제 9 장 대기열 및 메시징

새 대기열 기능이 4.2에서 구현되어 여러 대기열 유형과 여러 메시징 프로토콜을 유지보수할 수 있습니다. 대기열은 EAI 플랫폼 및 웹 서버를 비롯한 외부 소스/목적지와의 인바운드 및 아웃바운드 메시징을 처리하는 게이트웨이의 역할을 담당합니다.

참고: 메시징 프레임워크 설정에 대한 정보는 "메시징 프레임워크 구현"을 계속 참조하십시오.

참조용으로 다음 용어를 정의합니다.

- **대기열** - 메시지의 수신 및 전송 지점 역할을 하는 WebSphere Product Center 구성. 스크립트는 각 대기열을 지원합니다.
- **메시지** - UCCnet, EAI 플랫폼, 데이터 풀 또는 기타 메시지 소스에서 제공하는 XML 문서

새 대기열 기능을 사용하여 메시지를 프로세스의 일부분으로 사용할 수 있으므로 작업을 가져오거나 내보낼 때 모든 필수 상대에게 상태 메시지를 보낼 수 있습니다.

대기열 콘솔

대기열 콘솔 액세스

메뉴에서 협업 관리자 > 대기열 > 대기열 콘솔을 사용하여 대기열 콘솔을 표시하십시오.

대기열 세부사항 보기

대기열의 세부사항을 보려면 대기열 이름을 클릭하여 대기열 세부사항 화면을 표시하십시오. 다음 정보가 제공됩니다.

- 대기열 이름
- 설명
- 프로토콜
- 스크립트

대기열의 메시지 보기

대기열 콘솔에는 대기열에서 받은 메시지 번호에 대한 하이퍼링크가 있는 메시지 열이 있습니다. 메시지 번호를 클릭하여 메시지 내용을 보십시오.

대기열의 메시지 검색

1. 대기열에서 메시지를 검색하려면 메뉴에서 협업 관리자 > 대기열 > 메시지 콘솔을 사용하십시오. 대기열 메시지 검색 화면이 표시됩니다.

2. 다음 필드의 값을 선택하십시오.

- 수신 도착 날짜
- 전송 도착 날짜

3. 검색을 클릭하면 아래의 대기열 메시지 테이블에 결과가 표시됩니다.

대기열 작성

1. 메뉴에서 협업 관리자 > 대기열 > 새 대기열을 사용하십시오.

2. 다음의 필수 정보를 입력하십시오.

대기열 이름: 대기열의 이름을 입력하십시오.

설명: 대기열의 설명을 입력하십시오.

프로토콜: 외부 소스를 대기열과 연결하는 데 사용되는 메시징 프로토콜 목록에서 선택하십시오.

스크립트: 대기열과의 사이에 메시지를 라우트하기 위해 실행할 수 있는 사전 작성된 스크립트 목록에서 선택하십시오. 목적지/소스의 일반 메시지는 다음과 같습니다.

- 워크플로우 단계의 내부 또는 외부 스크립트에서 지원하는 워크플로우 협업 영역
- 사전 처리, 사후 처리 또는 사후 저장 스크립트에서 지원하는 카탈로그

메시징 프레임워크 구현

WebSphere Product Center의 메시징 프레임워크 구현으로 다음 대상 EAI 플랫폼과 통합이 가능합니다.

- IBM WBI
- SeeBeyond
- Tibco
- WebMethods

원하는 플랫폼은 다른 시스템 또는 플랫폼을 통해 프로그램들이 서로 통신할 수 있도록 신뢰할 만한 전송 메커니즘과 일관된 인터페이스를 제공해야 합니다.

WebSphere Product Center의 메시징 프레임워크는 다음 프로세스를 지원하도록 설계되었습니다.

