IT Skill Test GIC Myanmar

Duration: 60 Minutes　　　　Total Questions: 15

---

1. In a Spring MVC application, you need to pass a list of user objects from a controller to a view. Which of the following approaches is the most appropriate?

A.
```
@Controller
public class UserController {
    @GetMapping("/users")
    public String listUsers(HttpServletRequest request) {
        request.setAttribute("users", userService.getAllUsers());
        return "userList";
    }
}
```

B.
```
@Controller
public class UserController {
    @GetMapping("/users")
    public ModelAndView listUsers() {
        ModelAndView mav = new ModelAndView("userList");
        mav.addObject("users", userService.getAllUsers());
        return mav;
    }
}
```

C.
```
@Controller
public class UserController {
    @GetMapping("/users")
    public String listUsers(Model model) {
        model.addAttribute("users", userService.getAllUsers());
        return "userList";
    }
}
```

```
D. @RestController
   public class UserController {
       @GetMapping("/users")
       public List<User> listUsers() {
           return userService.getAllUsers();
       }
   }
```

2. In a Spring MVC application, you need to implement a controller method that accepts both GET and POST requests. Which annotation should you use?

   A.@RequestMapping(method = {RequestMethod.GET, RequestMethod.POST})

   B.@GetMapping @PostMapping

   C.@Controller(methods = {"GET", "POST"})

   D.@RequestMapping(value = "/", method = RequestMethod.ALL)

3. You're working on a Spring MVC application and need to implement a feature where a user can upload a file. Which of the following code snippets correctly handles file upload in a controller method?

```
A.@PostMapping("/upload")
  public String handleFileUpload(@RequestParam("file") String file) { ... }
```

```
B.@PostMapping("/upload")
  public String handleFileUpload(@RequestBody byte[] file) { ... }
```

```
C.@PostMapping("/upload")
  public String handleFileUpload(@RequestParam("file") MultipartFile file) { ... }
```

```
D.@PostMapping("/upload")
  public String handleFileUpload(HttpServletRequest request) {
      Part filePart = request.getPart("file");
      ...
  }
```

4. You're developing a Spring MVC application and need to implement a RESTful API endpoint that returns JSON data. Which of the following is the most appropriate approach?

A.
```
@Controller
public class UserController {
    @GetMapping("/api/users")
    public String getUsers(Model model) {
        model.addAttribute("users", userService.getAllUsers());
        return "userJson";
    }
}
```

B.
```
@RestController
public class UserController {
    @GetMapping("/api/users")
    public List<User> getUsers() {
        return userService.getAllUsers();
    }
}
```

C.
```
@Controller
public class UserController {
    @GetMapping("/api/users")
    @ResponseBody
    public List<User> getUsers() {
        return userService.getAllUsers();
    }
}
```

D.
```
@Controller
public class UserController {
    @GetMapping("/api/users")
    public ModelAndView getUsers() {
        return new ModelAndView("jsonView", "users",
userService.getAllUsers());
    }
}
```

5. In a Spring MVC application, you need to implement form validation. Which of the following approaches is the most appropriate?

A.public String submitForm(@Valid @ModelAttribute("user") User user, BindingResult result) { ... }

B.public String submitForm(@Validated User user, Model model) { ... }

C.
```
public String submitForm(@ModelAttribute("user") User user) {
    if (!user.isValid()) { ... }
}
```

D.
```
public String submitForm(HttpServletRequest request) {
    User user = new User(request.getParameter("name"));
    if (!user.isValid()) { ... }
}
```

6. You're debugging a Spring MVC application where a form submission is failing. The controller method is as follows:

```
@PostMapping("/saveUser")
public String saveUser(@ModelAttribute("user") User user) {
   userService.save(user);
   return "redirect:/userList";
}
```

The form submission results in a 400 Bad Request error. What could be the cause?

A.The @ModelAttribute annotation is incorrect; it should be @RequestBody.

B.The method should return a ResponseEntity instead of a String.

C.The form is submitting data that can't be bound to the User object properties.

D.The method is using POST instead of PUT for updating a user.

