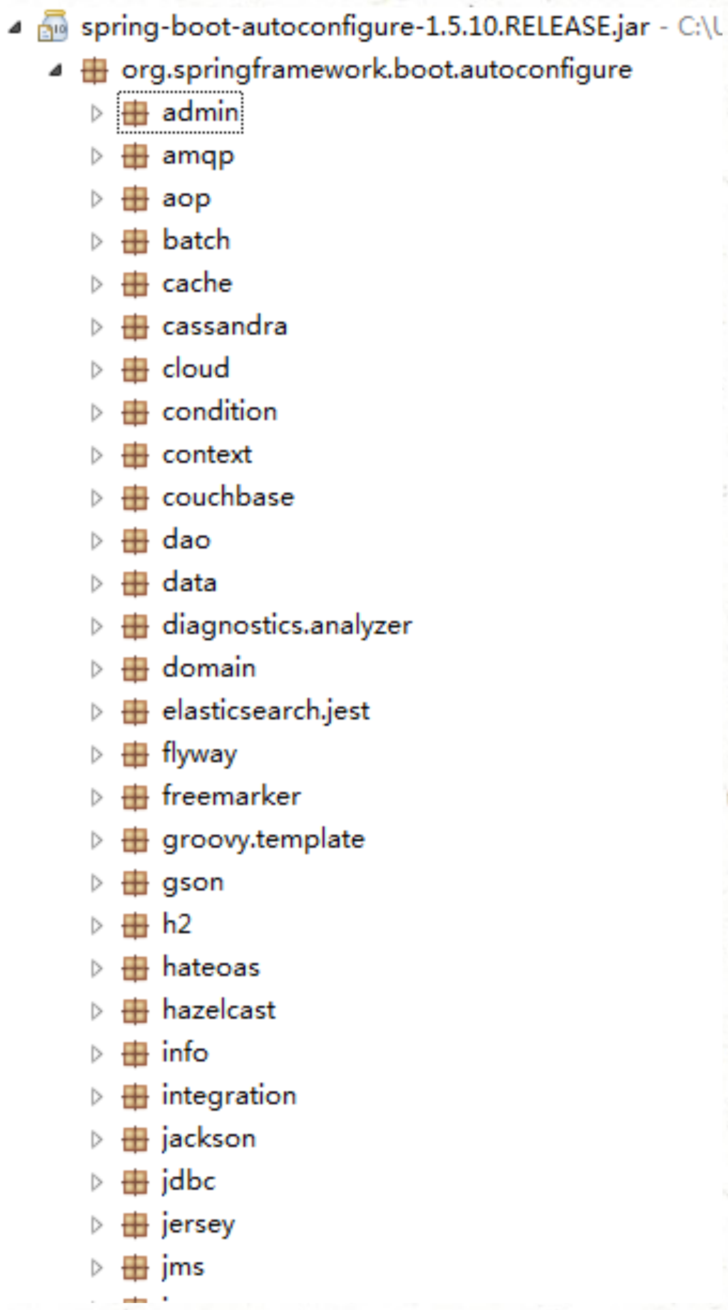
简要说明：

spring boot的作用就是简化开发，集成了绝大多数常用的插件。在spring boot里面叫做 starter pom。通过这些可以简化配置，因为有一些配置spring boot自动装配好了

自动装配的配置都在 org.springframework.boot.autoconfigure的源码包内。



```
spring-boot-autoconfigure-1.5.10.RELEASE.jar - C:\
  org.springframework.boot.autoconfigure
    admin
    amqp
    aop
    batch
    cache
    cassandra
    cloud
    condition
    context
    couchbase
    dao
    data
    diagnostics.analyzer
    domain
    elasticsearch.jest
    flyway
    freemarker
    groovy.template
    gson
    h2
    hateoas
    hazelcast
    info
    integration
    jackson
    jdbc
    jersey
    jms
```

