설정 자동화와 설정의 변경, @EnableWebMvc와 WebMvcConfigurer

Spring에서 제공하는 설정의 자동화와 변경

@Enable~ 을 이용한 설정 자동화

- 기존
  - 메세지를 변환하는 메세지 컨버터
  - 뷰 리졸버
  - 등 을 일일이 설정해야하고 새 프로젝트 만들때마다 동일한 설정을 해야했다.
- @Enable~ 어노테이션
  - @Configuration 설정 클래스에 붙임
- @EnableWebMvc
  - 스프링이 제공하는 웹과 관련된 최신 전략 빈들이 등록

```
@Configuration
@EnableWebMvc
public class WebMvcConfig {

}
```

~Configurer 인터페이스을 통한 설정의 변경

- 스프링이 제공해주는 자동 설정들 외에 추가의 설정이 필요할 때
- @Enable로 적용되는 인프라 빈에 대해 추가적인 설정을 할 수 있도록, ~Configurer로 끝나는 인터페이스(빈 설정자)를 제공
- @EnableWebMvc의 빈 설정자는 WebMvcConfigurer
- WebMvcConfigurer를 구현한 클래스를 만들고 @Configuration을 붙여 빈으로 등록

@EnableWebMvc와 WebMvcConfigurer

WebMvcConfigurer 인터페이스

- @EnableWebMvc를 통해 자동 설정되는 빈들의 설정자는 WebMvcConfigurer
- 메소드 이름 형태
  - add~ : 기본 설정이 없는 빈들에 대해 새로운 빈을 추가
  - configurer~ : 수정자를 통해 기존의 설정을 대신해 등록
  - extend~ : 기존의 설정을 이용해 추가로 설정을 확장

```
public interface WebMvcConfigurer{
    // 경로 매칭 설정을 구성합니다.
    default void configurePathMatch(PathMatchConfigurer configurer) {
    }
```

```
    // 컨텐츠 협상 설정을 구성합니다.
    default void configureContentNegotiation(ContentNegotiationConfigurer
configurer) {
    }

    // 비동기 지원 설정을 구성합니다.
    default void configureAsyncSupport(AsyncSupportConfigurer configurer)
{
    }

    // Default Servlet 처리를 구성합니다.
    default void
configureDefaultServletHandling(DefaultServletHandlerConfigurer
configurer) {
    }

    // Formatter와 Converter를 추가합니다.
    default void addFormatters(FormatterRegistry registry) {
    }

    // 인터셉터를 추가합니다.
    default void addInterceptors(InterceptorRegistry registry) {
    }

    // 정적 리소스 핸들러를 추가합니다.
    default void addResourceHandlers(ResourceHandlerRegistry registry) {
    }

    // CORS 매핑을 추가합니다.
    default void addCorsMappings(CorsRegistry registry) {
    }

    // 뷰 컨트롤러를 추가합니다.
    default void addViewControllers(ViewControllerRegistry registry) {
    }

    // 뷰 리졸버를 구성합니다.
    default void configureViewResolvers(ViewResolverRegistry registry) {
    }

    // 메소드 인자 리졸버를 추가합니다.
    default void addArgumentResolvers(List<HandlerMethodArgumentResolver>
resolvers) {
    }

    // 메소드 반환 값 핸들러를 추가합니다.
    default void
addReturnValueHandlers(List<HandlerMethodReturnValueHandler> handlers) {
    }

    // 메시지 컨버터를 구성합니다.
    default void configureMessageConverters(List<HttpMessageConverter<?>>
converters) {
    }
```

```
    // 기존의 메시지 컨버터를 확장합니다.
    default void extendMessageConverters(List<HttpMessageConverter<?>>
converters) {
    }

    // 예외 처리 핸들러를 구성합니다.
    default void
configureHandlerExceptionResolvers(List<HandlerExceptionResolver>
resolvers) {
    }

    // 기존의 예외 처리 핸들러를 확장합니다.
    default void
extendHandlerExceptionResolvers(List<HandlerExceptionResolver> resolvers)
{
    }

    // Validator 인스턴스를 반환합니다. 기본값은 null입니다.
    @Nullable
    default Validator getValidator() {
        return null;
    }

    // MessageCodesResolver 인스턴스를 반환합니다. 기본값은 null입니다.
    @Nullable
    default MessageCodesResolver getMessageCodesResolver() {
        return null;
    }
}
```

- 기본적으로 등록되지 않은것
    - interceptor는 addInterceptors() 메소드 제공
- 기본적으로 등록된 것
    - messageConveter
    - configureMessageConverters()와 extendMessageConverters ()

WebMvcConfigurer를 통한 설정의 변경 예시

- 상황
    - json기반 메세지 컨버터 구성에 더해 xml 기반 메세지 컨버터가 필요한 상황
    - 우리는 기존 메세지 컨버터를 확장해야하므로 extendMessageConverter를 오버라이딩
        - extend로 시작하는 이유 : 새로운 메세지 컨버터를 추가해도, 기존의 메세지 컨버터도 모두 등록됨
    - 기존 메세지 컨버터를 모두 비활성화하고 새로운 메세지 컨버터만 추가하고싶은 경우
        - configurerMessageConverter을 이용

```
   @Configuration
public class WebMvcConfig implements WebMvcConfigurer {
```

```
    @Override
    public void extendMessageConverters(List<HttpMessageConverter<?>>
messageConverters) {
        messageConverters.add(createXmlHttpMessageConverter());
    }

    private HttpMessageConverter<Object> createXmlHttpMessageConverter() {
        MarshallingHttpMessageConverter xmlConverter = new
MarshallingHttpMessageConverter();

        XStreamMarshaller xstreamMarshaller = new XStreamMarshaller();
        xmlConverter.setMarshaller(xstreamMarshaller);
        xmlConverter.setUnmarshaller(xstreamMarshaller);

        return xmlConverter;
    }

}
```