7. In a Spring MVC application, you need to implement a controller method that handles requests with query parameters. Which of the following is the correct way to access these parameters?

```
A.@GetMapping("/search")
  public String search(@PathVariable String query) { ... }

B.@GetMapping("/search")
  public String search(@RequestParam String query) { ... }

C.@GetMapping("/search")
  public String search(HttpServletRequest request) {
      String query = request.getParameter("query");
      ...
  }

D.@GetMapping("/search")
  public String search(@ModelAttribute SearchCriteria criteria) { ... }
```

8. You're working on a Spring MVC application and need to implement a controller method that returns different views based on a condition. Which of the following is the most appropriate approach?

```
A.@GetMapping("/user/{id}")
  public ModelAndView getUser(@PathVariable int id) {
      User user = userService.getUser(id);
      return user != null ? new ModelAndView("userView", "user", user) : new
  ModelAndView("error");
  }

B.@GetMapping("/user/{id}")
  public String getUser(@PathVariable int id, Model model) {
      User user = userService.getUser(id);
      if (user != null) {
          model.addAttribute("user", user);
          return "userView";
      } else {
          return "error";
      }
  }
```

C.@GetMapping("/user/{id}")

```
  public ResponseEntity<String> getUser(@PathVariable int id) {
      User user = userService.getUser(id);
      return user != null ? ResponseEntity.ok("userView") :
  ResponseEntity.notFound().build();
  }
```

D.@GetMapping("/user/{id}")

```
  public void getUser(@PathVariable int id, HttpServletResponse response) throws
  IOException {
      User user = userService.getUser(id);
      response.setContentType("text/html");
      response.getWriter().write(user != null ? "<h1>User found</h1>" : "<h1>User not
  found</h1>");
  }
```

9. You are developing a Spring MVC application and need to create a controller method that handles GET requests for '/users/{id}' where 'id' is a path variable. Which of the following code snippets correctly implements this requirement?

A.@GetMapping("/users")

```
  public String getUser(@RequestParam Long id) { ... }
```

B.@GetMapping("/users/{id}")

```
  public String getUser(@PathVariable Long id) { ... }
```

C.@PostMapping("/users/{id}")

```
  public String getUser(@PathVariable Long id) { ... }
```

D.@RequestMapping(value = "/users/{id}", method = RequestMethod.GET)

```
  public String getUser(@RequestParam Long id) { ... }
```

10. In a Spring MVC application, you need to pass a list of items from a controller to a view. Which of the following approaches is the most appropriate way to achieve this?

    A.Return the list directly from the controller method

    B.Use session attributes to store the list

    C.Add the list to the Model object and return the view name

    D.Use @ResponseBody to return the list as JSON

11. You are debugging a Spring MVC application and notice that a form submission is not updating the model correctly. The relevant controller method is as follows:

```
@PostMapping("/update")
public String updateUser(User user) {
    // Update logic here
    return "redirect:/users";
}
```

    What is the most likely cause of this issue?

    A.The method should use @RequestBody instead of directly accepting a User object

    B.The method is missing the @ModelAttribute annotation for the User parameter

    C.The return statement should be "forward:/users" instead of "redirect:/users"

    D.The method should return a ResponseEntity<User> instead of a String

12. In a Spring MVC application, you need to implement a controller method that accepts both GET and POST requests. Which of the following annotations is the most appropriate to use?

    A.@GetMapping

    B.@PostMapping

    C.@RequestMapping(method = {RequestMethod.GET, RequestMethod.POST})

D. @GetPostMapping

13. You are working on a Spring MVC application and need to implement a controller method that returns different views based on a condition. Which of the following code snippets correctly implements this requirement?

A.
```
@GetMapping("/check")
public String checkCondition(Model model) {
    if (someCondition()) {
        return new ModelAndView("view1");
    } else {
        return new ModelAndView("view2");
    }
}
```

B.
```
@GetMapping("/check")
public ModelAndView checkCondition() {
    if (someCondition()) {
        return "view1";
    } else {
        return "view2";
    }
}
```

C.
```
@GetMapping("/check")
public String checkCondition(Model model) {
    if (someCondition()) {
        return "view1";
    } else {
        return "view2";
    }
}
```

D.
```
@GetMapping("/check")
public View checkCondition(Model model) {
    if (someCondition()) {
        return new JstlView("view1");
    } else {
            return new JstlView("view2");
        }
    }
```

14. In a Spring MVC application, you need to implement form validation. Which of the following approaches is the most appropriate?

    A.Implement validation logic in the controller methods

    B.Use JavaScript to validate the form on the client-side only

    C.Use Bean Validation annotations on the model class and @Valid in the controller

    D.Create a custom Filter to intercept and validate all requests

15. You are debugging a Spring MVC application and notice that model attributes are not being passed correctly between requests. Which of the following annotations can you use on a controller method to ensure that specific model attributes are available for the next request?

    A.@ModelAttribute

    B.@SessionAttributes

    C.@RequestAttribute

    D.@PersistentAttributes