有两种方式查看 装配情况：

1 java -jar xxx.jar --debug

2 在application.properties配置文件中加上 debug=true

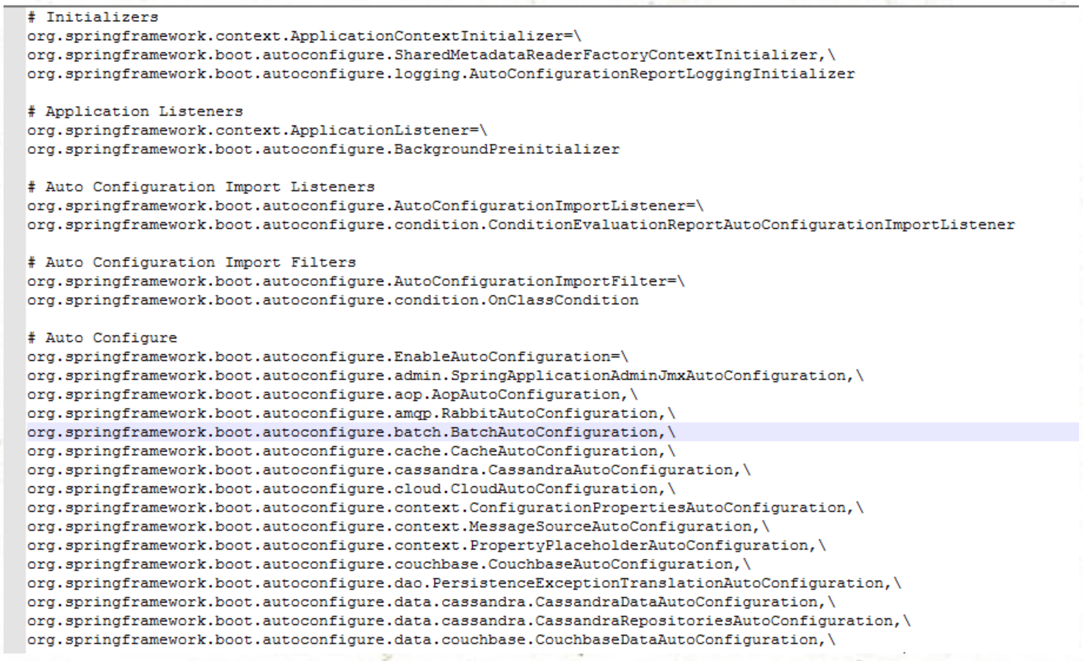
运行原理：

spring boot的核心入口在于 @SpringBootApplication注解，也就是spring boot 启动java 类的地方标注的注解。

@SpringBootApplication注解是一个组合注解，核心功能是由 @EnableAutoConfiguration这个注解提供。

@EnableAutoConfiguration这个注解通过 import 引入了 EnableAutoConfigurationImportSelector.class 这个类，在1.5版本之后 是 AutoConfigurationImportSelector.class这个类。 EnableAutoConfigurationImportSelector继承与 AutoConfigurationImportSelector

