Ajax Support in Spring MVC

"Code with Passion!"

Ajax Support in Spring 3

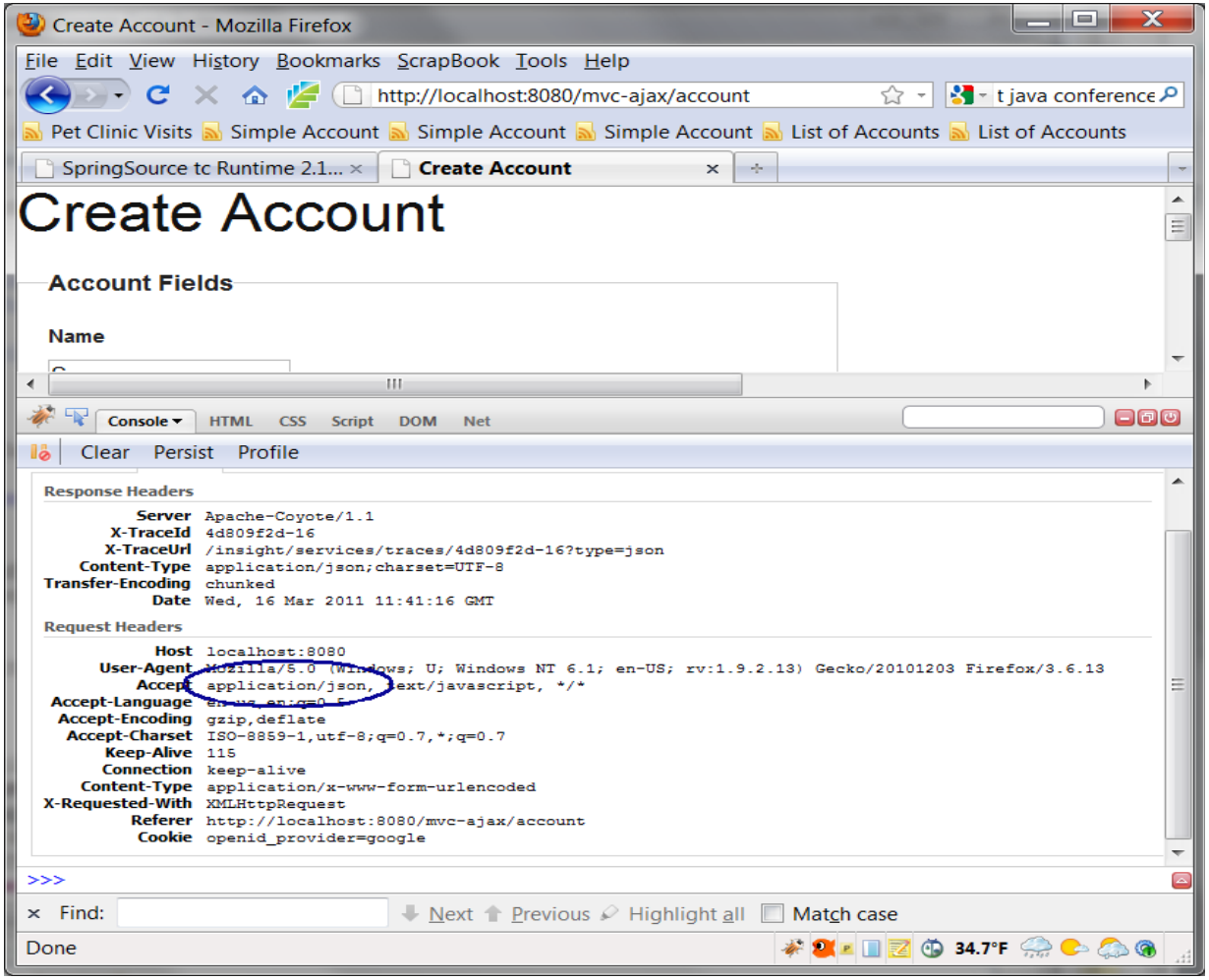
- Ajax remoting with JSON is provided as part of Spring MVC 3
  - Generating JSON responses
  - Binding JSON requests using the Spring MVC @Controller programming model.
- Use of jQuery or Dojo JavaScript libraries

Sending Ajax Request to the Server

```
$(document).ready(function() {
        // check name availability on focus lost
        $('#name').blur(function() {
                if ($('#name').val()) {
                        checkAvailability();
                }
        });
});

// Ask the server using Ajax call if the name you just entered is available.
// If it is not, an error message will be displayed and the form will
// remain disabled until you enter a name that is available.
function checkAvailability() {
    // Ask for data from the server in the JSON format
    $.getJSON("account/availability",    // URL
            { name: $('#name').val() },  // data to be sent
            function(availability) {     // callback function
                if (availability.available) {
                        fieldValidated("name", { valid : true });
                } else {
                        fieldValidated("name", { valid : false,
                 message : $('#name').val() + " is not available, try " + availability.suggestions });
                }
            }
    );
}
```

Http Request



Server-side Code that generates JSON Response

```
// Handle "/account/availability" Ajax request.
@RequestMapping(value="/availability", method=RequestMethod.GET)
// The "@ResponseBody" annotation instructs Spring MVC to serialize the
// "AvailabilityStatus" to the client. Spring MVC automatically serializes
// to JSON because that is the format client requested through .getJSON(..)
// jQuery method.
public @ResponseBody AvailabilityStatus getAvailability(@RequestParam String name) {
    for (Account a : accounts.values()) {
        if (a.getName().equals(name)) {
                return AvailabilityStatus.notAvailable(name);
        }
    }
    return AvailabilityStatus.available();
}
```

Sending JSON Ajax Request to Server

```
$(document).ready(function() {
    // check name availability on focus lost
    $('#name').blur(function() {
        if ($('#name').val()) {
                checkAvailability();
        }
    });
    // submit the form data in JSON format
    $("#account").submit(function() {
        var account = $(this).serializeObject();
        $.postJSON("account",                          // URL
                account,                                        // data
                function(data) {                       // callback function
                    // set the "#assignedId" field with the "data.id"
                    $("#assignedId").val(data.id);
                    showPopup();
        });
        return false;
    });
});
```

Server-side Code that Handles POST

```
// @RequestBody annotation instructs Spring MVC to map the body of the HTTP request
// to an Account object. Spring MVC knows to map from JSON because the client set
// the request Content Type to application/json through .getJSON(..) jQuery method.
@RequestMapping(method=RequestMethod.POST)
public @ResponseBody Map<String, ? extends Object> create(@RequestBody Account account,
HttpServletResponse response) {

        // Validates the Account object. If there are validation errors,
        // a HTTP 400 is returned with the error messages, otherwise a HTTP
        // 200 is returned with the assigned account ID.
        Set<ConstraintViolation<Account>> failures = validator.validate(account);

        // If there are validation error, send HTTP 400 along with
        // validation error messages.
        if (!failures.isEmpty()) {
                response.setStatus(HttpServletResponse.SC_BAD_REQUEST);
                return validationMessages(failures);
        }
        // If no validation error,add the account the accounts map
        // and send an account id back to the client.
        else {
                accounts.put(account.assignId(), account);
                return Collections.singletonMap("id", account.getId());
        }
}
```

Lab:

Exercise 1: Ajax operation with JSON
4953_spring3_mvc_ajax.zip

Code with Passion!