Annotation Interface WebMvcTest

```
@Target (TYPE )
@Retention (RUNTIME )
@Documented
@Inherited
@BootstrapWith
(org.springframework.boot.webmvc.test.autoconfigure.WebMvcTestContextBootstra
@ExtendWith(SpringExtension .class)
@OverrideAutoConfiguration(enabled=false)
@TypeExcludeFilters(org.springframework.boot.webmvc.test.autoconfigure.WebMvc
@AutoConfigureWebMvc
@AutoConfigureMockMvc
@ImportAutoConfiguration
public @interface WebMvcTest
```

Annotation that can be used for a Spring MVC test that focuses **only** on Spring MVC components.

Using this annotation only enables auto-configuration that is relevant to MVC tests. Similarly, component scanning is limited to beans annotated with:

- `@Controller`
- `@ControllerAdvice`
- `@JacksonComponent`
- `@JsonComponent` (Jackson 2, deprecated)

as well as beans that implement:

- `Converter`
- `DelegatingFilterProxyRegistrationBean`
- `ErrorAttributes`
- `Filter`
- `FilterRegistrationBean`
- `GenericConverter`
- `HandlerInterceptor`
- `HandlerMethodArgumentResolver`
- `HttpMessageConverter`
- `IDialect`, if Thymeleaf is available
- `JacksonModule`, if Jackson is available
- `Module` (deprecated), if Jackson 2 is available
- `SecurityFilterChain`
- `WebMvcConfigurer`
- `WebMvcRegistrations`
- `WebSecurityConfigurer`

By default, tests annotated with `@WebMvcTest` will also auto-configure Spring Security and `MockMvc` (include support for HtmlUnit WebClient and Selenium WebDriver). For more fine-grained control of MockMVC the `@AutoConfigureMockMvc` annotation can be used.

Typically `@WebMvcTest` is used in combination with `@MockitoBean` or `@Import` to create any collaborators required by your `@Controller` beans.

If you are looking to load your full application configuration and use MockMVC, you should consider `@SpringBootTest` combined with `@AutoConfigureMockMvc` rather than this annotation.

When using JUnit 4, this annotation should be used in combination with `@RunWith(SpringRunner.class)`.

Since:

4.0.0

Author:

Phillip Webb, Artsiom Yudovin

See Also:

`AutoConfigureWebMvc`, `AutoConfigureMockMvc`

Optional Element Summary

Optional Elements

Modifier and Type	Optional Element	Description
`Class<?>[]`	**`controllers`**	Specifies the controllers to test.
`Class<?>[]`	**`excludeAutoConfiguration`**	Auto-configuration exclusions that should be applied for this test.
`ComponentScan.Filter[]`	**`excludeFilters`**	A set of exclude filters which can be used to filter beans that would otherwise be added to the application context.
`ComponentScan.Filter[]`	**`includeFilters`**	A set of include filters which can be used to add otherwise filtered beans to the application context.
`String[]`	**`properties`**	Properties in form key=value that should be added to the Spring `Environment` before the test runs.
`boolean`	**`useDefaultFilters`**	Determines if default filtering should be used with `@SpringBootApplication`.

`Class <?>[]`	**`value`**	Specifies the controllers to test.

Element Details

properties

```
String [] properties
```

Properties in form key=value that should be added to the Spring `Environment` before the test runs.

Returns:

the properties to add

Default:

`{}`

value

```
@AliasFor ("controllers")
Class <?>[] value
```

Specifies the controllers to test. This is an alias of `controllers()` which can be used for brevity if no other attributes are defined. See `controllers()` for details.

Returns:

the controllers to test

See Also:

`controllers()`

Default:

`{}`

controllers

```
@AliasFor ("value")
Class <?>[] controllers
```

Specifies the controllers to test. May be left blank if all `@Controller` beans should be added to the application context.

Returns:

the controllers to test

See Also:

`value()`

Default:

`{}`

useDefaultFilters

```
boolean useDefaultFilters
```

Determines if default filtering should be used with `@SpringBootApplication`. By default only `@Controller` (when no explicit `controllers` are defined), `@ControllerAdvice` and `WebMvcConfigurer` beans are included.

Returns:

if default filters should be used

See Also:

`includeFilters()`, `excludeFilters()`

Default:

`true`

includeFilters

```
ComponentScan.Filter [] includeFilters
```

A set of include filters which can be used to add otherwise filtered beans to the application context.

Returns:

include filters to apply

Default:

`{}`

excludeFilters

```
ComponentScan.Filter [] excludeFilters
```

A set of exclude filters which can be used to filter beans that would otherwise be added to the application context.

Returns:

exclude filters to apply

Default:

`{}`

excludeAutoConfiguration

```
@AliasFor (annotation =ImportAutoConfiguration.class,
          attribute ="exclude")
Class <?>[] excludeAutoConfiguration
```

Auto-configuration exclusions that should be applied for this test.

Returns:

auto-configuration exclusions to apply

Default:

`{}`