Ext JS 4 Spring MVC CRUD example - DZone Java

Ext JS 4 Spring MVC CRUD example

(0)

Comment ()

Save

Join the DZone community and get the full member experience.

Join For Free

What every Java engineer should know about microservices: Reactive Microservices Architecture. Brought to you in partnership with Lightbend.

In my last post on ExtJS 4 MVC, I have demonstrated the use ofExtJS 4 MVC to create a simple Create-Read-Update-Delete application using ExtJS only. Today we will go to see how to use that ExtJS part for UI and use Spring MVC to manage the books records on server side using spring.

Let first start by creating Spring's WebMVC configurer class. I am using spring's new xml-free approach and if you are new to it, I recommend you to please go through my previous post.

```
package org.techzoo.springmvc.bootstrap;

import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.ComponentScan;
import org.springframework.context.annotation.Configuration;
import org.springframework.web.servlet.config.annotation.EnableWebMvc;
import org.springframework.web.servlet.config.annotation.ResourceHandlerRegistry;
import org.springframework.web.servlet.config.annotation.ViewControllerRegistry;
import org.springframework.web.servlet.config.annotation.WebMvcConfigurerAdapter;
import org.springframework.web.servlet.view.InternalResourceViewResolver;

@Configuration
@EnableWebMvc
@ComponentScan(basePackages = {"org.techzoo.springmvc"})
public class MyWebMvcConfigurer extends WebMvcConfigurerAdapter {

@Override
    public void addResourceHandlers(ResourceHandlerRegistry registry) {
registry.addResourceHandler("/resources/**")
.addResourceLocations("/resources/");
    }

@Override
public void addViewControllers(ViewControllerRegistry registry) {
registry.addViewController("/")
.setViewName("index");
}

@Bean
public InternalResourceViewResolver viewResolver() {
```

```
InternalResourceViewResolver resolver =
new InternalResourceViewResolver();
resolver.setPrefix("/WEB-INF/views/");
resolver.setSuffix(".jsp");
return resolver;
}
}
```

Create a Book bean now, this bean is work as a Value Object and represent a single record in grid.

```
package org.techzoo.springmvc.vo;

/**
 * Author: Tousif Khan
 */
public class Book {

private int id;
private String title;
private String author;
private int price;
private int qty;

public Book() {}

public Book(int id, String title, String author, int price, int qty) {
super();
this.id = id;
this.title = title;
this.author = author;
this.price = price;
this.setQty(qty);
}

  //getter-setters goes here...

@Override
public String toString() {
return String.format("Book [title = %s, author = %s]",
title, author);
}
}
```

Now create a BookDao interface and implement it.

```
package org.techzoo.springmvc.dao;

import java.util.List;

import org.springframework.stereotype.Component;
import org.techzoo.springmvc.vo.Book;

@Component
public interface BookDao {

public boolean addBook(Book book);
public void updateBook(Book book);
public List<Book> listBooks();
public Book getBookById(Integer bookId);
public boolean removeBook(Book book);
}
```

To make it more simple, I am using book list but you can extend it to save it relational table. See mySpringMVC-Hibernate GRUD tutorial for more details.

```
package org.techzoo.springmvc.dao;

import java.util.ArrayList;
import java.util.List;

import org.springframework.stereotype.Repository;
import org.techzoo.springmvc.vo.Book;

@Repository
public class BookDaoImpl implements BookDao {

private static List<Book> books = new ArrayList<Book>();

static {
books.add(new Book(1001, "JDBC, Servlet and JSP", "Santosh Kumar", 300, 12000));
books.add(new Book(1002, "Head First Java", "Kathy Sierra", 550, 2500));
books.add(new Book(1003, "Java SCJP Certification", "Khalid Mughal", 650, 5500));
books.add(new Book(1004, "Spring and Hinernate", "Santosh Kumar", 350, 2500));
books.add(new Book(1005, "Mastering C++", "K. R. Venugopal", 400, 1200));
}

@Override
public boolean addBook(Book book) {
return books.add(book);
}

@Override
public boolean removeBook(Book book) {
return books.remove(book);
}

@Override
public List<Book> listBooks() {
return books;
}

@Override
public void updateBook(Book book) {
int index = books.indexOf(book);
if(index != -1) {
books.add(index, book);
}
}

@Override
public Book getBookById(Integer bookId) {
Book b = null;
for(Book b1 : books) {
if(b1.getId() == bookId) {
```

```
return b1;
}
}
return b;
}

}
```

