M2soft
more than the most

ERS Reporting Server

개발 가이드 Ver. 7.4.1

㈜엠투소프트

M2soft
more than the most

CONTENTS

1. 매뉴얼 소개

1.1. 매뉴얼의 목적

본 매뉴얼은 ERS Reporting Server (이하 Reporting Server) 를 이용하여 Server-Side Reporting 을 하는 방법을 기술합니다.
Reporting Server 의 설치 방법은 ERS Reporting Server Installation Manual 를 참조하시기 바랍니다.

1.2. 매뉴얼의 전제 조건

Reporting Server 는 HTTP 프로토콜로 송수신 하며 본 매뉴얼의 독자는 HTTP 프로토콜에 대한 이해하고 있다는 가정 하에 매뉴얼을 기술합니다.
본 매뉴얼의 예제 코드는 Java를 이용한 코드 입니다. 본 매뉴얼의 독자는 Java 코드를 이해할 수 있다는 가정 하에 매뉴얼을 기술합니다.

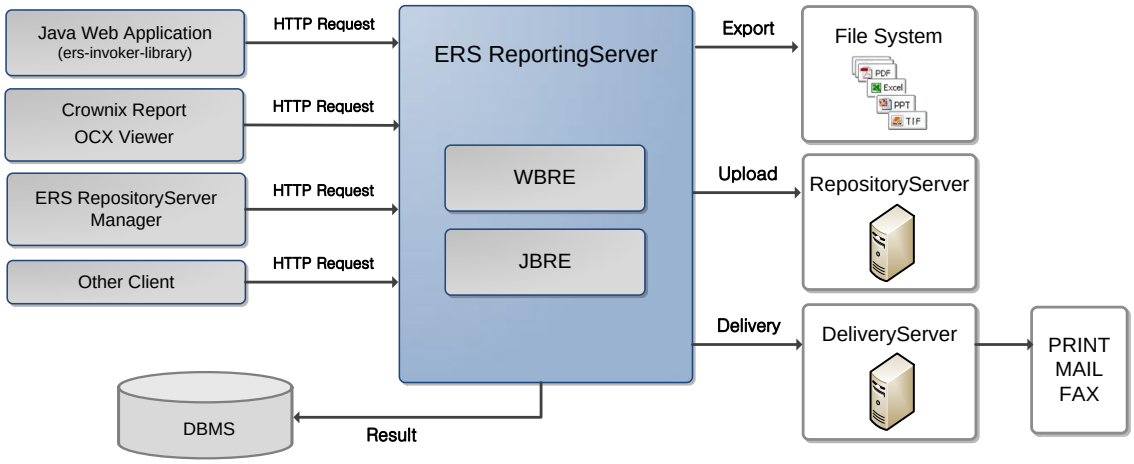
1.3. 약어

ReportingServer_HOME	Reporting Server 가 설치된 디렉터리를 의미합니다.
이탤릭체로 표기된 중괄호	Reporting Server의 환경 설정 파일(server.properties)의 설정값을 의미합니다. 예를 들어 *{report.save.dir}* 는 server.properties의 report.save.dir 의 값을 의미합니다.

2. ERS Reporting Server 아키텍처

2.1. 아키텍처

Reporting Server 는 HTTP Request 를 받아서 서버 내부에서 보고서를 생성하여 저장 합니다. 보고서 생성에 필요한 정보는 HTTP 프로토콜의 POST 파라미터로 전달합니다.
생성된 보고서는 Repository Server 로 업로드 하거나 Delivery Server 로 Print, Mail, Fax 요청을 할 수 있으며 처리 결과를 Database에 기록할 수 있습니다.



WBRE (Win32 based Reporting Engine)	ERS Reporting Server에 내장된 OCX 기반의 뷰어입니다. 문서를 리포팅하는 컴포넌트 입니다.
JBRE (Java based Reporting Engine)	ERS Reporting Server에 내장된 Java 기반의 뷰어입니다. 문서를 리포팅하는 컴포넌트 입니다.
Java Web Application	Java 기반의 Web Application 에서 Reporting Server와 함께 제공되는 ers-invoker 라이브러리를 이용하여 리포팅을 요청할 수 있습니다.
Other Client	HTTP 프로토콜을 지원하는 모든 클라이언트에서 Reporting Server 로 리포팅을 요청할 수 있습니다.
DBMS	리포팅에 대한 처리 결과를 Database에 기록할 수 있습니다.

Repository Server 의 자세한 내용은 *ERS Repository Server Installation Manual* 또는 *ERS Repository Server Developer Guide* 를 참고해 주시기 바랍니다.
Delivery Server 의 자세한 내용은 *ERS Delivery Server Installation Manual* 또는 *ERS Delivery Server Developer Guide* 를 참고해 주시기 바랍니다.

3. Server-Side Export

Server-Side Export 는 서버 내부에서 보고서를 실행하고 저장하는 기능입니다. 보고서 실행과 저장에 필요한 파라미터는 HTTP POST 파라미터로 전달합니다.

3.1. Server-Side Export 의 POST 파라미터

3.1.1. 파라미터 상세

키	설명	구분	비고
opcode	명령 처리 코드, Server-Side Export는 500 입니다. 500 이 아닌 다른 값일 경우 Server-Side Export 가 되지 않습니다.	필수	
mrd_path	mrd 파일의 경로 입니다. 경로는 URL 경로 또는 상대 경로를 사용합니다. 상대경로는 서버의 *{report.mrd.dir}* 경로를 기준으로 한 상대 경로 입니다.	필수	
mml_path	mml 파일의 경로 입니다. 경로는 URL 경로 또는 상대 경로를 사용합니다. 상대경로는 서버의 *{report.mrd.dir}* 경로를 기준으로 한 상대 경로 입니다. 이 키는 mrd_path가 적용된 경우에는 무효화 되므로 함께 사용할 수 없습니다.		
mml_data	mml 전문입니다. 이 키는 mrd_path나 mml_path가 적용된 경우에는 무효화 되므로 함께 사용할 수 없습니다		
mrd_param	mrd_path, mml_path, mml_data에 설정된 파일 또는 전문을 실행하는 파라미터 입니다. RDViewr.FileOpen() 의 두 번째 파라미터와 동일합니다.	필수	
export_type	export 할 파일의 타입입니다. 아래와 같은 포멧을 지원합니다. mrr, pdf, doc, xls, csv, ppt, hwp, png, jpg ※ 제약사항 - 저장 파라미터(예, /rcsv, /rexport등)와 동시 사용불가.	필수	
export_name	export 할 파일의 파일명 입니다. 파일명은 확장자를 포함합니다.	옵션	기본값: 서버에서 중복되지 않는 임의의 값으로 생성
protocol	export 결과를 응답하는 형식입니다. Reporting Server	옵션	기본값:

	의 응답방식은 async, sync, file 방식을 지원합니다		*{server.protocol}*
delete_file	protocol의 값이 file로 설정된 경우, 클라이언트로 파일을 내려주고나서, 서버에 만든 파일을 삭제할지 결정하기 위한 설정입니다. True로 설정하면 클라이언트로 파일을 내려준 후, 서버에 만든 파일을 삭제합니다. False로 설정하면 클라이언트로 파일을 내려준 후 서버에 만든 파일을 삭제하지 않고 그대로 둡니다. 추가된 버전 : 6.3.0.142	옵션	기본값 : true
mrd_data	보고서 실행에 필요한 데이터를 직접 전송할 때 사용합니다. RDViewer.SetRdata() 또는 /rdata 파라미터와 동일합니다.	옵션	

3.1.2. 파라미터 예제

키	값	설명
opcode	500	Server-Side Export 명령 코드
mrd_path	sample.mrd	*{report.mrd.dir}* 경로의 sample.mrd 를 이용하여 보고서를 실행함.
mml_path	sample.mml ENC*sample.mml	*{report.mrd.dir}* 경로의 sample.mml 를 이용하여 보고서를 실행합니다. 해당 mml이 암호화 된 경우 ENC* 을 앞에 붙인 경로로 사용합니다. mml이 암호화되는 경우 - InfoTalk Server의 storage 서블릿으로 mml을 저장할 때 암호화를 사용하는 경우.
mml_data	<?xml version="1.0" encoding="utf-8"?><MML ... </MML>	mml 전문으로 보고서를 실행합니다.
mrd_param	/rfn [http://192.168.10.10/ReportingServer/mrd/sample.txt]	보고서 실행에 필요한 데이터는 http://192.168.10.10/ReportingServer/mrd/sample.txt 를 사용함.
export_type	doc	doc 로 저장
export_name	sample_20110601.doc	파일명은 sample_20110601.doc 로 저장
protocol	sync	응답 방식을 sync 로 설정

delete_file	false	클라이언트로 파일을 내려주고, 서버에 만든 파일을 삭제하지 않고 그대로 둡니다.

3.2. Server-Side Export 의 Response

3.2.1. async 방식

요청을 정상적으로 Reporting Server가 수신했다면 바로 응답합니다. Server-Side Export는 응답 후 서버에서 순차 처리 됩니다.
이러한 방식은 Web Application Server의 Servlet Thread 수와 커넥션 수를 감소시키므로 시스템 자원을 절약할 수 있는 장점이 있습니다. Reporting Server의 기본 설정은 async 입니다.

- 형식

<결과코드>|<메시지>

- 결과코드

	결과코드	메시지	**응답**
수신 성공	1	임의의 값	**1|20121127141856439**
수신 실패	0	오류 메시지	**0|[ERS-11001]요청 정보를 파싱하는데 실패 했습니다.**

※ 임의의 값 – 서버에서 생성되는 중복되지 않는 임의의 값.

3.2.2. sync 방식

요청을 수신하고 Server-Side Export 완료 후에 처리 결과를 응답합니다. Server-Side Export 의 처리 결과를 반드시 응답 받아야 하는 경우에 사용할 수 있습니다.
이러한 방식은 처리 결과를 응답 받을 수 있는 장점이 있으나 Server-Side Export의 처리 시간 만큼 Web Application Server의 Servlet Thread 가 유지되고 커넥션이 증가한다는 단점이 있습니다.

- 형식

<결과코드>|<메시지>

- 결과코드

	결과코드	메시지	**응답**
수신 성공	1	결과 파일의 상대경로	**1|2011/06/01/DailyReport_12566.pdf**

수신 실패	0	오류 메시지	**0|[ERS- 12101] 보고서 실행 중에 오류가 발생했습니다.(java.io.FileNotFoundException)**

3.2.3. file 방식

요청을 수신하고 Server-Side Export 완료 후에 결과 파일을 응답합니다. 결과 파일은 확장자에 맞게 Content-Type 설정되어 요청을 보낸 클라이언트로 다운로드 됩니다.

- 응답

	HTTP Status	**응답**
성공	200	***결과 파일의 바이트***
실패	500	

- Content-Type

확장자	**Contect-Type**
mrr	application/octet-stream
pdf	application/pdf
doc	application/msword
xls	application/vnd.ms-excel
ppt	application/vnd.ms-powerpoint
hwp	application/haansofthwp
jpg, jpeg	image/jpeg
png	image/png
tif, tiff	image/tiff

※ 다운로드 된 파일이 "열기"가 되지 않을 경우에는 아래 URL을 참고하십시오.
- http://support.microsoft.com/kb/982995 : Internet Explorer 7 또는 Internet Explorer 8 Office 2010 문서를 보려고 할 때 새 응용 프로그램 창이 열립니다.

3.3. Server-Side Export 의 결과 파일

3.3.1. 저장 경로 설정

async 또는 sync 방식을 생성된 결과 파일은 *{report.save.dir}* 를 기준으로 하여 *{report.dir.pattern}* 의 날자 형식에 맞게 디렉터리를 생성하여 저장됩니다.

```
report.save.dir= /app/ReportingServer/report
```

report.dir.pattern=yyyy/MM/dd/

예를 들어 위와 같이 서버 설정이 되어 있고 현재 날짜가 2011/06/01 일 때 저장되는 경로는 /app/ReportingServer/report/2011/06/11 이 됩니다.