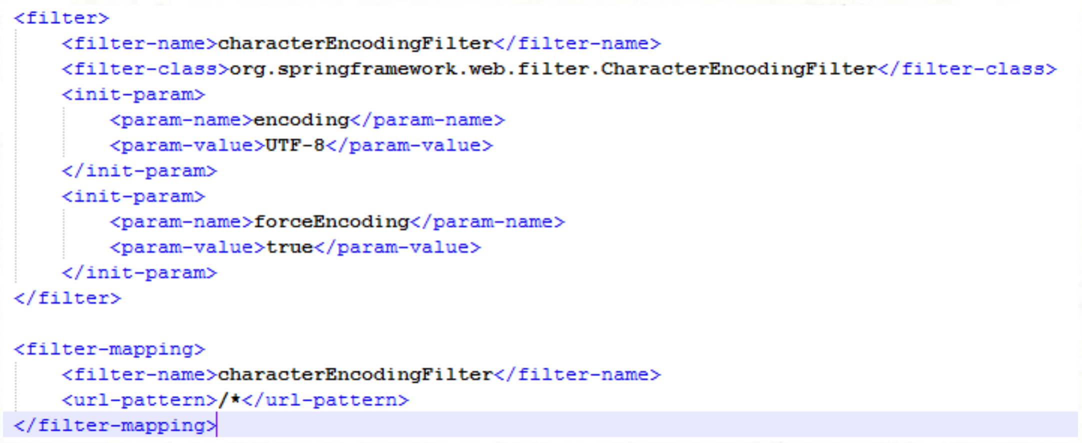
AutoConfigurationImportSelector通过 SpringFactoriesLoader.loadFactoryNames()方法来加载META-INF/spring.factories下面的文件，而spring-boot-autoconfigure-xxxx.jar 下面刚好就有这么一个文件，内如如下：



```
# Initializers
org.springframework.context.ApplicationContextInitializer=\
org.springframework.boot.autoconfigure.SharedMetadataReaderFactoryContextInitializer,\
org.springframework.boot.autoconfigure.logging.AutoConfigurationReportLoggingInitializer

# Application Listeners
org.springframework.context.ApplicationListener=\
org.springframework.boot.autoconfigure.BackgroundPreinitializer

# Auto Configuration Import Listeners
org.springframework.boot.autoconfigure.AutoConfigurationImportListener=\
org.springframework.boot.autoconfigure.condition.ConditionEvaluationReportAutoConfigurationImportListener

# Auto Configuration Import Filters
org.springframework.boot.autoconfigure.AutoConfigurationImportFilter=\
org.springframework.boot.autoconfigure.condition.OnClassCondition

# Auto Configure
org.springframework.boot.autoconfigure.EnableAutoConfiguration=\
org.springframework.boot.autoconfigure.admin.SpringApplicationAdminJmxAutoConfiguration,\
org.springframework.boot.autoconfigure.aop.AopAutoConfiguration,\
org.springframework.boot.autoconfigure.amqp.RabbitAutoConfiguration,\
org.springframework.boot.autoconfigure.batch.BatchAutoConfiguration,\
org.springframework.boot.autoconfigure.cache.CacheAutoConfiguration,\
org.springframework.boot.autoconfigure.cassandra.CassandraAutoConfiguration,\
org.springframework.boot.autoconfigure.cloud.CloudAutoConfiguration,\
org.springframework.boot.autoconfigure.context.ConfigurationPropertiesAutoConfiguration,\
org.springframework.boot.autoconfigure.context.MessageSourceAutoConfiguration,\
org.springframework.boot.autoconfigure.context.PropertyPlaceholderAutoConfiguration,\
org.springframework.boot.autoconfigure.couchbase.CouchbaseAutoConfiguration,\
org.springframework.boot.autoconfigure.dao.PersistenceExceptionTranslationAutoConfiguration,\
org.springframework.boot.autoconfigure.data.cassandra.CassandraDataAutoConfiguration,\
org.springframework.boot.autoconfigure.data.cassandra.CassandraRepositoriesAutoConfiguration,\
org.springframework.boot.autoconfigure.data.couchbase.CouchbaseDataAutoConfiguration,\
```

实例分析：

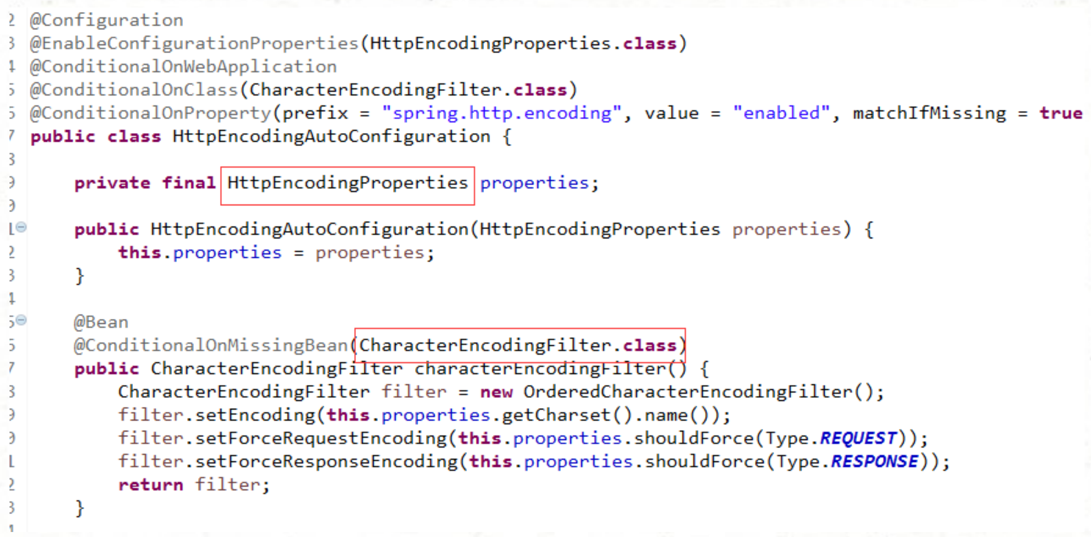
在常规的spring 项目中 web项目进行编码控制的做法是在web.xml中加上过滤器



```
<filter>
    <filter-name>characterEncodingFilter</filter-name>
    <filter-class>org.springframework.web.filter.CharacterEncodingFilter</filter-class>
    <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>characterEncodingFilter</filter-name>
    <url-pattern>/*</url-pattern>
</filter-mapping>
```

