```java
package com.shinyoung.spring.controller;

import com.fasterxml.jackson.databind.
ObjectMapper;
import com.shinyoung.spring.domain.Article;
import com.shinyoung.spring.dto.
AddArticleRequest;
import com.shinyoung.spring.dto.
UpdateArticleRequest;
import com.shinyoung.spring.repository.
BlogRepository;
import org.junit.jupiter.api.BeforeEach;
import org.junit.jupiter.api.DisplayName;
import org.junit.jupiter.api.Test;
import org.springframework.beans.factory.
annotation.Autowired;
import org.springframework.boot.test.
autoconfigure.web.servlet.
AutoConfigureMockMvc;
import org.springframework.boot.test.context.
SpringBootTest;
import org.springframework.http.MediaType;
import org.springframework.test.web.servlet.
MockMvc;
import org.springframework.test.web.servlet.
ResultActions;
import org.springframework.test.web.servlet.
setup.MockMvcBuilders;
import org.springframework.web.context.
WebApplicationContext;

import java.util.List;


```

```
24 import static org.assertj.core.api.Assertions
   .assertThat;
25 import static org.springframework.test.web.
   servlet.request.MockMvcRequestBuilders.get;
26 import static org.springframework.test.web.
   servlet.request.MockMvcRequestBuilders.post;
27 import static org.springframework.test.web.
   servlet.request.MockMvcRequestBuilders.delete
   ;
28 import static org.springframework.test.web.
   servlet.request.MockMvcRequestBuilders.put;
29 import static org.springframework.test.web.
   servlet.result.MockMvcResultMatchers.jsonPath
   ;
30 import static org.springframework.test.web.
   servlet.result.MockMvcResultMatchers.status;
31
32
33 @SpringBootTest
34 @AutoConfigureMockMvc
35 class BlogApiControllerTest {
36
37     @Autowired
38     protected MockMvc mockMvc;
39
40     @Autowired
41     private ObjectMapper objectMapper;
42
43     @Autowired
44     private WebApplicationContext context;
45
46     @Autowired
47     BlogRepository blogRepository;
48
49     @BeforeEach
```

```
50      public void mockMvcSetUp() {
51          this.mockMvc = MockMvcBuilders.
    webAppContextSetup(context).build();
52
53          blogRepository.deleteAll();
54      }
55
56      @DisplayName("addArticle: 블로그 글 추가에
    성공한다.")
57      @Test
58      public void addArticle() throws Exception
     {
59          final String url = "/api/articles";
60          final String title = "title";
61          final String content = "content";
62          final AddArticleRequest userRequest
     = new AddArticleRequest(title, content);
63
64          final String requestBody =
    objectMapper.writeValueAsString(userRequest);
65
66          // when
67          ResultActions result = mockMvc.
    perform(post(url)
68                  .contentType(MediaType.
    APPLICATION_JSON_VALUE)
69                  .content(requestBody));
70
71          //then
72          result.andExpect(status().isCreated
    ());
73
74          List<Article> articles =
    blogRepository.findAll();
75
```

```
 76          assertThat(articles.size()).
    isEqualTo(1);
 77          assertThat(articles.get(0).getTitle
    ()).isEqualTo(title);
 78          assertThat(articles.get(0).
    getContent()).isEqualTo(content);
 79
 80      }
 81
 82
 83      @DisplayName("find All Article: 블로그 글
    목록 조회에 성공")
 84      @Test
 85      public void findAllArticles() throws
    Exception {
 86
 87
 88          final String url = "/api/articles";
 89          final String title = "title";
 90          final String content = "content";
 91
 92          blogRepository.save(Article.builder
    ().title(title).content(content).build());
 93
 94
 95          final ResultActions resultActions =
    mockMvc.perform(get(url)
 96                  .accept(MediaType.
    APPLICATION_JSON_VALUE));
 97
 98          resultActions
 99                  .andExpect(status().isOk())
100                  .andExpect(jsonPath("$[0].
    content").value(content))
101                  .andExpect(jsonPath("$[0].
```

```
101 title").value(title));
102
103
104     }
105
106     @DisplayName("find Article: 블로그 글 조회
    성고")
107     @Test
108     public void findArticle() throws
    Exception {
109         final String url = "/api/articles/{
    id}";
110         final String title = "title";
111         final String content = "content";
112
113         Article savedArticle =
    blogRepository.save(Article.builder().
    content(content).title(title).build());
114
115         final ResultActions resultActions =
    mockMvc.perform(get(url, savedArticle.getId
    ()));
116
117         System.out.println(
    "++++++++++++++++++++");
118         System.out.println(savedArticle.
    getId());
119         System.out.println(
    "++++++++++++++++++++");
120
121         resultActions
122                 .andExpect(status().isOk())
123                 .andExpect(jsonPath("$.
    content").value(content))
124                 .andExpect(jsonPath("$.title
```

```
124 ").value(title));
125
126
127     }
128
129     @Test
130     @DisplayName("delete Article: 블로그 글
    삭제 성공")
131     public void deleteArticle() throws
    Exception {
132
133         final String url = "/api/articles/{
    id}";
134         final String title = "title";
135         final String content = "content";
136         Article savedArticle =
    blogRepository.save(Article.builder().
    content(content).title(title).build());
137
138         mockMvc.perform(delete(url,
    savedArticle.getId())).andExpect(status().
    isOk());
139
140         List<Article> articles =
    blogRepository.findAll();
141
142         assertThat(articles).isEmpty();
143     }
144
145     @Test
146     @DisplayName("update article: 블로그 글
    수정 성공")
147     public void updateArticle() throws
    Exception {
148         final String url = "/api/articles/{
```

```
148 id}";
149         final String title = "title";
150         final String content = "content";
151         Article savedArticle =
    blogRepository.save(Article.builder().
    content(content).title(title).build());
152
153
154         final String newTitle = " new 11
    title";
155         final String newContent = "new 11
    content";
156
157         UpdateArticleRequest request = new
    UpdateArticleRequest(newTitle, newContent);
158
159         ResultActions resultActions =
    mockMvc.perform(put(url, savedArticle.getId
    ()).contentType(MediaType.
    APPLICATION_JSON_VALUE).content(objectMapper
    .writeValueAsString(request)));
160
161         resultActions.andExpect(status().
    isOk());
162
163         Article article = blogRepository.
    findById(savedArticle.getId()).get();
164
165         assertThat(article.getTitle()).
    isEqualTo(newTitle);
166         assertThat(article.getContent()).
    isEqualTo(newContent);
167
168
169
```

```
170
171     }
172
173 }
```