저장 디렉터리의 날짜 패턴은 *{report.dir.pattern}* 를 수정하여 변경할 수 있습니다.

{report.dir.pattern} 값	저장 디렉터리
yyyy/MM/dd/	일 단위로 디렉터리를 생성하여 저장
yyyy/MM/dd/HH/	시간 단위로 디렉터리를 생성하여 저장
yyyy/MM/dd/HH/mm/	분 단위로 디렉터리를 생성하여 저장

3.3.2. 저장 파일 확인

Server-Side Export 에 성공하였다면 저장된 디렉터리에 결과 파일이 저장 된 것을 확인 할 수 있습니다.

```
/app/ReportingServer/report/2011/06/01>ls
sample_20110601.doc
```

4. Export To Repository Server

Export To Repository Server 는 서버 내부에서 보고서를 실행하여 저장한 보고서를 Repository Server 에 업로드하는 기능입니다. 보고서 실행과 저장/업로드에 필요한 파라미터는 HTTP POST 파라미터로 전달합니다.

4.1. Export To Repository Server 의 POST 파라미터

4.1.1. 파라미터 상세

키	설명	구분	비고
opcode	명령 처리 코드, Repository Server Export는 501 입니다. 501 이 아닌 다른 값일 경우 Repository Server Export 가 되지 않습니다.	필수	
mrd_path	mrd 파일의 경로 입니다. 경로는 URL 경로 또는 상대 경로를 사용합니다. 상대경로는 서버의 *{report.mrd.dir}* 경로를 기준으로 한 상대 경로 입니다.	필수	
mml_path	mml 파일의 경로 입니다. 경로는 URL 경로 또는 상대 경로를 사용합니다. 상대경로는 서버의 *{report.mrd.dir}* 경로를 기준으로 한 상대 경로 입니다. 이 키는 mrd_path가 적용된 경우에는 무효화 되므로 함께 사용할 수 없습니다.		
mml_data	mml 전문입니다. 이 키는 mrd_path나 mml_path가 적용된 경우에는 무효화 되므로 함께 사용할 수 없습니다		
mrd_param	mrd_path, mml_path, mml_data에 설정된 파일 또는 전문을 실행하는 파라미터 입니다. RDViewr.FileOpen() 의 두 번째 파라미터와 동일합니다.	필수	
export_type	export 할 파일의 타입입니다. 아래와 같은 포멧을 지원합니다. mrr, pdf, doc, xls, csv, ppt, hwp, png, jpg. ※ 제약사항 - 저장 파라미터(예, /rcsv, /rexport등)와 동시 사용불가.	옵션	기본값: mrr
export_name	export 할 파일의 파일명 입니다. 파일명은 확장자를 포함합니다.	옵션	기본값: 서버에서 중복되지 않는 임의의 값으로 생성

protocol	export 결과를 응답하는 형식입니다. Reporting Server의 응답방식은 async, sync 방식을 지원합니다	옵션	기본값: *{server.protocol}*
delete_file	protocol의 값이 file로 설정된 경우, 클라이언트로 파일을 내려주고나서, 서버에 만든 파일을 삭제할지 결정하기 위한 설정입니다. True로 설정하면 클라이언트로 파일을 내려준 후, 서버에 만든 파일을 삭제합니다. False로 설정하면 클라이언트로 파일을 내려준 후 서버에 만든 파일을 삭제하지 않고 그대로 둡니다. 추가된 버전 : 6.3.0.142	옵션	기본값 : true
mrd_data	보고서 실행에 필요한 데이터를 직접 전송할 때 사용합니다. RDViewer.SetRdata() 또는 /rdata 파라미터와 동일합니다.	옵션	
user_id	Repository Server의 사용자 계정입니다.	필수	
user_pwd	Repository Server의 사용자 계정 패스워드 입니다.	필수	
category_id	문서를 저장할 Repository Server의 카테고리 ID 입니다.	필수	
file_title	Repository Server에 업로드 할 문서의 제목입니다.	옵션	
file_description	사용자가 입력한 문서의 수정 내역입니다.	옵션	
service_url	업로드 할 Repository Server의 service 주소 입니다.	필수	

4.1.1.1. HTML4 변환 파라미터

RDON에서 config.ini에 설정하여 사용하는 html_option을 ReportingServer에서도 사용가능하게 파라미터입니다.

· ERS ReportingServer 6.3.4.189부터 지원
· Java Engine 5.5.8.235부터 지원
· Win32 Engine은 다음 버전에서 지원
- RD50, RD50u, RD50r, RD55j, RD60, RD60u, RD60j, RD70, RD70u, RD70j
- server.properties의 agent.ocx.version에 지정

키	설명	구분	비고
html_left	Crownix Viewer의 SetHtmlPageInfo 메소드의 첫번째 인자 값	옵션	기본값 0
Html_top	Crownix Viewer의 SetHtmlPageInfo 메소드의 두번째 인자	옵션	기본값 0

	값		
html_width	Crownix Viewer의 SetHtmlPageInfo 메소드의 세번째 인자값	옵션	기본값 -1
html_height	Crownix Viewer의 SetHtmlPageInfo 메소드의 네번째 인자값	옵션	기본값 -1
htmlStyleMode	Crownix Viewer의 SetHtmlStyleMode 메소드 true/false로 설정	옵션	기본값 false
htmlExportMode	Crownix Viewer의 SetHtmlExportMode 메소드 true/false로 설정	옵션	기본값 false
htmlCellMergeGapsHorz	Crownix Viewer의 SetHtmlCellMergeGaps 메소드의 첫번째 인자값	옵션	기본값 -1
htmlCellMergeGapsVert	Crownix Viewer의 SetHtmlCellMergeGaps 메소드의 두번째 인자값	옵션	기본값 -1
htmlAlignMode	Crownix Viewer의 SetHtmlAlignMode 메소드 true/false로 설정	옵션	기본값 false

4.1.2. 파라미터 예제

키	값	설명
opcode	501	Server-Side Export 명령 코드
mrd_path	sample.mrd	*{report.mrd.dir}* 경로의 sample.mrd 를 이용하여 보고서를 실행함.
mml_path	sample.mml	*{report.mrd.dir}* 경로의 sample.mml 를 이용하여 보고서를 실행함.
mml_data	<?xml version="1.0" encoding="utf-8"?><MML ... </MML>	mml 전문으로 보고서를 실행함
mrd_param	/rfn [http://192.168.10.10/ReportingServer/mrd/sample.txt]	보고서 실행에 필요한 데이터는 http://192.168.10.10/ReportingServer/mrd/sample.txt 를 사용함.
export_type	doc	doc 로 저장
export_name	sample_20110601.doc	파일명은 sample_20110601.doc 로 저장
protocol	sync	응답 방식을 sync 로 설정
delete_file	false	클라이언트로 파일을 내려주고, 서버에 만든 파일을 삭제하지 않고 그대로 둡니다.
user_id	admin	Repository Server의 사용자 계정

user_pwd	1234	Repository Server의 사용자 계정 패스워드
category_id	11	문서를 저장할 Repository Server의 카테고리 ID
file_title	영업보고서	Repository Server에 업로드 할 문서의 제목
file_description	AA_1.MRR	사용자가 입력한 문서의 수정 내역
service_url	http://192.168.10.10/ReportingServer/service	업로드 할 Repository Server의 service 주소

4.2. Export To Repository Server 의 Response

Reporting Server 의 응답방식은 async, sync 방식을 지원합니다.

4.2.1. async 방식

요청을 정상적으로 Reporting Server가 수신했다면 바로 응답합니다. Export To Repository Server 는 응답 후 서버에서 순차 처리 됩니다.
이러한 방식은 Web Application Server의 Servlet Thread 수와 커넥션 수를 감소시키므로 시스템 자원 절약할 수 있는 장점이 있습니다. Reporting Server의 기본 설정은 async 입니다.

- 형식

<결과코드>|<메시지>

- 결과코드

구분	결과코드	메시지	응답
수신 성공	1	없음	**1|**
수신 실패	0	오류 메시지	**0|[ERS-11001]요청 정보를 파 싱하는데 실패 했습니다.**

4.2.2. sync 방식

요청을 수신하고 Export To Repository Server 완료 후에 처리 결과를 응답합니다. Export To Repository Server 의 처리 결과를 반드시 응답 받아야 하는 경우에 사용할 수 있습니다.
이러한 방식은 처리 결과를 응답 받을 수 있는 장점이 있으나 Export To Repository Server 의 처리 시간 만큼 Web Application Server의 Servlet Thread 가 유지되고 커넥션이 증가한다는 단점이 있습니다.

- 형식

<결과코드>|<메시지>

- 결과코드

구분	결과코드	메시지	응답
성공	1	없음	**1|**
실패	0	오류 메시지	**0|[ERS- 12101] 보고서 실행 중에 오류가 발생했습니다.(java.io.FileNotFoundException)**

4.3. Export To Repository Server 의 결과 파일

4.3.1. 저장 경로 설정

async 또는 sync 방식을 생성된 결과 파일은 *{report.save.dir}* 를 기준으로 하여 *{report.dir.pattern}* 의 날자 형식에 맞게 디렉터리를 생성하여 저장됩니다.

```
report.save.dir= /app/ReportingServer/report
report.dir.pattern=yyyy/MM/dd/
```

예를 들어 위와 같이 서버 설정이 되어 있고 현재 날짜가 2011/06/01 일 때 저장되는 경로는 /app/ReportingServer/report/2011/06/11 이 됩니다.

저장 디렉터리의 날짜 패턴은 *{report.dir.pattern}* 를 수정하여 변경할 수 있습니다.

{report.dir.pattern} 값	저장 디렉터리
yyyy/MM/dd/	일 단위로 디렉터리를 생성하여 저장
yyyy/MM/dd/HH/	시간 단위로 디렉터리를 생성하여 저장
yyyy/MM/dd/HH/mm/	분 단위로 디렉터리를 생성하여 저장

4.3.2. 저장 파일 확인

Export To Repository Server 에 성공하였다면 저장된 디렉터리에 결과 파일이 저장 된 것을 확인 할 수 있습니다.

```
/app/ReportingServer/report/2011/06/01>ls
sample_20110601.doc
```

저장되어진 파일을 이용하여 Repository Server 로 업로드 요청을 하게 됩니다.

5. Server-Side MultiSend

Server-Side MultiSend 는 서버 내부에서 보고서를 실행하여 저장한 보고서를 Repository Server 에 업로드 또는 Delivery Server 로 Print, Mail, Fax 을 요청하는 기능입니다. 보고서 실행과 저장, Repository Server에 업로드, DeliveryServer 로 요청에 필요한 파라미터는 HTTP POST 파라미터로 전달합니다.

