outputAction10.0.3

1.entity

```
package com.fr.schedule.base.entity.output;

import com.fr.schedule.base.bean.output.OutputFtp;
import com.fr.schedule.base.entity.AbstractScheduleEntity;
import com.fr.third.javax.persistence.Column;
import com.fr.third.javax.persistence.Entity;
import com.fr.third.javax.persistence.Table;


/**
 * Created by Zed on 2017/12/21.
 * ftp
 */
@Entity
@Table(name = "fine_output_ftp") //
@TableAssociation(associated = true)
public class OutputFtpEntity extends AbstractScheduleEntity {

    public static final String COLUMN_SERVER_ADDRESS = "serverAddress";
    public static final String COLUMN_PORT = "port";
    public static final String COLUMN_SAVE_PATH = "savePath";
    public static final String COLUMN_USERNAME = "username";
    public static final String COLUMN_PASSWORD = "password";

    @Column(name = COLUMN_SERVER_ADDRESS)
    private String serverAddress = null;

    @Column(name = COLUMN_PORT)
    private String port = null;

    @Column(name = COLUMN_SAVE_PATH)
    private String savePath = null;

    @Column(name = COLUMN_USERNAME)
    private String username = null;

    @Column(name = COLUMN_PASSWORD)
    private String password = null;

    public OutputFtpEntity() {

    }

        //bean
    @Override
    public OutputFtp createBean() {
        return new OutputFtp()
                .id(this.getId())
                .serverAddress(this.getServerAddress())
                .port(this.getPort())
                .savePath(this.getSavePath())
                .username(this.getUsername())
                .password(this.getPassword());
    }

    public OutputFtpEntity id(String id) {
        setId(id);
        return this;
    }

    public String getServerAddress() {
        return serverAddress;
    }

    public void setServerAddress(String serverAddress) {
```

```
        this.serverAddress = serverAddress;
    }

    public OutputFtpEntity serverAddress(String serverAddress) {
        setServerAddress(serverAddress);
        return this;
    }

    public String getPort() {
        return port;
    }

    public void setPort(String port) {
        this.port = port;
    }

    public OutputFtpEntity port(String port) {
        setPort(port);
        return this;
    }

    public String getSavePath() {
        return savePath;
    }

    public void setSavePath(String savePath) {
        this.savePath = savePath;
    }

    public OutputFtpEntity savePath(String savePath) {
        setSavePath(savePath);
        return this;
    }

    public String getUsername() {
        return username;
    }

    public void setUsername(String username) {
        this.username = username;
    }

    public OutputFtpEntity username(String username) {
        setUsername(username);
        return this;
    }

    public String getPassword() {
        return password;
    }

    public void setPassword(String password) {
        this.password = password;
    }

    public OutputFtpEntity password(String password) {
        setPassword(password);
        return this;
    }


}
```

2.bean

```
```

```
package com.fr.schedule.base.bean.output;

import com.fr.schedule.base.entity.AbstractScheduleEntity;
import com.fr.schedule.base.entity.output.OutputFtpEntity;
import com.fr.schedule.base.type.RunType;
import com.fr.third.fasterxml.jackson.annotation.JsonSubTypes;

/**
 * Created by Zed on 2017/12/19.
 * ,ftp
 */
@JsonSubTypes.Type(value = OutputFtp.class, name = "OutputFtp")
public class OutputFtp extends BaseOutputAction {

    private static final long serialVersionUID = 8245931480823179622L;
    private String serverAddress = null;
    private String port = null;
    private String savePath = null;
    private String username = null;
    private String password = null;

    public OutputFtp() {
        super();
    }

        //
    @Override
    public boolean willExecuteByUser() {
        return false;
    }

    @Override
    public RunType runType() {
        return RunType.SEND_FTP;
    }

        //entity
    @Override
    public Class<? extends AbstractScheduleEntity> outputActionEntityClass() {
        return OutputFtpEntity.class;
    }

        //entity
    @Override
    public OutputFtpEntity createOutputActionEntity() {
        return new OutputFtpEntity()
                .id(this.getId())
                .password(this.getPassword())
                .serverAddress(this.getServerAddress())
                .port(this.getPort())
                .savePath(this.getSavePath())
                .username(this.getUsername());
    }

    public OutputFtp id(String id) {
        setId(id);
        return this;
    }

    public String getServerAddress() {
        return serverAddress;
    }

    public void setServerAddress(String serverAddress) {
        this.serverAddress = serverAddress;
    }

    public OutputFtp serverAddress(String serverAddress) {
        setServerAddress(serverAddress);
        return this;
    }
```

```
    public String getPort() {
        return port;
    }

    public void setPort(String port) {
        this.port = port;
    }

    public OutputFtp port(String port) {
        setPort(port);
        return this;
    }

    public String getSavePath() {
        return savePath;
    }

    public void setSavePath(String savePath) {
        this.savePath = savePath;
    }

    public OutputFtp savePath(String savePath) {
        setSavePath(savePath);
        return this;
    }

    public String getUsername() {
        return username;
    }

    public void setUsername(String username) {
        this.username = username;
    }

    public OutputFtp username(String username) {
        setUsername(username);
        return this;
    }

    public String getPassword() {
        return password;
    }

    public void setPassword(String password) {
        this.password = password;
    }

    public OutputFtp password(String password) {
        setPassword(password);
        return this;
    }
}
```

