处理创建请求

@RequestBody：映射请求体到java方法的参数

```
@PostMapping
 public User create(@RequestBody User user) {
 System.out.println(user.getUsername());
 System.out.println(user.getPassword());
 System.out.println(user.getId());

 user.setId("1");
 return user;
 }
```

```
@Test
 public void whenCreateSuccess() throws Exception {
 String content = "{\"username\":\"tom\",\"password\":null}";
 mockMvc.perform(post("/user").contentType(MediaType.APPLICATION_JSON_UTF8)
 .content(content))
 .andExpect(status().isOk())
 .andExpect(jsonPath("$.id").value("1"));
 }
```

日期类型参数的处理：使用时间戳作为参数，时间戳可以和时间类型相互转换

```
@Test
 public void whenCreateSuccess() throws Exception {
 Date date = new Date();
 System.out.println(date.getTime());
 String content = "{\"username\":\"tom\",\"password\":null,\"birth\":"+date.getTime()+"}";
 String result = mockMvc.perform(post("/user").contentType(MediaType.APPLICATION_JSON_UTF8)
 .content(content))
 .andExpect(status().isOk())
 .andExpect(jsonPath("$.id").value("1"))
  .andReturn().getResponse().getContentAsString();
  System.out.println(result);
  }
```

```
@PostMapping
 public User create(@RequestBody User user) {

```

```java
System.out.println(user.getUsername());
System.out.println(user.getPassword());
System.out.println(user.getId());
System.out.println(user.getBirth());

user.setId("1");
 return user;
 }
```

@Valid注解和BindingResult：验证请求参数的合法性并处理校验结果

@Valid

```java
@NotBlank
private String password;

 public User create(@Valid @RequestBody User user) {
```

BindingResult 必须和@Valid一起使用

```java
public User create(@Valid @RequestBody User user, BindingResult errors) {
 if(errors.hasErrors()) {
 errors.getAllErrors().stream().forEach(error ->
System.out.println(error.getDefaultMessage()));
 }
```