5.1. Server-Side MultiSend 의 POST 파라미터

5.1.1. 파라미터 상세

키	설명	구분	비고
opcode	명령 처리 코드, Server-Side Export는 540 입니다. 540 이 아닌 다른 값일 경우 Server-Side MultiSend가 되지 않습니다.	필수	
batch	요청할 작업을 아래와 같은 형식으로 순서대로 작성합니다. Export 후 순서대로 요청하게 됩니다. prefix@RPS - Repository Server 로 업로드 요청 prefix@DLV:mail – Delivery Server 로 작업 요청	필수	DLV 지원가능 기능 mail, fax, prn(인쇄)
mrd_path	mrd 파일의 경로 입니다. 경로는 URL 경로 또는 상대경로를 사용합니다. 상대경로는 서버의 *{report.mrd.dir}* 경로를 기준으로 한 상대 경로 입니다.	필수	
mml_path	mml 파일의 경로 입니다. 경로는 URL 경로 또는 상대경로를 사용합니다. 상대경로는 서버의 *{report.mrd.dir}* 경로를 기준으로 한 상대 경로 입니다. 이 키는 mrd_path가 적용된 경우에는 무효화 되므로 함께 사용할 수 없습니다.		
mml_data	mml 전문입니다. 이 키는 mrd_path나 mml_path가 적용된 경우에는 무효화 되므로 함께 사용할 수 없습니다		
mrd_param	mrd_path, mml_path, mml_data에 설정된 파일 또는 전문을 실행하는 파라미터 입니다. RDViewr.FileOpen() 의 두 번째 파라미터와 동일합니다.	필수	
protocol	export 결과를 응답하는 형식입니다. Reporting Server 의 응답방식은 async, sync 방식을 지원합니다	옵션	기본값: *{server.protocol}*
delete_file	protocol의 값이 file로 설정된 경우, 클라이언트로 파일	옵션	기본값 : true

	을 내려주고나서, 서버에 만든 파일을 삭제할지 결정하기 위한 설정입니다. True로 설정하면 클라이언트로 파일을 내려준 후, 서버에 만든 파일을 삭제합니다. False로 설정하면 클라이언트로 파일을 내려준 후 서버에 만든 파일을 삭제하지 않고 그대로 둡니다. 추가된 버전 : 6.3.0.142		
mrd_data	보고서 실행에 필요한 데이터를 직접 전송할 때 사용합니다. RDViewer.SetRdata() 또는 /rdata 파라미터와 동일합니다.	옵션	
ignore_err	여러 개의 요청을 수행하는 경우에 중간에 오류가 발생하게 되면 다음 요청을 이어서 수행하도록 설정 가능합니다. true 면 오류를 무시하고 다음 요청을 수행하게 되며 false 이면 오류가 난 시점에 요청을 중지합니다.	옵션	기본값: false
prefix@export_type	export 할 파일의 타입입니다. 아래와 같은 포멧을 지원합니다. mrr, pdf, doc, xls, csv, ppt, hwp, png, jpg ※ 제약사항 - 저장 파라미터(예, /rcsv, /rexport등)와 동시 사용불가.	옵션 (필수)	prefix@DLV:mail 인 경우에만 필수
prefix@export_name	export 할 파일의 파일명 입니다. 파일명은 확장자를 포함합니다.	옵션	기본값: 서버에서 중복되지 않는 임의의 값으로 생성
prefix@user_id	Repository Server의 사용자 계정입니다.	필수	prefix@RPS 인 경우
prefix@user_pwd	Repository Server의 사용자 계정 패스워드 입니다.	필수	
prefix@category_id	문서를 저장할 Repository Server의 카테고리 ID 입니다.	필수	
prefix@file_title	Repository Server에 업로드 할 문서의 제목입니다.	옵션	
prefix@file_description	Repository Server에 업로드 할 문서의 설명입니다.	옵션	
prefix@mail_from	보내는 사람 주소입니다.	필수	prefix@DLV:mail 인 경우
prefix@mail_to	받는 사람의 메일 주소입니다.	필수	
prefix@mail_cc	참조입니다.	옵션	

prefix@mail_bcc	숨은 참조입니다.	옵션	
prefix@mail_subject	메일의 제목입니다.	옵션	
prefix@mail_msg	메일의 내용입니다.	옵션	
prefix@mail_attach	추가적인 첨부파일이 있을 경우 추가할 수 있습니다.	옵션	
prefix@fax_from	보내는 사람의 fax 번호입니다.	옵션	prefix@DLV:fax 인 경우
prefix@fax_to	받는 사람의 fax 번호 입니다.	옵션	
prefix@fax_from_name	보내는 곳/사람의 이름입니다.	옵션	
prefix@fax_to_name	받는 곳/사람의 이름입니다.	옵션	
prefix@fax_from_mail	보내는 사람의 메일 주소 입니다.	옵션	
prefix@server_url	전송할 Repository/Delivery Server 의 service 주소 입니다.	옵션	

※ prefix : 동일한 파라미터를 구분하기 위하여 사용되며 요청 순서를 결정합니다. prefix 는 1 부터 요청 순서대로 기입해주시기 바랍니다.

5.1.2. 파라미터 예제

키	값	설명
opcode	540	Server-Side Export 명령 코드
batch	1@RPS,2@DLV:mail,3@DLV:prn	Repository Server 에 업로드 요청 후 Delivery Server 에 Mail, Print 를 요청함.
mrd_path	sample.mrd	*{report.mrd.dir}* 경로의 sample.mrd 를 이용하여 보고서를 실행함.
mml_path	sample.mml	*{report.mrd.dir}* 경로의 sample.mml 를 이용하여 보고서를 실행함.
mml_data	<?xml version="1.0" encoding="utf-8"?><MML ... </MML>	mml 전문으로 보고서를 실행함
mrd_param	/rfn [http://192.168.10.10/ReportingServer/m	보고서 실행에 필요한 데이터는 http://192.168.10.10/ReportingServer/mrd/sampl

	rd/sample.txt]	e.txt 를 사용함.
protocol	sync	응답 방식을 sync 로 설정
delete_file	false	클라이언트로 파일을 내려주고, 서버에 만든 파일을 삭제하지 않고 그대로 둡니다.
ignore_err	true	한 요청이 실패해도 다음 요청을 실행함.
1@export_type	pdf	pdf 로 저장
1@user_id	admin	Repository Server의 사용자 계정
1@user_pwd	1234	Repository Server의 사용자 계정 패스워드
1@category_id	11	문서를 저장할 Repository Server의 카테고리 ID
1@file_title	영업보고서	Repository Server에 업로드 할 문서의 제목
1@file_description	실적 발표 자료	Repository Server에 업로드 할 문서의 설명
1@server_url	http://192.168.10.10/RepositoryServer/service	업로드 할 Repository Server의 service 주소
2@export_type	doc	doc 로 저장
2@export_name	sample_20110601.doc	파일명은 sample_20110601.doc 로 저장
2@mail_from	mail_from@m2soft.co.kr	보내는 사람의 주소
2@mail_to	mail_to@m2soft.co.kr	받는 사람의 메일 주소
2@mail_subject	Mail Title	메일 제목
2@mail_msg	Content..중략...	메일 내용
3@server_url	http://192.168.10.10/DeliveryServer/service	인쇄를 요청할 Delivery Server의 service 주소

5.2. Server-Side MultiSend 의 Response

5.2.1. async 방식

요청을 정상적으로 Reporting Server가 수신했다면 바로 응답합니다. Server-Side MultiSend는 응답 후 서버에서 순차 처리 됩니다.
이러한 방식은 Web Application Server의 Servlet Thread 수와 커넥션 수를 감소시키므로 시스템 자원 절약할 수 있는 장점이 있습니다. Reporting Server의 기본 설정은 async 입니다.

- 형식

<결과코드>|<메시지>

- 결과코드

	결과코드	메시지	**응답**
수신 성공	1	임의의 값	**1|20121127141856439**
수신 실패	0	오류 메시지	**0|[ERS-11001]요청 정보를 파싱하는데 실패 했습니다.**

※ 임의의 값 – 서버에서 생성되는 중복되지 않는 임의의 값.

5.2.2. sync 방식

요청을 수신하고 Server-Side MultiSend 완료 후에 처리 결과를 응답합니다. Server-Side MultiSend 의 처리 결과를 반드시 응답 받아야 하는 경우에 사용할 수 있습니다.
이러한 방식은 처리 결과를 응답 받을 수 있는 장점이 있으나 Server-Side MultiSend 의 처리 시간 만큼 Web Application Server의 Servlet Thread 가 유지되고 커넥션이 증가한다는 단점이 있습니다.

- 형식

<결과코드>|<메시지>
:
<결과코드>|<메시지>

요청한 갯수에 따라 결과의 갯수도 달라집니다.

- 결과코드

	결과코드	메시지	**응답**
수신 성공	1	성공메세지	**1|m2soft.ers.handler.export.TransferHandler|2012/11/27/20121127141410010.doc|20121127141410783**
수신 실패	0	오류 메시지	**0|[ERS- 12101] 보고서 실행 중에 오류가 발생했습니다.(java.io.FileNotFoundException)**

- 결과메세지

1|m2soft.ers.report.core.handler.ExportHandler|2012/11/27/20121127141410010.pdf;2012/11/27/20121127141410010.doc;2012/11/27/20121127141410010.mrr
1|m2soft.ers.handler.export.TransferHandler|2012/11/27/20121127141410010.pdf|1
1|m2soft.ers.handler.export.TransferHandler|2012/11/27/20121127141410010.doc|20121127141410783
1|m2soft.ers.handler.export.TransferHandler|2012/11/27/20121127141410010.mrr|20121127141410784

위와 같이 결과메세지가 발생한 경우 첫번째줄은 ExportHandler 를 실행하여 필요한 파일을 모두 export 한 결과 입니다. 이 결과는 요청 개수에 상관없이 무조건 한번 나타납니다. 두번째줄 부터는 요청한 개수만큼 결과 값이 표시 됩니다.

5.3. Server-Side MultiSend 의 결과 파일

5.3.1. 저장 경로 설정

async 또는 sync 방식을 생성된 결과 파일은 *{report.save.dir}* 를 기준으로 하여 *{report.dir.pattern}* 의 날자 형식에 맞게 디렉터리를 생성하여 저장됩니다.

```
report.save.dir= /app/ReportingServer/report
report.dir.pattern=yyyy/MM/dd/
```

예를 들어 위와 같이 서버 설정이 되어 있고 현재 날짜가 2011/06/01 일 때 저장되는 경로는 /app/ReportingServer/report/2011/06/11 이 됩니다.

저장 디렉터리의 날짜 패턴은 *{report.dir.pattern}* 를 수정하여 변경할 수 있습니다.

{report.dir.pattern} 값	저장 디렉터리
yyyy/MM/dd/	일 단위로 디렉터리를 생성하여 저장
yyyy/MM/dd/HH/	시간 단위로 디렉터리를 생성하여 저장
yyyy/MM/dd/HH/mm/	분 단위로 디렉터리를 생성하여 저장

5.3.2. 저장 파일 확인

Server-Side MultiSend 에 성공하였다면 저장된 디렉터리에 결과 파일이 저장 된 것을 확인 할 수 있습니다.

```
/app/ReportingServer/report/2011/06/01>ls
sample_20110601.doc
```

저장되어진 파일을 이용하여 Repository Server 로 업로드 또는 Deliver Server 로 Mail, Fax, Print 요청을 하게 됩니다.

6. ers-invoker 를 이용한 Server-Side Reporting

ers-invoker는 ReportingServer와 함께 제공되는 Java 기반의 라이브러리 입니다. ers-invoker는 ReportingServer 로 HTTP 프로토콜로 요청을 전송하고 응답을 받을 수 있습니다.