3.handler

```
package com.fr.schedule.feature.output;

import com.fr.ftp.client.FTPClient;
import com.fr.ftp.client.FTPReply;
import com.fr.io.utils.ResourceIOUtils;
import com.fr.schedule.base.bean.output.OutputFtp;
import com.fr.schedule.base.constant.ScheduleConstants;
import com.fr.stable.ArrayUtils;
import com.fr.stable.StringUtils;
```

```
import java.io.InputStream;
import java.util.Map;

/**
 * Created by Zed on 2018/01/10.
 */
public class FTPHandler extends OutputActionHandler<OutputFtp> {

    private static final int CONNECTION_TIME_OUT = 10000;
    private static final int DATA_TIME_OUT = 60000;
    // FTPiso-8859-1
    private static final String SERVER_CHARSET = "ISO-8859-1";
    //
    private static final String LOCAL_CHARSET = "GBK";
    //ftputf-8
    private static final String UTF8_CHARSET = "UTF-8";


    @Override
    public void doAction(OutputFtp ftp, Map<String, Object> map) throws Exception {

        FTPClient client = null;
        try {
            String[] files = (String[]) map.get(ScheduleConstants.OUTPUT_FILES);

            client = createFtpClient(ftp);

            if (FTPReply.isPositiveCompletion(client.getReplyCode())) {
                //
                changeFtpDir(client, ftp.getSavePath());
                for (String file : files) {
                    uploadFile(client, file);
                }
            }

        } catch (Exception e) {
            throw e;
        } finally {
            if (client != null) {
                client.disconnect();
            }
        }

    }

    public boolean testFtp(OutputFtp ftp) throws Exception {
        FTPClient client = null;
        try {
            client = createFtpClient(ftp);
            changeFtpDir(client, ftp.getSavePath());
            return FTPReply.isPositiveCompletion(client.getReplyCode());
        } catch (Exception e) {
            throw e;
        } finally {
            if (client != null) {
                client.disconnect();
            }
        }
    }

    private FTPClient createFtpClient(OutputFtp ftp) throws Exception {

        FTPClient client = new FTPClient();
        client.setDataTimeout(DATA_TIME_OUT);
        client.setConnectTimeout(CONNECTION_TIME_OUT);
        client.connect(ftp.getServerAddress(), Integer.valueOf(ftp.getPort()));
        client.login(ftp.getUsername(), ftp.getPassword());
        client.setFileType(FTPClient.BINARY_FILE_TYPE);

        if (FTPReply.isPositiveCompletion(client.sendCommand("OPTS UTF8", "ON"))) {
```

```
            client.setControlEncoding(UTF8_CHARSET);
        } else {
            client.setControlEncoding(LOCAL_CHARSET);
        }

        return client;
    }

    private void uploadFile(FTPClient client, String file) throws Exception {
        InputStream in = ResourceIOUtils.read(file);
        try {
            if (!client.storeFile(encodeFtpName(ResourceIOUtils.getName(file), client.getControlEncoding()),
in)) {
                throw new Exception("upload files failed");
            }

        } catch (Exception e) {
            throw e;
        } finally {
            in.close();
        }
    }

    private void changeFtpDir(FTPClient client, String savePath) throws Exception {
        String[] parent = savePath.split("/");
        if (ArrayUtils.isNotEmpty(parent)) {
            for (String son : parent) {
                if (StringUtils.isEmpty(son)) {
                    continue;
                }
                if (!client.changeWorkingDirectory(encodeFtpName(son, client.getControlEncoding()))) {
                    throw new Exception(":Non-existent such path: " + savePath);
                }
            }
        }
    }

    private String encodeFtpName(String s, String charset) throws Exception {
        return new String(s.getBytes(charset), SERVER_CHARSET);
    }
}
```

4.handlerentity

```
OutputActionHandler.registerHandler(new FTPHandler(), OutputFtp.class.getName());
ScheduleOutputActionEntityRegister.getInstance().addClass(OutputFtpEntity.class);
```