Spring MVC - Cache-Control support

[Updated: Feb 18, 2017, Created: Jan 29, 2017]

Spring web MVC provides different ways to configure "Cache-Control" headers for an application. In this tutorial we will demonstrate how to set "Cache-Control" header in the @Controller methods for different scenarios.

Examples

Setting "Cache-Control" with handler returning a view name

The Controller

```
@Controller
public class TheController {

  @RequestMapping(value = "/test1")
  public String handle1 (HttpServletResponse response) {
      String headerValue = CacheControl.maxAge(10, TimeUnit.SECONDS)
                                       .getHeaderValue();

      response.addHeader("Cache-Control", headerValue);
      return "myView";
  }
    .............
}
```

CacheControl class is based on builder pattern, a very convenient way to create 'Cache-Control' headers with different directives. In above code we are specifying that browser cache should expire after 10 secs.

/src/main/webapp/WEB-INF/pages/myView.jsp

```
<%@ page language="java"
    contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<%@ page import="java.time.LocalDateTime"%>
<html>
  <body style="margin:20px;">
  JSP page
<p> Page Created:  <%= LocalDateTime.now()%></p>
<a href='test1'>test1</a>
  </body>
</html>
```

Note that, in above JSP, we are printing LocalDateTime.now(), so that we can confirm that our page is not fetched from the server more than once, within 10 sec of cache max age.

Spring boot main class

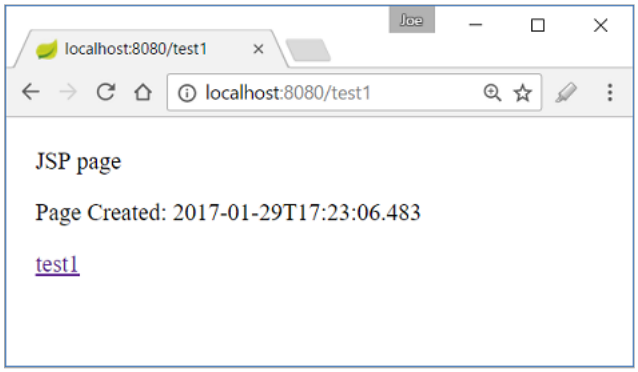
```
@SpringBootApplication
public class Main {
```

```
  public static void main (String[] args) {
      SpringApplication.run(Main.class, args);
  }
    .............
}
```

Running application

```
mvn spring-boot:run
```

Output:



Clicking on 'test1' link will retrieve the page from browser's local cache, but clicking on reload button (or F5) will reload the page from server. This happens if we are on the same tab. If we open new tab or new browser window and enter the url in the address bar, the page will be retrieved from the cache. This behavior is consistent in the latest versions of Chrome, FireFox and Edge browsers. (a discussion here).

If we click on the link 'test1' various times within 10 secs period then the page created date will not change, after 10 secs it will change. That confirms that browser is using local cache within those 10 secs instead of making server requests.

Setting "Cache-Control" with handler returning ResponseEntity

In this case the controller returns RESTful style response by using @ResponseBody and returning ResponseEntity object:

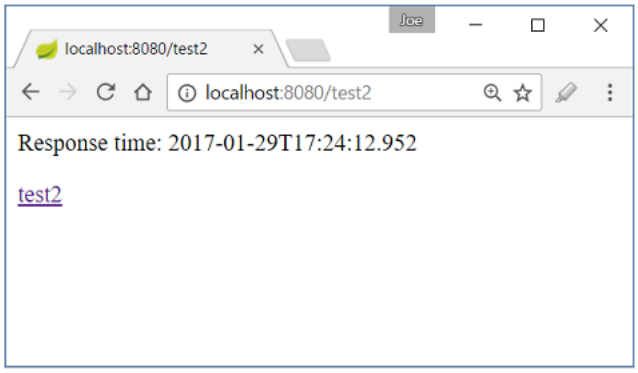
The Controller

```
@Controller
public class TheController {
    .............
  @ResponseBody
  @RequestMapping(value = "/test2")
  public ResponseEntity<String> handle2 () {

      CacheControl cacheControl = CacheControl.maxAge(10, TimeUnit.SECONDS);

      String testBody = "<p>Response time: " + LocalDateTime.now() +
                "</p><a href=''>test2</a>";
      return ResponseEntity.ok()
                           .cacheControl(cacheControl)
                           .body(testBody);
  }
}
```

output



Again, clicking on test2 link won't reload the page from server within 10 secs period.

Setting "Cache-Control" for the static resources

We need to use ResourceHandlerRegistration#setCachePeriod(..) while registering the resource location.