6.1. 사용 환경

	S/W
OS	Unix Linux Windows 2000 이상
JRE	JRE 1.6 이상

6.2. ers-invoker API

Class	
m2soft.ers.invoker.http.ReportingServerInvoker Reporting Server 로 요청을 전송하고 응답을 받는 클래스 입니다.	
Constructor Detail	
java.lang.String	**ReportingServerInvoker**(java.lang.String host) Reporting Server로 요청을 전송하고 응답을 받는 invoker를 생성합니다. host는 Reporting Server의 서비스 URL 입니다. (예) http://www.m2soft.co.kr/ReportingServer/service **Parameters:** host – Reporting Server 의 서비스 URL
Method Detail	

void	**addParameter**(java.lang.String key, java.lang.String value) POST 로 전송할 파라미터를 추가합니다. **Parameters:** key – 파라미터 키 value – 파라미터 값
void	**addParameter**(java.lang.String key, byte[] value) POST 로 전송할 바이트 배열의 파라미터를 추가합니다. **Parameters:** key – 파라미터 키 value – 파라미터 값
java.lang.String	**invoke**() throws InvokerException Reporting Server 로 요청을 전송하고 응답을 받습니다. Sync 또는 Async 방식으로 요청할 때 사용합니다. **Return:** Reporting Server의 응답 메시지 **Throws:** m2soft.ers.invoker.InvokerException – 송수신 중에 예외가 발생할 경우
void	**invoke**(java.io.File file) throws InvokerException ReportingServer 로 요청을 전송하고 전송된 파일을 저장합니다. File 방식으로 요청할 때 사용합니다. **Parameters:** file - 전송된 파일을 저장할 File 객체 **Throws:** m2soft.ers.invoker.InvokerException - 송수신중 예외가 발생할 경우
void	**invoke**(java.lang.String filePath) throws InvokerException ReportingServer 로 요청을 전송하고 전송된 파일을 저장합니다. File 방식으로 요청할 때 사용합니다. **Parameters:** filePath - 전송된 파일을 저장할 경로

	Throws: m2soft.ers.invoker.InvokerException - 송수신중 예외가 발생할 경우
void	**setConnectTimeout**(int second) Connection Timeout 을 설정합니다. 설정한 second 만큼 접속을 기다립니다. 설정하지 않을 경우 기본값은 3초 입니다. 이 값의 설정은 invoke() 호출 전에 설정해야 합니다. **Parameters:** second - Connection Timeout 시간(초단위) **Throws:** java.lang.IllegalArgumentException – second 값이 0보다 작은 값일 경우
void	**setReconnectionCount**(int count) 접속 시도 회수를 설정합니다. 0보다 작은 count 값일 경우 기본값 3회를 시도 합니다. 설정하지 않을 경우 3회 접속을 시도 합니다. 이 값의 설정은 invoke() 호출 전에 설정해야 합니다. **Parameters:** count - 접속 시도 회수 **Throws:** java.lang.IllegalArgumentException – count 값이 1보다 작은 값일 경우
void	**setReadTimeout**(int second) 송수신 Timeout 을 설정합니다. 설정한 second 동안 Reporting Server 와의 패킷 송수신이 없을 경우 InvokerException 를 발생시키며 종료됩니다.

	설정하지 않을 경우 기본값은 0초 입니다. Timeout 이 0 일 경우 Timeout 없이 무한 대기 합니다. 이 값의 설정은 invoke() 호출 전에 설정해야 합니다. **Parameters:** second - 송수신 타임아웃 시간(초단위) **Throws:** java.lang.IllegalArgumentException – second 값이 0보다 작은 값일 경우
void	**setCellInfo**(int index, java.lang.String title, java.lang.String variable, java.lang.String summarize, java.lang.String command, java.lang.String attribute, long option, java.lang.String refField) 런타임리포트 객체에 보고서 실행에 필요한 데이터 매핑 정보를 지정하는 메소드 입니다. 반드시 appendTableColumn 메소드와 함께 사용합니다. **Parameters:** index - CELL 인덱스. 1부터 시작 title - CELL에 표시 할 텍스트 내용 variable - 필드명. 데이터 셋 연결 대화 상자 필드명. 변수명을 지정 summarize - 요약 작성. 데이터 셋 연결 대화 상자 요약 작성. 요약 작성 내용 지정 command - 수행코드. CELL속성을 바꾸기 위한 코드를 나타내는 문자열 attribute - 수행코드에 대한 속성값 option - 셀 병합 관련 옵션. 데이터 셋 연결 대화 상자의 셀 병합 관련 옵션 지정.option 값은 다음 중 한 개 이거나 여러 개를 조합(논리적OR연산)하여 사용할 수 있습니다. 셀 병합 (1), 모두합치기 (2), 옆 셀 병합 (4), 한번만쓰기 (8) refField - 참조 필드명. 데이터 셋 연결 대화 상자의 참조 필드 리스트 지정. 필드명이 하나 이상인 경우 @ 문자로 구분하여 여러 개의 필드명을 지정합니다.
void	**appendTableColumn**(java.lang.String objectName, long width, int option) 런타임리포트 객체에 하나의 열(컬럼)을 오른쪽으로 추가를 하는 메소드 입니다. 런

	타임리포트 표 객체의 첫번째 셀 필드명을 KEY 로 해서 해당 표에 열(컬럼)을 추가 합니다. 열(컬럼) 추가시 열의 너비를 주고 추가하며 열의 너비가 0 인 경우는 열(컬럼)을 추가 하는 것이 아니라 표의 크기를 자동으로 조정하는 기능을 수행 합니다. **Parameters:** objectName - 열(컬럼)을 추가 할 런타임리포트 표 객체의 첫번째 셀 필드명 width - 열(컬럼) 너비. 단, 값을 0으로 준 경우는 추가할 열의 너비를 의미하는 것이 아니라, 표 크기 자동조정 기능을 합니다. 0 : 페이지 너비/높이에 맞추게 표 크기 자동 조정. 표의 크기가 페이지 너비/높이 보다 크거나 작은경우 페이지 너비/높이에 맞추어 표 전체 크기를 자동으로 조정합니다. -1 : 표 객체를 삭제 합니다. option - 1 또는 2 의 값을 갖습니다. width가 0인 경우와 0보다 큰 경우, option의 인수는 다음과 같이 각각 다르게 적용됩니다. 경우1) 열너비를 0보다 큰 값으로 지정한 경우 1 : 표의 왼쪽 x좌표 고정. 표의 좌측이 고정된 상태로 열(컬럼)은 오른쪽으로 추가됩니다. 2 : 표의 오른쪽 x좌표 고정. 표의 우측이 고정된 상태로 열(컬럼)은 오른쪽으로 추가됩니다. 경우2) 열너비를 0으로 지정한 경우 1 : 페이지 우측 가상선에 맞추어 Adjust 합니다. 표 가로 크기를 조정 합니다. 2 : 페이지 하단 가상선에 맞추어 Adjust 합니다. 표 세로 크기를 조정 합니다.
Java.lang.String	**makeUuid**() 요청을 처리하기 전에 외부에서 식별할 수 있는 고유 ID를 생성합니다. 이 ID로 요청 처리중인 작업 중 취소할 작업을 선택해서 cancel(uuid)로 취소할 수 있습니다. **Return:** 요청 고유 ID (UUID)
void	**cancel**() **cancel**(java.lang.String uuid) d) 현재 요청중인 작업 취소

	Parameters:
	uuid- 취소할 요청을 지정하여 취소할 때 사용합니다.
	
	같은 JSP 페이지 또는 같은 Class에서 작업을 취소할 경우, 같은 invoker를 사용하여 cancle() API를 실행하므로 보통 uuid를 설정할 필요가 없습니다.
	ReportingServerInvoker invoker = new ReportingServerInvoker();
	...
	invoker.invoke();
	...
	invoker.cancel();
	
	그러나 다른 페이지나 다른 클래스의 경우라면 어떤 요청을 취소하는지 알 수 없으므로 uuid가 필요합니다.
	Page1
	ReportingServerInvoker invoker = new ReportingServerInvoker();
	...
	request_uuid.add(invoker.makeUuid()); //공유할수 있는 곳에 uuid 추가
	invoker.invoke();
	Pase2
	ReportingServerInvoker invoker = new ReportingServerInvoker();
	...
	invoker.cancel(uuid); //특정 조건에 맞는 uuid로 취소

6.3. Server-Side Export 샘플 (Sync, Async 방식)

```
01 import m2soft.ers.invoker.InvokerException;
02 import m2soft.ers.invoker.http.ReportingServerInvoker;
03
04 public class Sample {
05
06    public static void main(String[] args)
07    {
08       ReportingServerInvoker invoker = new ReportingServerInvoker(
09       "http://localhost:8080/ReportingServer/service");
10
```

```
11      invoker.setCharacterEncoding("euc-kr");   //캐릭터셋
12      invoker.setReconnectionCount(3);        //재접속 시도 회수
13      invoker.setConnectTimeout(5);          //커넥션 타임아웃
14      invoker.setReadTimeout(30);            //송수신 타임아웃
15
16      invoker.addParameter("opcode", "500");
17      invoker.addParameter("mrd_path", "sample.mrd");
18      invoker.addParameter("mrd_param",
19      "/rfn [http://localhost:8080/ReportingServer/mrd/sample.txt]");
20      invoker.addParameter("export_type", "doc");
21      invoker.addParameter("export_name","sample_20110601153654.doc");
22      invoker.addParameter("protocol", "sync");
23
24      try
25      {
26         String response = invoker.invoke();   //요청전송 응답받기
27         System.out.println(response);
28      }
29      catch(InvokerException e)
30      {
31         e.printStackTrace();
32      }
33   }
34 }
```

- ers-invoker를 사용하기 위해서는 ers-invoker.jar 파일을 클래스패스에 추가 해야 합니다.

- **import** m2soft.ers.invoker.InvokerException;
  import m2soft.ers.invoker.http.ReportingServerInvoker;

  필요한 클래스들을 import 합니다.

- ReportingServerInvoker invoker = **new** ReportingServerInvoker(
  "http://localhost:8080/ReportingServer/service");

ReportingSrever 와 통신하기 위한 ReportingServerInvoker 를 생성합니다. 생성자의 인수로 Reporting Server의 service url을 입력합니다.

- invoker.setCharacterEncoding("euc-kr"); //캐릭터셋
  invoker.setReconnectionCount(3); //재접속 시도 회수
  invoker.setConnectTimeout(5); //커넥션 타임아웃
  invoker.setReadTimeout(30); //응답 타임아웃

  캐릭터 셋과 통신에 필요한 설정을 셋팅합니다. 자세한 설명은 *4.1 ers-invoker API* 를 참조합니다.

- invoker.addParameter("opcode", "500");
  ...(생략)

  Server-Side Export 에 필요한 파라미터를 추가합니다. 추가된 파라미터는 POST 파라미터로 Reporting Server에 전송됩니다.

- String response = invoker.invoke();

  Reporting Server 로 요청을 전송하고 응답을 받습니다. Reporting Server 로 전송에 실패했을 때는 m2soft.ers.invoker.InvokerException 이 발생합니다.
  response 는 '1' 로 시작할 경우 성공이며 '0'으로 시작할 경우 실패입니다. response 형식에 대한 설명은 *3.2 Server-Side Export 의 Response* 를 참조합니다.