Now it’s time to create a Book Controller.

```
package org.techzoo.springmvc.controller;

import java.util.HashMap;
import java.util.List;
import java.util.Map;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.validation.ObjectError;
import org.springframework.web.bind.MethodArgumentNotValidException;
import org.springframework.web.bind.annotation.ExceptionHandler;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.ResponseBody;
import org.techzoo.springmvc.dao.BookDao;
import org.techzoo.springmvc.vo.Book;

@Controller
@RequestMapping("api")
public class BookController {

BookDao bookBao;

@Autowired
public BookController(BookDao bookBao) {
this.bookBao = bookBao;
}

@RequestMapping(value = "/index")
public String index()
{
return "index";
}

@RequestMapping (value = "book/save", method = RequestMethod.POST)
@ResponseBody
public boolean saveBook(@RequestBody Book book)
{
return bookBao.addBook(book);
}

@ResponseBody
@RequestMapping (value = "book/loadBooks")
public Map<String, List<Book>> loadAllBooks()
{
```

```
Map<String, List<Book>> books = new HashMap<String, List<Book>>();
books.put("books", bookBao.listBooks());
return books;
}

@RequestMapping (value = "book/delete", method = RequestMethod.POST)
@ResponseBody
public boolean deleteBooks(@RequestBody Book book)
{
return bookBao.removeBook(book);
}

@RequestMapping (value = "book/updateBook", method = RequestMethod.POST)
@ResponseBody
public boolean updateBooks(@RequestBody Book book)
{
bookBao.updateBook(book);
return true;
}

}
```

