home")
    public String home(String name , HttpServletRequest request){
        System.out.println("hello " + name);
        request.setAttribute("name", name);
        return "home";
    }

}
```

- 3 重启tomcat 测试

http://127.0.0.1:8080/ssm/home?name=jack



my name is jack

引入mybatis

1 在src/main/resources 新建 数据库连接配置文件 jdbc.properties
2 在src/main/resources 新建 mybatis配置文件 mybatis-config.xml

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE configuration PUBLIC "-//mybatis.org//DTD Config 3.0//EN"
"http://mybatis.org/dtd/mybatis-3-config.dtd">
<configuration>
    <typeAliases>
        <!-- 配置别名扫描包 -->
        <package name="com.ssm.entity"/>
    </typeAliases>
</configuration>
```

3 配置spring配置文件，在上面配置 spring + spring mvc 时 spring核心配置文件并没有任何配置，这里采用声明式 注解事务

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:context="http://www.springframework.org/schema/context"
       xmlns:tx="http://www.springframework.org/schema/tx"
       xmlns:aop="http://www.springframework.org/schema/aop"
       xsi:schemaLocation="
         http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans-4.2.xsd
         http://www.springframework.org/schema/context
http://www.springframework.org/schema/context/spring-context-4.2.xsd
         http://www.springframework.org/schema/tx
http://www.springframework.org/schema/tx/spring-tx-4.2.xsd
         http://www.springframework.org/schema/aop
http://www.springframework.org/schema/aop/spring-aop-4.2.xsd">

        <context:property-placeholder location="classpath:jdbc.properties"/>

        <!-- 自动扫描 -->
        <context:component-scan base-package="com.ssm.service" />

    <bean id="dataSource"

class="org.springframework.jdbc.datasource.DriverManagerDataSource">
                <property name="driverClassName"
value="${jdbc.driverClassName}"/>
                <property name="url" value="${jdbc.url}"/>
                <property name="username" value="${jdbc.username}"/>
                <property name="password" value="${jdbc.password}"/>
        </bean>
        <!-- 配置mybatis的sqlSessionFactory -->
        <bean id="sqlSessionFactory"
class="org.mybatis.spring.SqlSessionFactoryBean">
                <property name="dataSource" ref="dataSource" />
                <!-- 自动扫描mappers.xml文件 -->
                <property name="mapperLocations"
value="classpath:com/ssm/mapper/*.xml"></property>
                <!-- mybatis配置文件 -->
                <property name="configLocation" value="classpath:mybatis-
config.xml"></property>
        </bean>

        <!-- DAO接口所在包名，Spring会自动查找其下的类 -->
        <bean class="org.mybatis.spring.mapper.MapperScannerConfigurer">
                <property name="basePackage" value="com.ssm.dao" />
                <property name="sqlSessionFactoryBeanName"
value="sqlSessionFactory"></property>
        </bean>

        <!-- 配置事务管理器 -->
    <bean id="transactionManager"
class="org.springframework.jdbc.datasource.DataSourceTransactionManager
">
      <property name="dataSource" ref="dataSource" />
    </bean>

    <!-- 配置事物的注解方式注入 -->
    <tx:annotation-driven transaction-manager="transactionManager"/>

</beans>
```

4 User javabean

```java
package com.ssm.entity;

public class User {
	private Integer id;
	private String nickname;
	private String username;
	private String password;
	public Integer getId() {
		return id;
	}
	public void setId(Integer id) {
		this.id = id;
	}
	public String getNickname() {
		return nickname;
	}
	public void setNickname(String nickname) {
		this.nickname = nickname;
	}
	public String getUsername() {
		return username;
	}
	public void setUsername(String username) {
		this.username = username;
	}
	public String getPassword() {
		return password;
	}
	public void setPassword(String password) {
		this.password = password;
	}
}
```

5 UserDAO

```
package com.ssm.dao;

import com.ssm.entity.User;

public interface UserDAO {
	int add(User user);
}
```

6 UserMapper.xml 在 com.ssm.mapper包下

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE mapper
PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"
"http://mybatis.org/dtd/mybatis-3-mapper.dtd">

<mapper namespace="com.ssm.dao.UserDAO">

	<resultMap type="User" id="UserResult">
		<id property="id" column="id"/>
		<result property="nickname" column="nickname"/>
		<result property="username" column="username"/>
		<result property="password" column="password"/>
	</resultMap>

	<insert id="add" parameterType="User">
		insert into t_user(nickname,username,password) values(#
{nickname},#{username},#{password});
	</insert>
</mapper>
```

7 UserService

```
package com.ssm.service;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Transactional;

import com.ssm.dao.UserDAO;
import com.ssm.entity.User;

@Service
public class UserService {

	@Autowired
	private UserDAO userDAO;

	/**
	 * 这里加上事务注解标签，那么当整个方法执行完毕，没有异常才会提交事
务
	 * 如果没有加上注解标签，在 int result = userDao.add(user) 这里就会提交
到数据库
	 */
	@Transactional
	public int add(User user) {
		int result = userDAO.add(user);

		System.out.println("insert");
		String s = null;
		return result;
	}
}
```

8 UserController

```
package com.ssm.controller;

import javax.servlet.http.HttpServletRequest;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestMapping;

import com.ssm.entity.User;
import com.ssm.service.UserService;

@Controller
@RequestMapping("user")
public class UserController {

	@Autowired
	private UserService userService;

	@RequestMapping("add.do")
	public String add(String username ,String password , String nickname ,
HttpServletRequest req) {
		User u = new User();
		u.setNickname(nickname);
		u.setPassword(password);
		u.setUsername(username);
		userService.add(u);
		return "";
	}
}
```

9 测试

可以通过测试类如下： 也可以通过浏览器

测试类需要引入 jar包

```
        <dependency>
                <groupId>org.springframework</groupId>
                <artifactId>spring-test</artifactId>
                <version>${spring-version}</version>
         </dependency>


package com.ssm.test;

import javax.annotation.Resource;

import org.junit.Test;
import org.junit.runner.RunWith;
import org.springframework.test.context.ContextConfiguration;
import org.springframework.test.context.junit4.SpringJUnit4ClassRunner;

import com.ssm.entity.User;
import com.ssm.service.UserService;

@RunWith(SpringJUnit4ClassRunner.class)
@ContextConfiguration(locations={"classpath:applicationContext.xml"})
public class UserTest {

    @Resource
    private UserService userService;

    @Test
    public void test(){
        User user = new User();
        user.setNickname("张4");
        userService.add(user);
    }
}
```

项目附件

ssm.zip
27.82KB