6.4. Server-Side Export 샘플 (File 방식)

```
01 import java.io.File;
02 import m2soft.ers.invoker.InvokerException;
03 import m2soft.ers.invoker.http.ReportingServerInvoker;
04
05 public class Sample {
06
07    public static void main(String[] args)
```

```
08  {
09      ReportingServerInvoker invoker = new ReportingServerInvoker(
10      "http://localhost:8080/ReportingServer/service");
11
12      invoker.setCharacterEncoding("euc-kr"); //캐릭터셋
13      invoker.setReconnectionCount(3);        //재접속 시도 회수
14      invoker.setConnectTimeout(5);           //커넥션 타임아웃
15      invoker.setReadTimeout(30);             //송수신 타임아웃
16
17      invoker.addParameter("opcode", "500");
18      invoker.addParameter("mrd_path", "sample.mrd");
19      invoker.addParameter("mrd_param",
20      "/rfn [http://localhost:8080/ReportingServer/mrd/sample.txt]");
21      invoker.addParameter("export_type", "doc");
22      invoker.addParameter("export_name","sample_20110601153654.doc");
23      invoker.addParameter("protocol", "file");
24
25      try
26      {
27        // 방법 1,2,3 중 한 가지를 선택하여 사용.
28         // 방법 1 : 파일 경로 전달
29         invoker.invoke("C:/sample.doc");
30
31        // 방법 2 : 파일 객체 전달
32        invoker.invoke(new File("C:/sample2.doc"));
33
34        // 방법 3 : 스트림 전달
35        InputStream is = null;
36        OutputStream os = null;
37        try
38        {
39           is = invoker.invokeAndGetStream();
40           os = new FileOutputStream(new File("C:/sample3.doc"));
41
42           byte[] buffer = new byte[1024];
43           int read = -1;
```

```
44            while((read = is.read(buffer)) != -1)
45            {
46               os.write(buffer, 0, read);
47            }
48         }
49         finally
50         {
51            try{ if(is != null) is.close(); } catch(IOException ignore){}
52            try{ if(os != null) os.close(); } catch(IOException ignore){}
53
54            invoker.disconnect();
55          }
56       }
57       catch(InvokerException e)
58       {
59          e.printStackTrace();
60       }
61    }
62 }
```

6.5. AppendReport 기능 사용하기

AppendReport 기능은 여러 개의 보고서를 더하여 하나의 보고서를 생성하는 기능입니다. AppendReport 를 사용할 경우 전달하는 파라미터를 구분자를 이용하여 더한 후에 전송합니다. 다음의 코드는 파라미터를 전달하는 예제 입니다.

```
// 이전 코드 생략

char separator = (char)0x04;      // 구분자

String mrd_path = "sample_1.mrd" + separator + "smaple_2.mrd";
String mrd_param = "/rp [m2soft]" + separator + "/rp [013654]";
String mrd_data = "엠투소프트@기술지원본부@"
             + separator + "02-0000-0000@홍길동@";
```

```
invoker.addParameter("opcode", "500");
invoker.addParameter("mrd_path", mrd_path);
invoker.addParameter("mrd_param", mrd_param);
invoker.addParameter("mrd_data", mrd_data);
invoker.addParameter("export_type", "doc");
invoker.addParameter("export_name","sample_20110601153654.doc");
invoker.addParameter("protocol", "sync");

// 이하 코드 생략
```

- **0x04 문자를 구분자로 사용하여 파라미터를 더합니다. 구분자 문자는 일반문자와의 중복을 피하기 위하여 아스키 제어문자인 0x04 를 사용합니다.**

- **mrd_path 와 mrd_param 은 필수 값이므로 반드시 포함해야 합니다.**
  단, mrd_path 대신 mml_path 또는 mml_data를 사용할 수 있습니다.

- **mrd_data 는 옵션 값입니다. 데이터를 직접 전달하는 경우에만 전송하면 됩니다.**

- **구분자로 전달되는 파라미터 값의 갯수가 일치하지 않으면 오류가 발생합니다.**

6.6. Runtime Report 기능 사용하기

Runtime Report 기능은 보고서에 표시되는 표의 컬럼을 동적으로 지정하는 기능입니다.
Runtime Report 기능은 Java Engine을 사용하는 경우에만 지원합니다.

```
01 import m2soft.ers.invoker.InvokerException;
02 import m2soft.ers.invoker.http.ReportingServerInvoker;
03
04 public class Sample {
05
06 public static void main(String[] args)
07    {
08       ReportingServerInvoker invoker = new ReportingServerInvoker(
09       "http://localhost:8080/ReportingServer/service");
10
11       invoker.setCharacterEncoding("euc-kr");   //캐릭터셋
```

```
12      invoker.setReconnectionCount(3);          //재접속 시도 회수
13      invoker.setConnectTimeout(5);             //커넥션 타임아웃
14      invoker.setReadTimeout(30);               //응답 타임아웃
15
16      invoker.addParameter("opcode", "500");
17      invoker.addParameter("mrd_path", " runtime.mrd");
18      invoker.addParameter("mrd_param", "/rv Title[거래내역조회] custName[홍길동]
19      countNo[123-45678-90] startDate[20110107] endDate [20110707] /rfn
20      [http:// localhost:8080/ReportingServer/mrd/runtime.txt]");
21      invoker.addParameter("export_type", "mrr");
22      invoker.addParameter("protocol", "sync");
23
24      invoker.setCellInfo(1, "거래일시", "", "", "", "", 0,"");
25      invoker.setCellInfo(2, "", "a1", "", "", "", 0,"");
26      invoker.setCellInfo(3, "합계", "", "", "", "", 4,"");
27      invoker.appendTableColumn("TABLE_A", 130, 1);
28
29      invoker.setCellInfo(1, "적요", "", "", "", "", 0,"");
30      invoker.setCellInfo(2, "", "a3", "", "", "", 0,"");
31      invoker.setCellInfo(3, "합계", "", "", "", "", 4,"");
32      invoker.appendTableColumn("TABLE_A", 60, 1);
33
34      invoker.setCellInfo(1, "기재내용", "", "", "", "", 0,"");
35      invoker.setCellInfo(2, "", "a4", "", "", "", 0,"");
36      invoker.setCellInfo(3, "합계", "", "", "", "", 4,"");
37      invoker.appendTableColumn("TABLE_A", 100, 1);
38
39      invoker.setCellInfo(1, "취급점", "", "", "", "", 0,"");
40      invoker.setCellInfo(2, "", "a8", "", "", "", 0,"");
41      invoker.setCellInfo(3, "", "", "", "", "", 0,"");
42      invoker.appendTableColumn("TABLE_A", 60, 1);
43
44      invoker.setCellInfo(1, "메모", "", "", "", "", 0,"");
45      invoker.setCellInfo(2, "", "a2", "", "", "", 0,"");
46      invoker.setCellInfo(3, "", "", "", "", "", 0,"");
47      invoker.appendTableColumn("TABLE_A", 80, 1);
```

```
48
49          invoker.appendTableColumn("TABLE_A", 0, 1);
50
51          try
52          {
53             String response = invoker.invoke();   //요청 전송, 응답받기
54
55             System.out.println(response);
56          }
57          catch(InvokerException e)
58          {
59             e.printStackTrace();
60          }
61       }
62    }
```

7. Web Service

7.1. Web Service 소개

Web Service는 다른 기종간의 통신을 위해 만들어진 분산 컴퓨팅 시스템입니다. Web Service라는 이름을 가지고 있지만, WWW과 혼동하지 않아야 합니다. 비슷한 목적으로 만들어진 시스템으로는 CORBA, RMI, DCOM들이 있습니다. Web Service는 통신 end point들이 XML을 이용해서 통신을 하고, 메시지 지향적입니다.

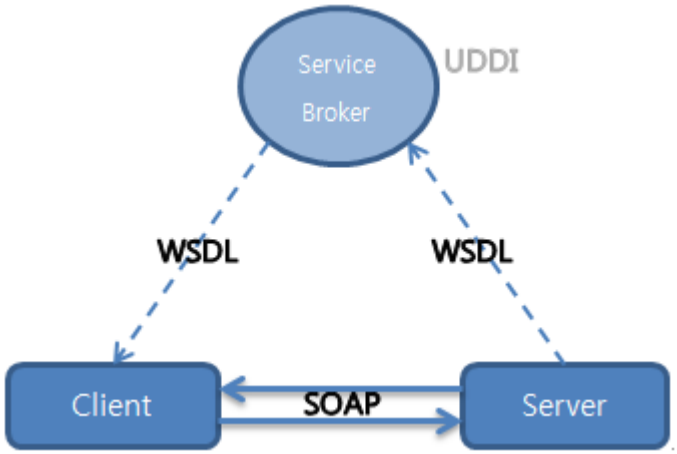


그림 7.1

이 시스템에서 알아야 할 기술은 크게 UDDI, WSDL, SOAP입니다. 이들은 그림 7.1처럼 맞물려 돌아갑니다. 위 그림에서 Server는 Reporting Server이고 앞으로의 설정에서 WSDL을 client가 사용 가능하도록 설정할 것입니다. 그림을 통해서 Web Service의 동작 시나리오를 설명하자면 다음과 같습니다.

1. Server는 서비스를 구현한 뒤에 client에게 공개할 인터페이스를 정합니다.
2. Server는 공개할 인터페이스를 WSDL로 기술합니다.
3. WSDL을 UDDI를 이용해 service broker에 등록합니다
4. 이제 client는 service broker에게 service를 요청합니다
5. Service broker는 server의 위치와, 인터페이스를 기술한 WSDL을 알려줍니다.
6. Client는 WSDL의 정보를 기반으로 server와 통신합니다.

만약 이 리스트에서 client가 server의 WSDL을 이미 가지고 있다면 service broker에게 WSDL을 요청하는 과정은 생략 될 수 있습니다. Service broker는 WSDL을 등록하고 있는 커다란 저장 장치이고, 그 정보들을 조작하는 것이 UDDI이다. UDDI는 그런 것들을 만들고 조작하기 위한 스팩입니다.
Reporting Server는 기본적으로 UDDI를 사용하지 않습니다.

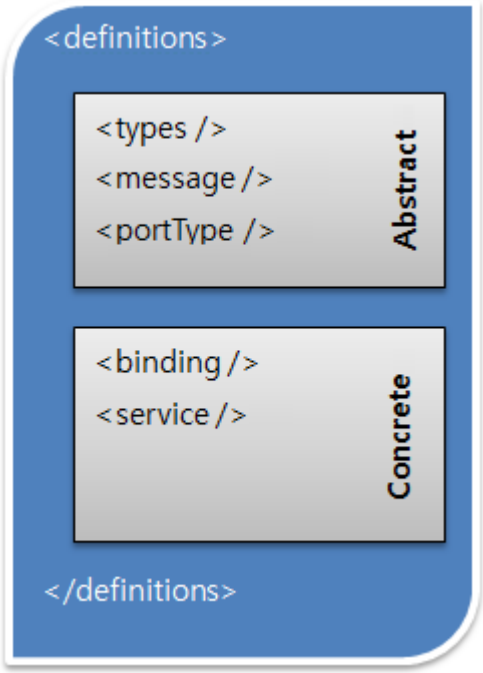
7.1.1. WSDL

WSDL 소개

WSDL은 XML로 기술된 언어로, Web Service Description Language의 약자이다. 커뮤니케이션의 양 끝에서 사용되며 어떤 서비스에서 어떤 operation이 가능한지, 그리고 그 operation의 input과 output은 무엇인지 등이 기술되어 있습니다.

다시 말해 WSDL은 client와 service provider 간의 인터페이스를 정의 한 것입니다. 그리고WSDL을 사용해 자신이 만든 인터페이스의 유효성을 확인할 수도 있습니다.

WSDL의 포맷



definitions : WSDL의 root element 입니다.

types : WSDL에서 사용되는 data type을 정의합니다..

message : Operation을 수행하기 위한 파라미터를 정의합니다.

portType : Opertaion을 정의합니다. 언어 차원에서 보면 interface와 같으며 여러 개의 메서드를 정의 하고 있습니다.

binding : portType과 실제 코드를 연결합니다.

service : Interface나 함수들을 담고 있는 컨테이너입니다.

포맷이 굉장히 복잡한 것 같지만, java 언어의 interface와 비슷합니다. 우선 types와 message, portType이 인터페이스를 기술합니다. types가 data type과 같고 message가 input parameter와 return parameter를 설정해주고, portType이 그 것들을 사용하는 함수를 정의합니다. 그리고 binding과 service는 위에서 정의한 것들을 실제 코드와 연결시킵니다.

7.1.2. WSDD

WSDD는 Web Service Deployment Descriptor의 약자로 XML로 기술되어있습니다. WSDD는 service들이 AXIS 노드 안에 어떻게 설치되어있는지를 기술합니다. Server side와 client side 두 곳에서 쓰일 수 있는데, 그 두 곳에서의 용도를 설명하면 다음과 같습니다.

Server side	어떤 Web Service가 설치되어 있는지, 그 웹 서비스에 어떤 operation이 있는지

	등이 기술되어 있다.
Client side	만들어져 있는 웹 서비스를 deploy또는 undeploy하기 위한 정보들이 들어있다.