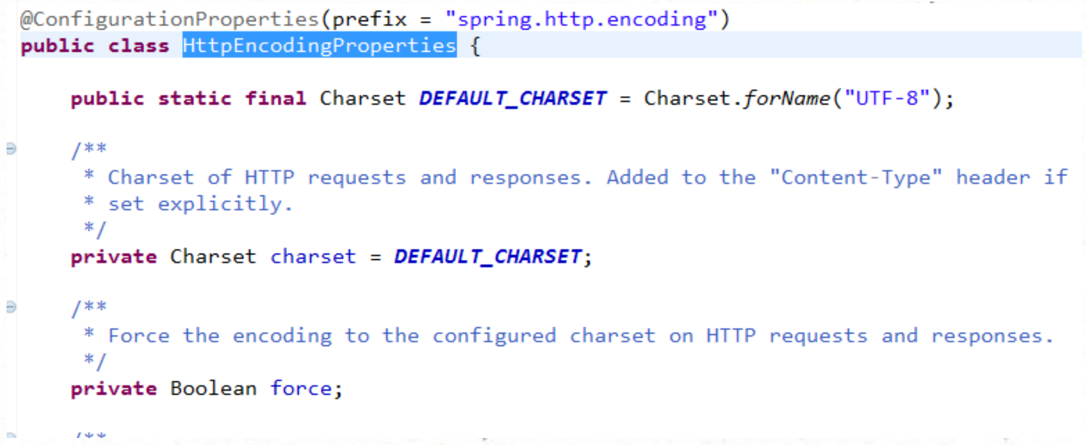
而在spring boot中，通过上面的自动装配配置文件可以找到，org.springframework.boot.autoconfigure.web.HttpEncodingAutoConfiguration 这么一段，说明自动装配进行了编码相关的配置。

查看源码：



```java
@Configuration
@EnableConfigurationProperties(HttpEncodingProperties.class)
@ConditionalOnWebApplication
@ConditionalOnClass(CharacterEncodingFilter.class)
@ConditionalOnProperty(prefix = "spring.http.encoding", value = "enabled", matchIfMissing = true
public class HttpEncodingAutoConfiguration {

    private final HttpEncodingProperties properties;

    public HttpEncodingAutoConfiguration(HttpEncodingProperties properties) {
        this.properties = properties;
    }

    @Bean
    @ConditionalOnMissingBean(CharacterEncodingFilter.class)
    public CharacterEncodingFilter characterEncodingFilter() {
        CharacterEncodingFilter filter = new OrderedCharacterEncodingFilter();
        filter.setEncoding(this.properties.getCharset().name());
        filter.setForceRequestEncoding(this.properties.shouldForce(Type.REQUEST));
        filter.setForceResponseEncoding(this.properties.shouldForce(Type.RESPONSE));
        return filter;
    }
```

里面进行了常规中的过滤器filter-class的设置，编码的值在HTTPEncodingProperties类中进行了设置。代码如图：



```java
@ConfigurationProperties(prefix = "spring.http.encoding")
public class HttpEncodingProperties {

    public static final Charset DEFAULT_CHARSET = Charset.forName("UTF-8");

    /**
     * Charset of HTTP requests and responses. Added to the "Content-Type" header if
     * set explicitly.
     */
    private Charset charset = DEFAULT_CHARSET;

    /**
     * Force the encoding to the configured charset on HTTP requests and responses.
     */
    private Boolean force;
```

可以通过spring boot的配置文件对这些值进行设置，如果不设置那么就是默认值。设置方式如下：

spring.http.encoding.charset=

spring.http.encoding.force=

自定义starter pom

http://blog.csdn.net/liuchuanhong1/article/details/55057135

https://gitee.com/jiuxiao/mystarter-spring-boot-starter.git