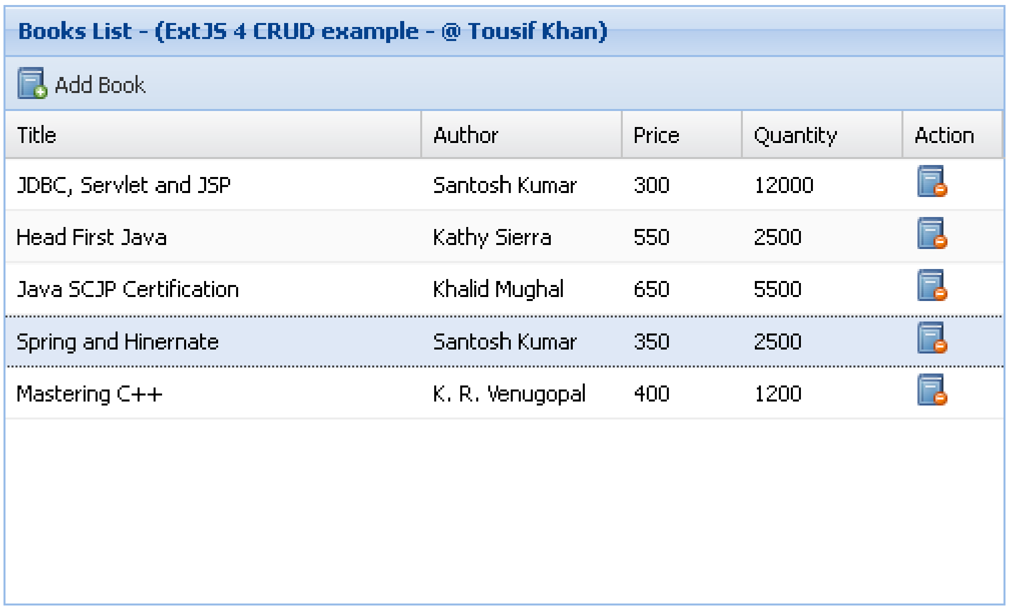
ExtJS code looks similar to previous example with a little change. I have changes the local store to server ajax proxy and after each delete/add/update operation, we will load the all record from server to make grid update.

```
Ext.onReady(function () {
  Ext.define('TechZoo.model.Book', {
    extend: 'Ext.data.Model',
    fields: [
      {name: 'id', type: 'int'},
      {name: 'title', type: 'string'},
      {name: 'author',  type: 'string'},
      {name: 'price', type: 'int'},
      {name: 'qty',  type: 'int'}
    ]
  });

  Ext.define('TechZoo.store.Books', {
    extend  : 'Ext.data.Store',
    storeId: 'bookStore',
    model   : 'TechZoo.model.Book',
    fields  : ['id', 'title', 'author','price', 'qty'],
    proxy: {
        type: 'ajax',
        url: '/SpringMVC-ExtJS-CRUD/api/book/loadBooks',
        reader: {
            type: 'json',
            root: 'books'
        }
    },
    autoLoad: true
  });

  Ext.define('TechZoo.view.BooksList', {
    extend: 'Ext.grid.Panel',
    alias: 'widget.bookslist',
    title: 'Books List - (ExtJS SpringMVC example - @ Tousif Khan)',
    store: 'Books',
    initComponent: function () {
      this.tbar = [{
        text    : 'Add Book',
        action  : 'add',
        iconCls : 'book-add'
      }];
      this.columns = [
        { header: 'Title', dataIndex: 'title', flex: 1 },
        { header: 'Author', dataIndex: 'author' },
        { header: 'Price', dataIndex: 'price' , width: 60 },
        { header: 'Quantity', dataIndex: 'qty', width: 80 },
        { header: 'Action', width: 50,
          renderer: function (v, m, r) {
            var id = Ext.id();
            Ext.defer(function () {
              Ext.widget('image', {
                renderTo: id,
                name: 'delete',
                src : 'resources/images/book_delete.png',
                listeners : {
                  afterrender: function (me) {
                    me.getEl().on('click', function() {
                      var grid = Ext.ComponentQuery.query('bookslist')[0];
                      if (grid) {
                        var sm = grid.getSelectionModel();
                        var rs = sm.getSelection();
                        if (!rs.length) {
                          Ext.Msg.alert('Info', 'No Book Selected');
                          return;
                        }
                        Ext.Msg.confirm('Remove Book',
                          'Are you sure you want to delete?',
                          function (button) {
                            if (button == 'yes') {
                              //grid.store.remove(rs[0]);
                            var book = rs[0].getData();
                            Ext.Ajax.request({
                                url: '/SpringMVC-ExtJS-CRUD/api/book/delete',
                                method  : 'POST',
                                jsonData: book,
                                success: function(response){
                                var grid = Ext.ComponentQuery.query('bookslist')[0];
                                    grid.getStore().load();
                                }
                            });
                            }
                        });
                      }
                    });
                  }
                }
              });
            }, 50);
          return Ext.String.format('<div id="{0}"></div>', id);
        }
      }
    ];
    this.callParent(arguments);
  }
});

  Ext.define('TechZoo.view.BooksForm', {
    extend  : 'Ext.window.Window',
    alias   : 'widget.booksform',
    title   : 'Add Book',
    width   : 350,
    layout  : 'fit',
    resizable: false,
    closeAction: 'hide',
    modal   : true,
    config  : {
      recordIndex : 0,
      action : ''
    },
    items   : [{
      xtype : 'form',
      layout: 'anchor',
      bodyStyle: {
        background: 'none',
        padding: '10px',
        border: '0'
      },
      defaults: {
        xtype : 'textfield',
        anchor: '100%'
      },
      items : [{
        name  : 'title',
        fieldLabel: 'Book Title'
      },{
        name: 'author',
        fieldLabel: 'Author Name'
      },{
        name: 'price',
        fieldLabel: 'Price'
      },{
        name: 'qty',
        fieldLabel: 'Quantity'
      }]
    }],
    buttons: [{
      text: 'OK',
      action: 'add'
    },{
      text    : 'Reset',
      handler : function () {
        this.up('window').down('form').getForm().reset();
      }
    },{
      text   : 'Cancel',
      handler: function () {
        this.up('window').close();
      }
    }]
  });

Ext.define('TechZoo.controller.Books', {
  extend  : 'Ext.app.Controller',
  stores  : ['Books'],
  views   : ['BooksList', 'BooksForm'],
  refs    : [{
    ref   : 'formWindow',
    xtype : 'booksform',
    selector: 'booksform',
    autoCreate: true
  }],
  init: function () {
    this.control({
      'bookslist > toolbar > button[action=add]': {
        click: this.showAddForm
      },
      'bookslist': {
        itemdblclick: this.onRowdblclick
      },
      'booksform button[action=add]': {
        click: this.doAddBook
      }
    });
  },
  onRowdblclick: function(me, record, item, index) {
    var win = this.getFormWindow();
    win.setTitle('Edit Book');
    win.setAction('edit');
    win.setRecordIndex(index);
    win.down('form').getForm().setValues(record.getData());
    win.show();
  },
  showAddForm: function () {
    var win = this.getFormWindow();
    win.setTitle('Add Book');
    win.setAction('add');
    win.down('form').getForm().reset();
    win.show();
  },
  doAddBook: function () {
    var win = this.getFormWindow();
    var store = this.getBooksStore();
    var values = win.down('form').getValues();

    var action = win.getAction();
 //   var book = Ext.create('TechZoo.model.Book', values);
    var url = '';
    if(action == 'edit') {
```

```
      url = '/SpringMVC-ExtJS-CRUD/api/book/updateBook';
      }
      else {
      url = '/SpringMVC-ExtJS-CRUD/api/book/save';
      }
      Ext.Ajax.request({
  url: url,
    method  : 'POST',
    jsonData: values,
    success: function(response){
        store.load();
    }
    });
      win.close();
    }
  });

  Ext.application({
    name  : 'TechZoo',
    controllers: ['Books'],
      launch: function () {
        Ext.widget('bookslist', {
          width : 500,
          height: 300,
          renderTo: 'output'
        });
      }
    }
  );
});
```

Output:

Open the HTML file in any browser, the Output will look similar to below. You can click to add new Book record in GRID.



Books List - (ExtJS 4 CRUD example - @ Tousif Khan)

Add Book

Title	Author	Price	Quantity	Action
JDBC, Servlet and JSP	Santosh Kumar	300	12000	
Head First Java	Kathy Sierra	550	2500	
Java SCJP Certification	Khalid Mughal	650	5500	
Spring and Hinernate	Santosh Kumar	350	2500	
Mastering C++	K. R. Venugopal	400	1200	