7.2. Web Service 설정

Web Service는 Reporting Server를 설치가 정상적으로 되어있는 상태에서 추가적인 설정이 필요 없습니다. 다만 Web Service 주소를 바꾸고 싶다면 7.2.2장을 참고하시기 바랍니다.

7.2.1. Web Service 동작 확인

웹 브라우저로 *http://<Reporting Server주소>/webservice/Service?wsdl* 를 열어서 다음과 같은 화면이 나오면 정상적으로 웹 서비스가 deploy되었다고 할 수 있습니다.(그림 7.2)

그림 7.2

웹 브라우저로 http://*<Reporting Server주소>*/webservice/Service?wsdl 을 열면 WSDL파일의 내용을 볼 수 있습니다. 브라우저마다 조금씩 다르게 출력되니 이 점을 염두 해두시기 바랍니다. 만약 XML구조가 보이지 않는다면 오른쪽 마우스 버튼을 눌러 소스보기를 하면 XML소스를 볼 수 있습니다. 앞으로 WSDL의 주소는 위와 같을 것이며, 클라이언트를 만드는 데에 필요하므로 기억해두어야 합니다.

7.2.2. Web Service 이름 변경

Web Service의 URL은 필요에 의해 변경될 수 있습니다. 그 방법은 다음과 같습니다.

1. 탐색기등을 이용해 Reporting Server가 설치된 디렉토리로 이동합니다.
2. /WEB-INF/web.xml 을 열어 servlet-mapping 중 AxisServlet이란 이름을 가진 매핑 규칙을 찾습니다.
3. url-pattern을 원하는 것으로 바꿉니다. 여기선 '**A/***'으로 바꾸었다고 가정합니다.
4. /WEB-INF/conf/axis2.xml을 열어 parameter 태그 중 name 어트리뷰트가 servicePath인 것을 찾아 **A**로 바꿉니다.
5. /WEB-INF/webservices/Services/META-INF/services.xml을 열어 service 태그의 name 어트리뷰트를 원하는 이름으로 바꿉니다. 여기선 '**B**'로 바꾸었다고 가정합니다.
6. Tomcat을 재 시작 합니다.

위의 단계를 수행하면 Web Service 주소가 *http://<Reporting Server주소>/**A/B**?wsdl* 로 바뀐 것을 확인할 수 있습니다.

7.3. Client 개발

Web Service의 client를 개발하는 방법은 여러가지가 있습니다. 여기서는 Java와 Eclipse를 이용해 Web Servcie에 요청을 보내는 코드를 자동 생성하는 방법을 소개합니다.

7.3.1. Eclipse를 이용한 Java Client

Eclipse를 이용한 개발은 간편함이 장점이라고 할 수 있습니다. WSDL의 주소만 알고 있다면, Eclipse가 WSDL을 소스로 필요한 interface, service provider와 통신하기 위해 필요한 proxy까지 자동으로 생성해줍니다. 개발자가 할 것은 단지 class를 사용하는 것 뿐입니다. 아래의 간단한 튜토리얼을 따라가면 기본적인 client를 쉽게 만들 수 있습니다.

1. Ctrl + N을 누르거나 메뉴의 File > New > Other를 선택해서 새 프로젝트를 생성할 준비를 합니다. 그림 7.3과 같이 새 프로젝트의 타입는 Dynamic Web Project를 선택합니다.

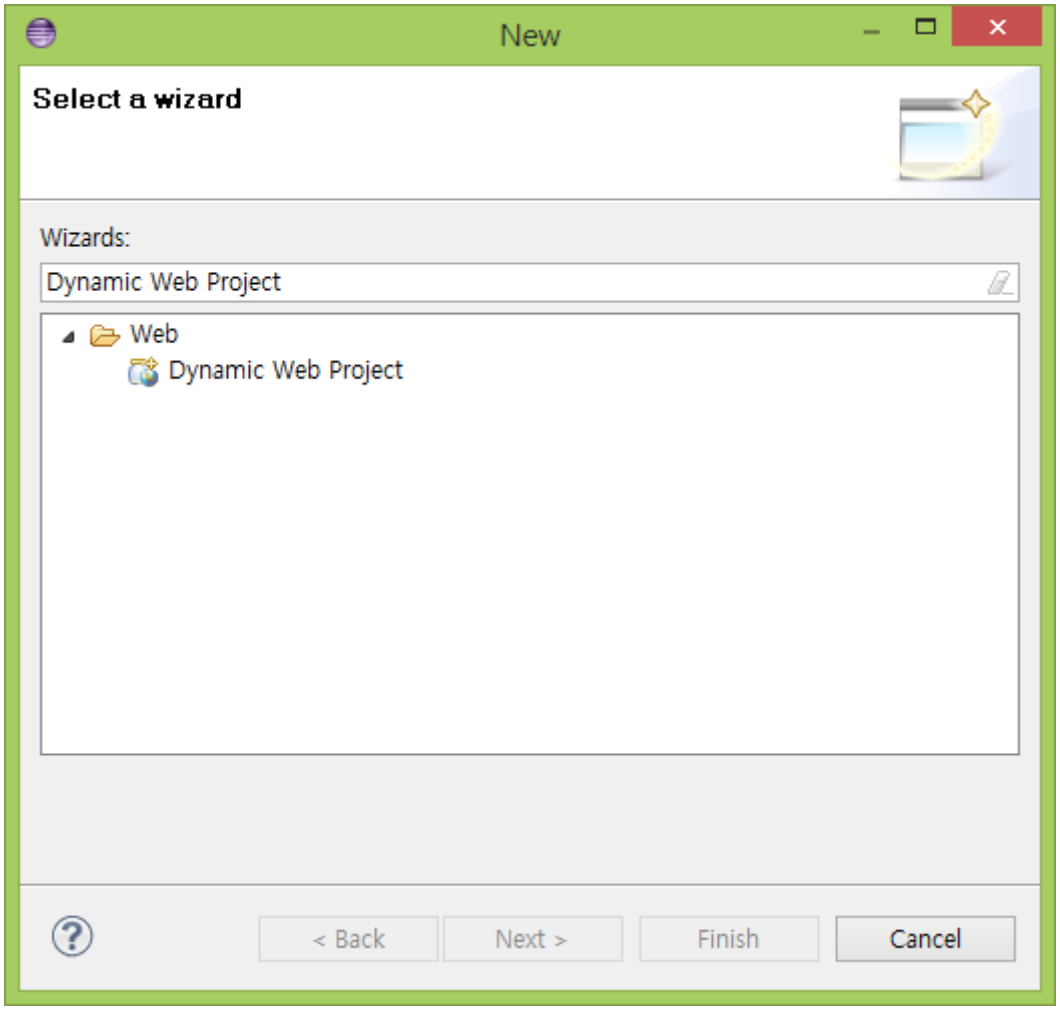


그림 7.3

Project name란에 원하는 프로젝트 이름을 적고, tomcat과 dynamic web module version의 설정을 합니다. 여기서 프로젝트 이름은 web service client로 지었습니다.(그림 7.4)

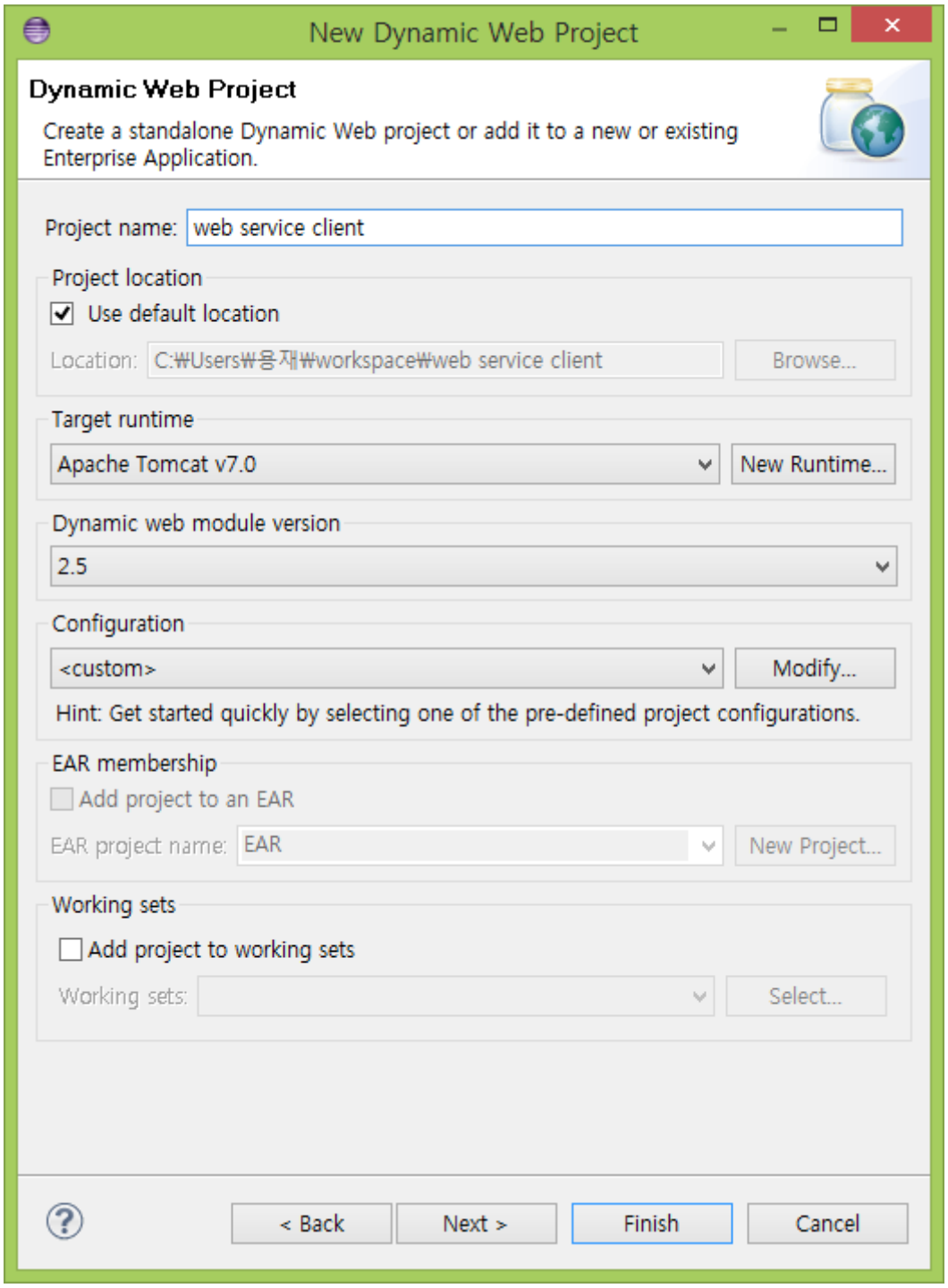


그림 7.4

2. Configuration 섹션에서 Modify를 눌러 Project Facets 창을 띄우고 Axis2 Web Service, Dynamic Web Module, Java가 체크되어있는지 확인하고 체크되어있지 않다면 체크합니다. 그리고 OK 버튼을 눌러 창을 닫습니다.(그림 7.5)

Finish를 눌러 프로젝트를 생성합니다. (그림 7.3)