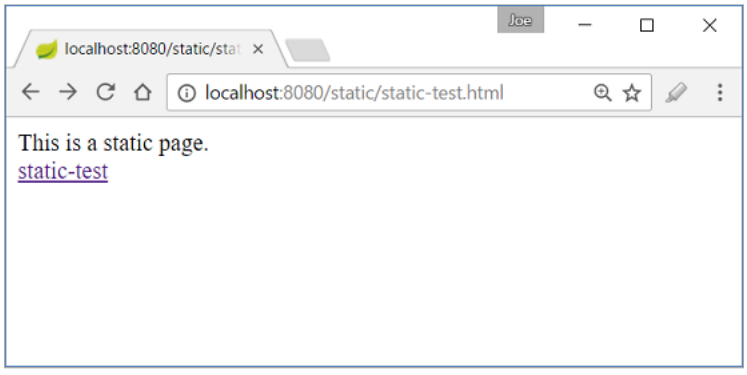
```
@SpringBootApplication
public class Main {
    .............
  @Bean
  public WebMvcConfigurerAdapter webConfigurer () {
      return new WebMvcConfigurerAdapter() {
          @Override
          public void addResourceHandlers (ResourceHandlerRegistry registry) {
              registry.addResourceHandler("/static/**")
                      .addResourceLocations("/static/")
                      .setCachePeriod(30);
          }
      };
  }
}
```

All static pages under static folder (in classpath) will cause to create browser cache for 30 secs.

/src/main/webapp/static/static-test.html

```
<html>
<body>
This is a static page.
<br/>
<a href="static-test.html">static-test</a>
</body>
</html>
```

Output



Clicking on static-test link won't reload the page from server within 30 secs period. To confirm, we can change content of the html page and click the link again, the page won't update from the server. Note: spring-boot:run

executes the application in exploded form and contents under webapp are not copied to the 'target' folder, that's the reason we can make changes under webapp folder without restarting the server. Even if we restart after modifying static-test.html and access the page again (without F5) the page will be retrieved from the browser cache without making a server request.

Default Cache-Control

According to the specification , if there's no cache-control header specified by the response, the browser will always cache the contents. It will probably be cached for a long period or may be for good, unless user uses F5 or Ctrl+F5 (hard refresh) or clears the cache manually by using browser settings.

In the next tutorials, we will demonstrate how to set Last-Modified/If-Modified-Since and ETag headers in Spring MVC.

Example Project

Dependencies and Technologies Used:

- spring-boot-starter-web 1.4.4.RELEASE: Starter for building web, including RESTful, applications using Spring MVC. Uses Tomcat as the default embedded container.
  Corresponding Spring version: 4.3.6.RELEASE
- tomcat-embed-jasper 8.5.11: Core Tomcat implementation.
- JDK 1.8
- Maven 3.3.9

JAX-RS - Injecting @Context annotati
Jackson JSON pars
Spring MVC - Form Validator Interface
Java Swing - JCom Highlighting Exam
Spring MVC - Form Validation API and
JPA - Detached En
Java 9 - Platform L
Java Util Logging -
Spring ORM - Usin LocalContainerEnti setting up JPA Enti Programmatically

Spring Cache Control Example

```
cache-control-example
  src
    main
      java
        com
          logicbig
            example
              Main.java
              TheController.java
      resources
        application.properties
      webapp
        WEB-INF
          pages
            myView.jsp
          static
            static-test.html
  pom.xml
```

```java
package com.logicbig.example;

import org.springframework.http.CacheControl;
import org.springframework.http.ResponseEntity;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.ResponseBody;

import javax.servlet.http.HttpServletResponse;
import java.time.LocalDateTime;
import java.util.concurrent.TimeUnit;

@Controller
public class TheController {

    @RequestMapping(value = "/test1")
    public String handle1 (HttpServletResponse response) {
        String headerValue = CacheControl.maxAge(10, TimeUnit.SECONDS)
                                         .getHeaderValue();

        response.addHeader("Cache-Control", headerValue);
        return "myView";
    }

    @ResponseBody
    @RequestMapping(value = "/test2")
    public ResponseEntity<String> handle2 () {
```

Project Structure

```
cache-control-example
  src
    main
      java
```

```
        com
          logicbig
            example
              Main.java
              TheController.java
      resources
        application.properties
      webapp
        WEB-INF
          pages
            myView.jsp
        static
          static-test.html
  pom.xml
```

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
 <modelVersion>4.0.0</modelVersion>

 <groupId>com.logicbig.example</groupId>
 <artifactId>cache-control-example</artifactId>
 <version>1.0-SNAPSHOT</version>
 <packaging>war</packaging>

 <parent>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-parent</artifactId>
  <version>1.4.4.RELEASE</version>
 </parent>