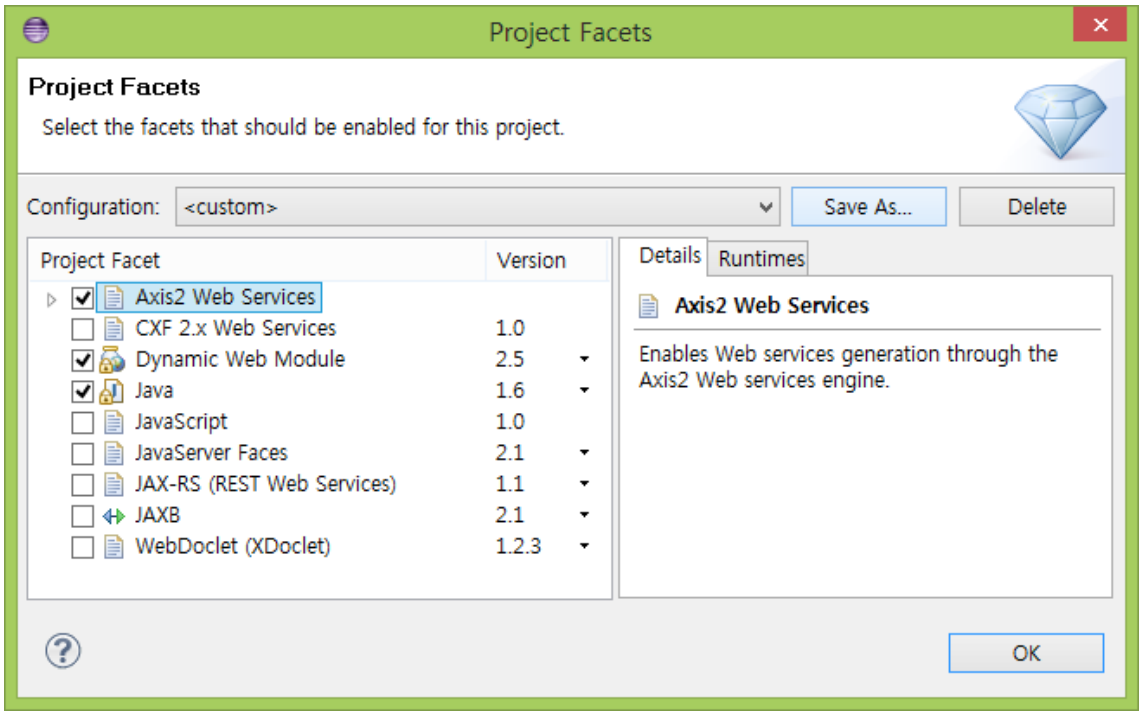


그림 7.5

3. Dynamic Web Project 프로젝트를 만들고 곧바로 project explorer에서 그 프로젝트를 오른쪽 클릭해서 문맥메뉴를 호출합니다. 그 문맥메뉴의 New 통해 새로운 프로젝트를 만듭니다. 이때 만들어야 할 프로젝트의 타입은 web service client입니다. 만약 New 메뉴에 web service client가 없다면 New > Other 를 선택합니다.(그림 7.6)

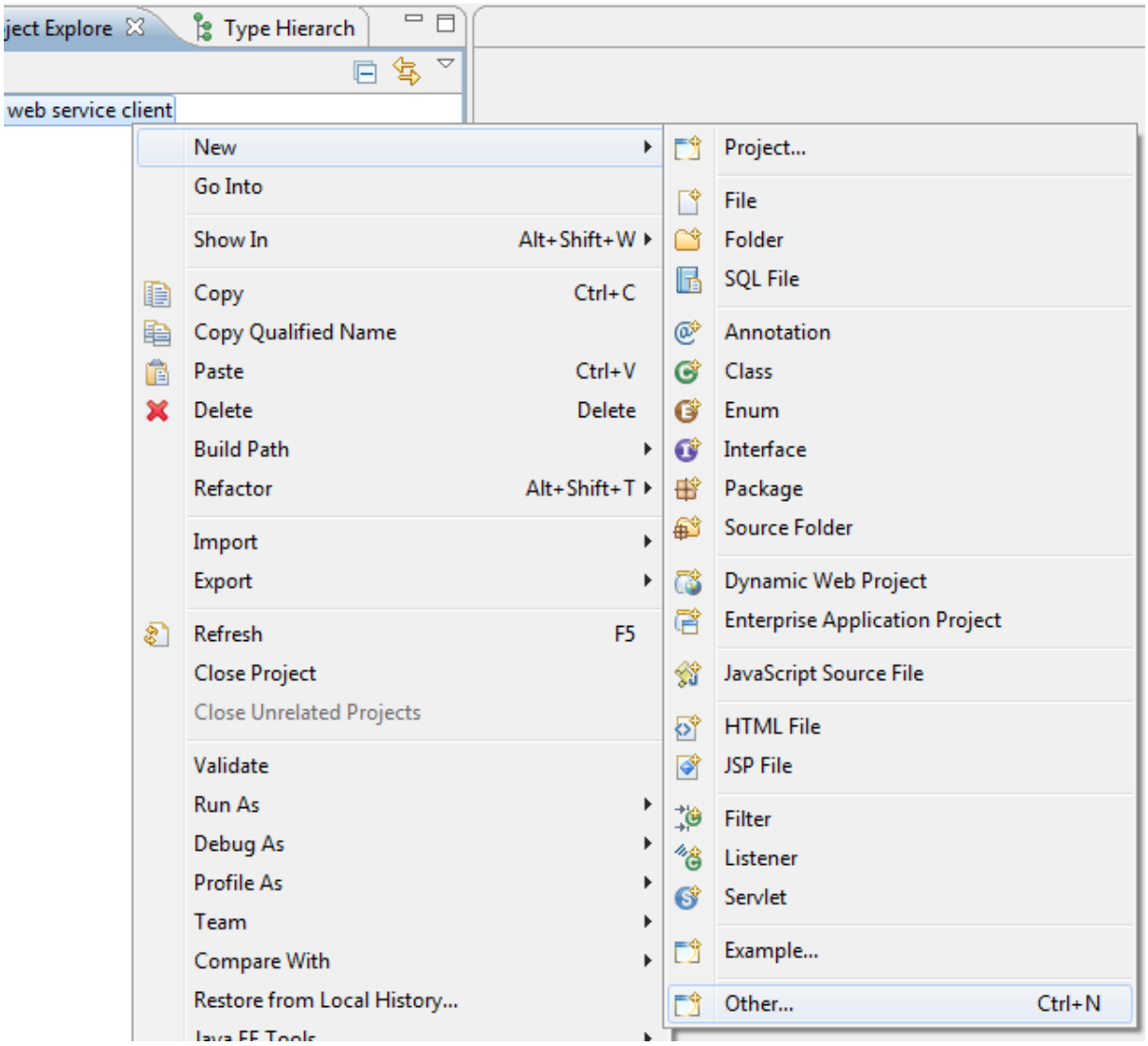


그림 7.6

4. New > Other를 눌러서 나오는 프로젝트 타입 창에서 web service client를 선택한 뒤 Next를 누릅니다.(그림 7.7)

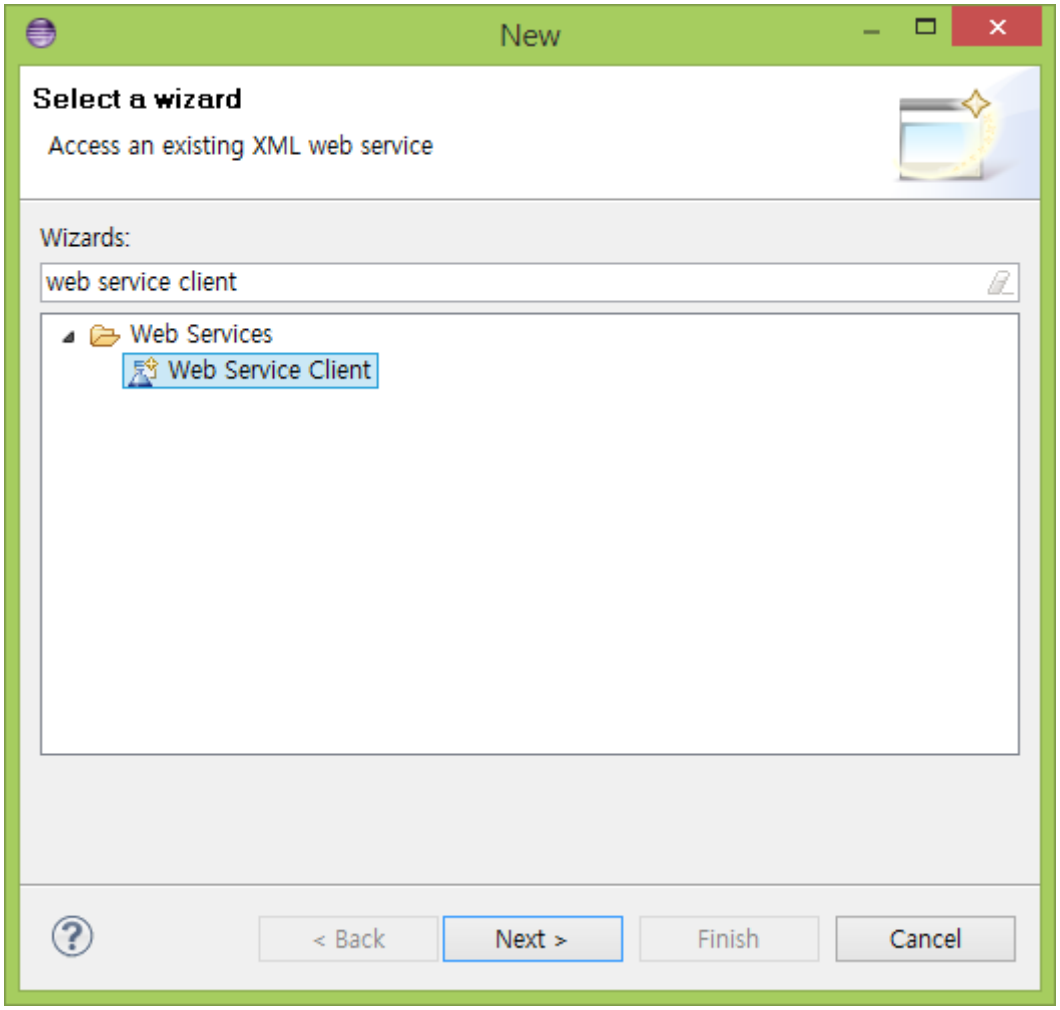


그림 7.7

5. web service client를 선택하면 프로젝트 초기 설정을 시작합니다. 아래와 같은 설정창의 Service definition에 Reporting Server의 WSDL주소를 적어 넣습니다. 그리고 아래에서 client 개발의 단계를 설정해줍니다. 각 단계는 다음과 같은 의미가 있습니다. 여기에선 Testing을 선택하도록 하겠습니다(그림 7.7).

   A. Develop : this will create the client code

   B. Assemble : this ensures that the project that will host the client gets associated to an EAR when required by the target application server

   C. Deploy : this will create the deployment code for the client

   D. Install : this will install the client on the choen server

E. Start this will start the server once the client has been installed on it

F. Test : this will provide various options for testing the client

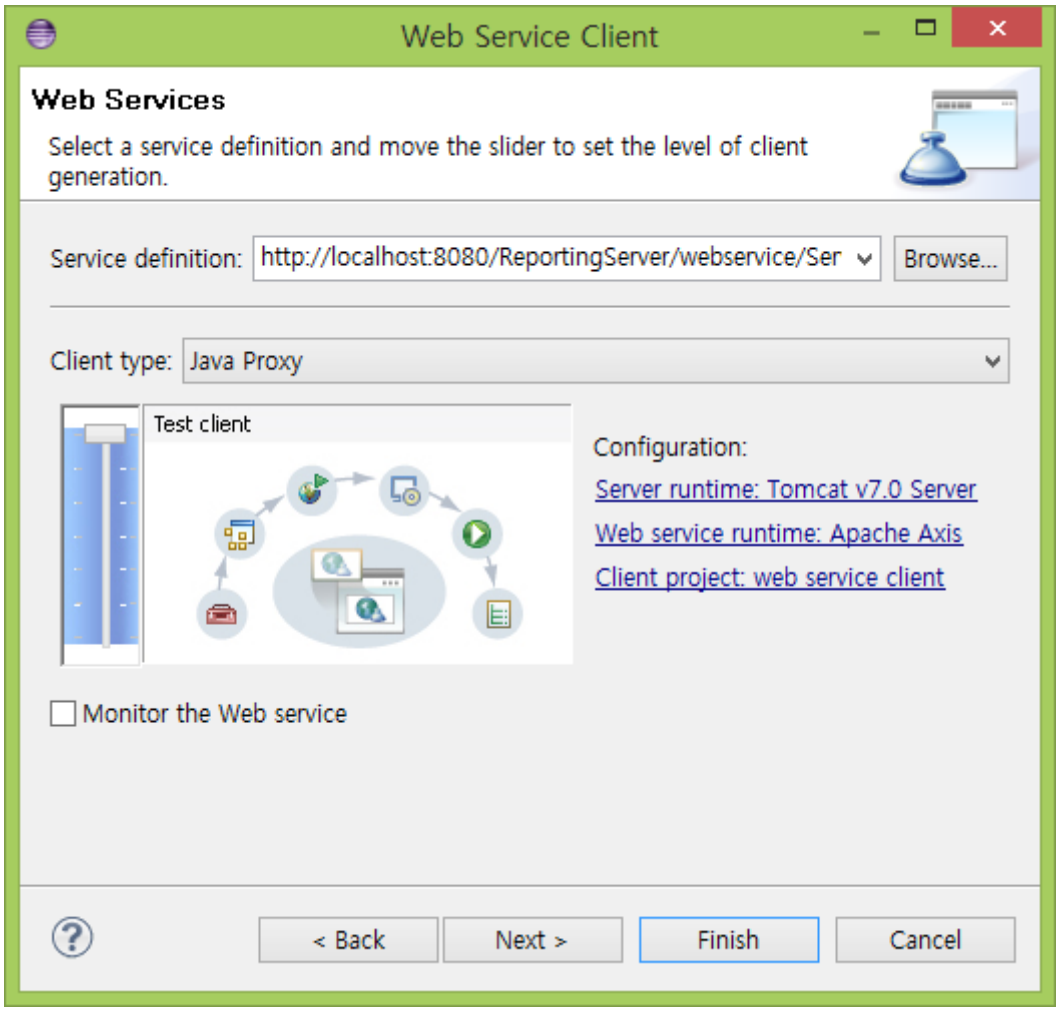


그림 7.8

6. 그림 7.8에서는 Configuration 항목중 Web service runtime이 Apache Axis로 되어있는 것을 확인할 수 있습니다. 이를 클릭하여 설정 창을 띄웁니다. 여기에서 web service의 runtime환경을 골라줍니다. 여기서는 Axis2를 고릅니다.

OK를 눌러 설정을 적용합니다.

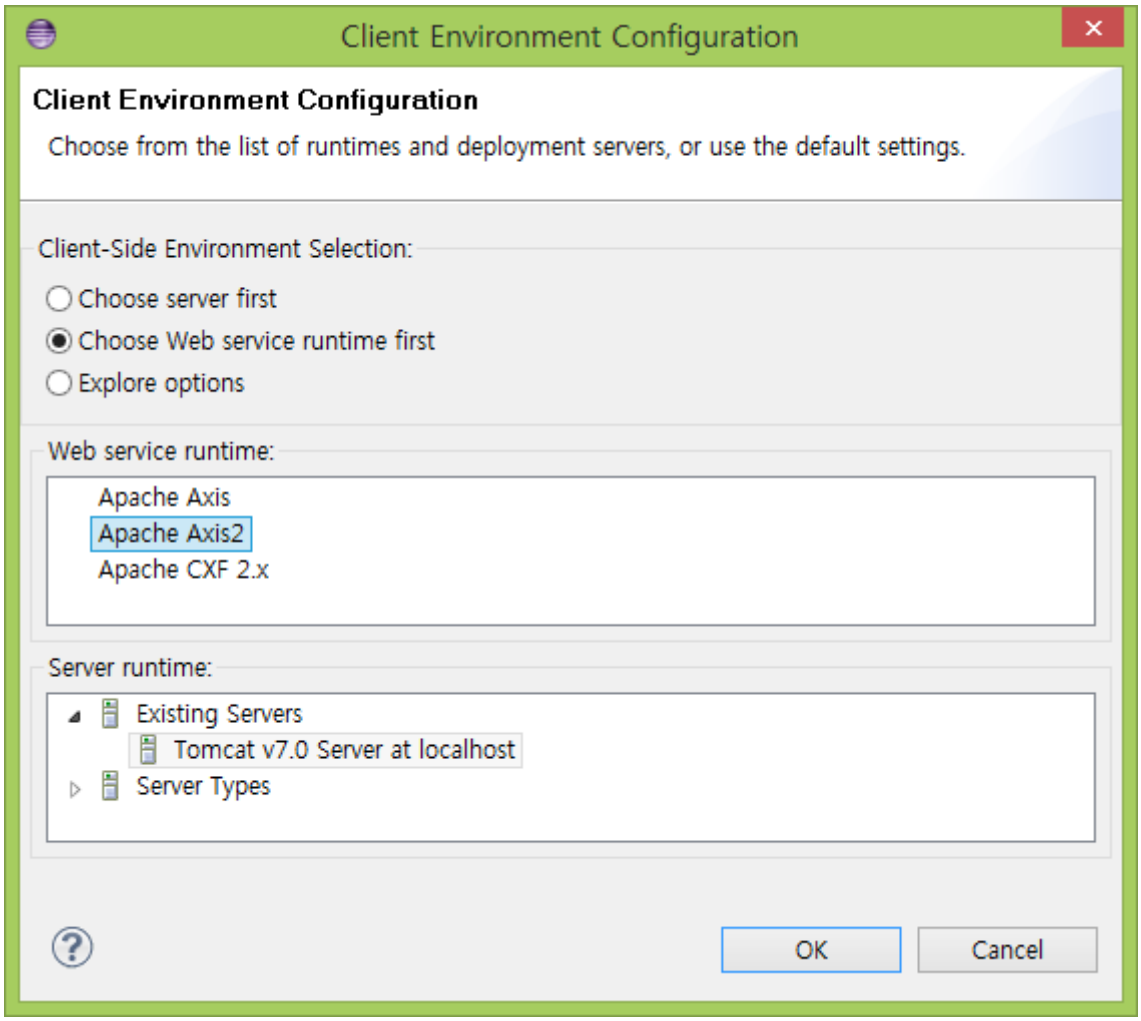


그림 7.9

7. 설정을 마쳤으면 Finish를 누릅니다.

이제 아래 그림 7.10와 같은 디렉토리 구조를 가진 프로젝트가 생성 되었습니다. m2soft.ers.report.core.webservice 패키지의 파일들은 개발자가 쉽게 web service에 요청/응답을 하기 쉽게 만들어진 래퍼 코드입니다. 우리가 실제로 사용하는 클래스는 ServiceStub입니다.

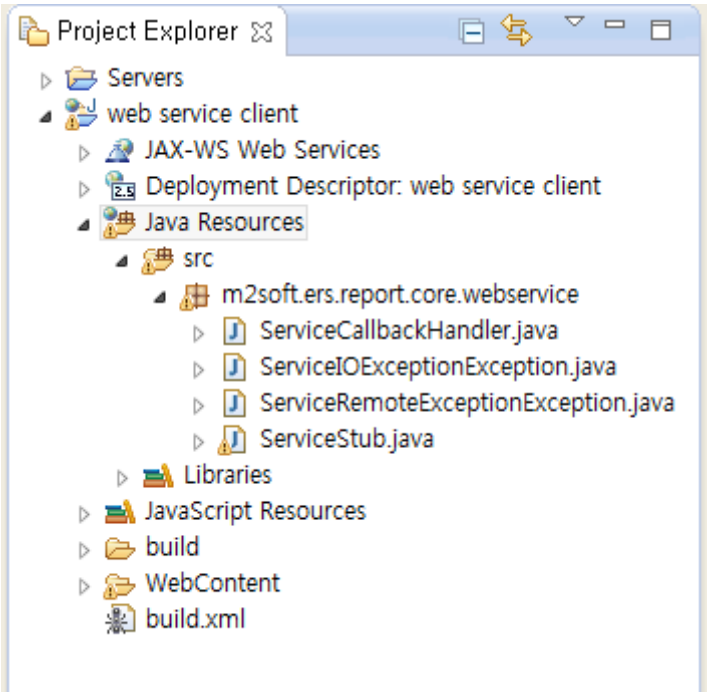


그림 7.10

8. 웹 클라이언트가 만들어졌습니다. 이제 이 클래스를 이용해 코드를 작성을 할 수 있습니다. 아래에 간단한 샘플 코드가 있습니다. 여기서는 클래스 이름이 Map2, Entry2등으로 정해져있지만 환경마다 다를 수 있으므로 반드시 생성된 소스 코드를 보고 작성해야 합니다.

```
ServiceStub stub = new ServiceStub();
Request req = new Request();
Map2 data = new Map2();

Entry2 opcode_entry = new Entry2();
opcode_entry.setKey("opcode");
opcode_entry.setValue("500");

Entry2 mrd_path_entry = new Entry2();
mrd_path_entry.setKey("mrd_path");
mrd_path_entry.setValue("sample.mrd");

Entry2 mrd_param_entry = new Entry2();
mrd_param_entry.setKey("mrd_param");
mrd_param_entry.setValue("/rfn [sample.txt]");

Entry2 protocol_entry = new Entry2();
```

```
protocol_entry.setKey("protocol");
protocol_entry.setValue("sync");

Entry2 export_type_entry = new Entry2();
export_type_entry.setKey("export_type");
export_type_entry.setValue("pdf");

// 요청할 데이터를 설정합니다.
data.addEntry(opcode_entry);
data.addEntry(mrd_path_entry);
data.addEntry(mrd_param_entry);
data.addEntry(protocol_entry);
data.addEntry(export_type_entry);

req.setParams(data);

// 요청을 하고 결과를 받아옵니다.
RequestResponse res = stub.request(req);

//결과를 출력합니다.
System.out.println(res.get_return());
```

9. 위 코드를 실행하면 다음과 같은 결과를 볼 수 있습니다. (그림 7.10)

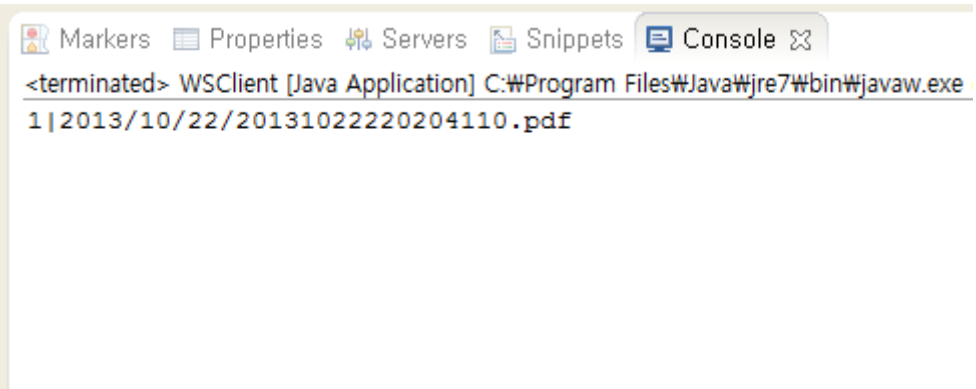


그림 7.11

2020년 8월 **제품공급자: ㈜엠투소프트**

133-120 서울시 성동구 성수동2가 333-140번지
서울숲 코오롱 디지털타워 2차 702호
TEL) 02-2188-8504 (대표) 02-2188-8500 (기술지원)
FAX) 02-2188-8508
Home) http://www.m2soft.co.kr