 <properties>
  <java.version>1.8</java.version>
 </properties>
 <dependencies>
  <dependency>
   <groupId>org.springframework.boot</groupId>
   <artifactId>spring-boot-starter-web</artifactId>
  </dependency>
  <dependency>
   <groupId>org.apache.tomcat.embed</groupId>
   <artifactId>tomcat-embed-jasper</artifactId>
   <scope>provided</scope>
  </dependency>
 </dependencies>

 <build>
  <plugins>
   <plugin>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-maven-plugin</artifactId>
   </plugin>
  </plugins>
 </build>

</project>
```

```
package com.logicbig.example;
```

```java
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.context.annotation.Bean;
import org.springframework.web.servlet.config.annotation.ResourceHandlerRegistry;
import org.springframework.web.servlet.config.annotation.WebMvcConfigurerAdapter;

@SpringBootApplication
public class Main {

    public static void main (String[] args) {
        SpringApplication.run(Main.class, args);
    }

    @Bean
    public WebMvcConfigurerAdapter webConfigurer () {
        return new WebMvcConfigurerAdapter() {
            @Override
            public void addResourceHandlers (ResourceHandlerRegistry registry) {
                registry.addResourceHandler("/static/**")
                        .addResourceLocations("/static/")
                        .setCachePeriod(30);
            }
        };
    }
}
```

```java
package com.logicbig.example;

import org.springframework.http.CacheControl;
import org.springframework.http.ResponseEntity;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.ResponseBody;

import javax.servlet.http.HttpServletResponse;
import java.time.LocalDateTime;
import java.util.concurrent.TimeUnit;

@Controller
public class TheController {

    @RequestMapping(value = "/test1")
    public String handle1 (HttpServletResponse response) {
        String headerValue = CacheControl.maxAge(10, TimeUnit.SECONDS)
                                         .getHeaderValue();

        response.addHeader("Cache-Control", headerValue);
        return "myView";
    }

    @ResponseBody
    @RequestMapping(value = "/test2")
    public ResponseEntity<String> handle2 () {

        CacheControl cacheControl = CacheControl.maxAge(10, TimeUnit.SECONDS);

        String testBody = "<p>Response time: " + LocalDateTime.now() +
                  "</p><a href=''>test2</a>";
        return ResponseEntity.ok()
                             .cacheControl(cacheControl)
                             .body(testBody);
```

```
    }
}
```

```
spring.mvc.view.prefix= /WEB-INF/pages/
spring.mvc.view.suffix= .jsp
spring.messages.basename= msgs/msg
```

```
<html>
<body>
This is a static page.
<br/>
<a href="static-test.html">static-test</a>
</body>
</html>
```

```
<%@ page language="java"
    contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<%@ page import="java.time.LocalDateTime"%>
<html>
  <body style="margin:20px;">
  JSP page
<p> Page Created:  <%= LocalDateTime.now()%></p>
<a href='test1'>test1</a>
  </body>
</html>
```

See Also

- "Cache-Control" specifications.
- Spring Boot Tutorials
- Spring Core Tutorials
- Global Exception handling with @ControllerAdvice
- Mixing web.xml and Spring exception handling
- Ordering and customization of default HandlerExceptionResolvers
- Using multiple HandlerExceptionResolvers together
- Creating a custom HandlerExceptionResolver
- Understanding Spring internal MVC Exceptions
- Using @ResponseStatus on Exception classes
- Annotation based Exception handling using @ExceptionHandler
- Exception to view mapping using SimpleMappingExceptionResolv
- Spring MVC quick examples

Java 9 Tutorials

- Java 9 Module System
- Java 9 JShell
- Java 9 Misc Features

Recent Tutorials

- Using Entity Subgraph with @NamedSubgraph
- Spring Boot - Custom Error Page in Thymeleaf
- Java Swing - Understanding TreeNode and Creating JTree from DefaultMutableTreeNode
- Java Swing - Using custom TreeCellRenderer for JTree
- Java Swing - Creating JTree from a Hashtable and listening to selections
- Spring MVC - Handling HTTP PUT Request
- PrimeFaces - Ajax ProgressBar Example
- Understanding Entity Graphs, @NamedEntityGraph with @NamedAttributeNode Example
- Spring Boot - Custom Error Page in JSP
- Java Swing - JTable Lazy Pagination with JPA backend and @DisplayAs annotation to map columns/rows via reflection
- Java Swing - JTable Lazy Pagination with JPA

Spring MVC - Testing Multipart uploads with MockMultipartFile

Spring MVC - File upload by using StandardServletMultipartResolver

Spring MVC - Understanding MultipartResolver

Bean Validation integration in JPA

Spring Boot - Using Mustache View

Spring Boot - Using Thymeleaf View

PrimeFaces - Client Side ProgressBar Example

Spring Security - HTTP/HTTPS Channel Security

Prev