- 수신 시 수신확인 메시지 제공을 포함하여 항목 세트가 들어 있는 메시지 수신
- 전송 이후 수신확인 메시지 수신을 포함하여 항목 세트가 들어 있는

메시지 전송

WebSphere Product Center에는 XML 문서를 구문 분석 및 작성하고 EAI 플랫폼 대기열로 메시지를 전송하고 이 대기열에서 메시지를 수집하는 기능이 있습니다. 메시지는 외부 소스에서 제공하는 XML 문서로 정의됩니다. 이러한 모든 기능은 WebSphere Product Center 스크립트 엔진에서 액세스할 수 있습니다. WebSphere Product Center와 EAI 플랫폼 어댑터 간의 상호작용을 가능하게 하려면 이 기능을 사용하는 스크립트를 설치해야 합니다.

WebSphere Product Center는 메시징의 수신 및 전송 지점 역할을 담당하는 구성인 대기열을 지원합니다. WebSphere Product Center 대기열은 외부 소스/목적지와의 인바운드 및 아웃바운드 메시징을 처리하는 게이트웨이의 역할을 담당하며 각 대기열은 WebSphere Product Center 스크립트 연산에서 지원합니다.

대기열은 다음의 기능을 제공합니다.

외부 메시지 소스를 대기열과 연결하는 메시지 전송 프로토콜을 설정할 수 있도록 하며 다음 메시지 전송 프로토콜을 제공합니다.

- MQ
- JMS 지점간
- JMS 공개 및 등록
- HTTP
- HTTP/S

대기열과의 사이에 메시지를 라우트하는 스크립트를 실행합니다. 일반 메시지 대상/소스는 다음과 같습니다.

- 워크플로우 협업 영역 - 워크플로우 단계의 내부 또는 외부 스크립트에서 지원
- 카탈로그 - 사전 처리, 사후 처리 또는 사후 저장 스크립트에서 지원

다음 목록은 메시징 기능의 프레임워크를 설명합니다.

- 외부 소스는 EAI 플랫폼에 메시지를 전송합니다. 이를 처리하는 중요성은 외부 소스에 따라 다양하며 WebSphere Product Center가 메시지를 검색하는 방법에 영향을 주지 않습니다.
- EAI 플랫폼 설정에는 인바운드 및 아웃바운드 대기열이 포함되어야 합니다. 이 설정은 써드 파티에서 수행하며 WebSphere Product Center가 메시지를 검색 및 전송하기 위해 알아야 하는 한 가지 정보는 WebSphere Product Center에서 액세스하는 대기열의 ID입니다.
- 파일이 EAI 플랫폼 대기열에 놓이면 WebSphere Product Center에서 설정된 해당 대기열은 지원되는 프로토콜(MQ, HTTP/s 또는 JMS)을 통해 메시지를 검색할 수 있습니다.
- WebSphere Product Center 스크립트를 사용하면 메시지 본문을 구문 분석하여 메시지 유형, 메시지 ID 및 메시지 소스를 알 수 있습니다. 이 정보는 모든 메시지를 포함시키기 위해 작성된 대상 WebSphere Product Center 카탈로그로 라우트됩니다.
- D에서 발생한 이벤트는 워크플로우를 사용하여 추적하며 메시지가

정상적으로 기록되었으면 이벤트는 수신확인 워크플로우를 트리거하여 메시지 카탈로그에서 협업 영역 설정으로 새 레코드를 체크 아웃합니다.
- 수신확인 워크플로우는 확인을 위해 메시지 소스에 수신확인 메시지를 전송합니다. 완료했으면 수신확인 워크플로우는 카탈로그에 메시지를 다시 체크 인합니다.

이 프로세스를 작동하려면 메시징 프레임워크를 빌드해야 합니다.

메시징 프레임워크 빌드

다음 프로세스는 EAI 플랫폼 프레임워크를 통합할 때 제안하는 방법을 요약한 것입니다. 이 프로세스는 특정 요구사항에 따라 사용자 정의할 수 있습니다.

메시지 수신

이 절에서는 수신 시 수신확인 메시지 제공을 포함하여 항목 세트가 들어 있는 메시지를 수신하는 프로세스를 설명합니다. 두 개의 프로세스인 설정 및 런타임이 메시지를 수신하는 프로세스를 지원합니다. 프로세스는 일반적이며 대부분의 목적에 적용 가능합니다.

설정

1. 기술 비즈니스 프로세스 분석자는 다음을 작성합니다.

2. 기술 비즈니스 프로세스 분석자는 스크립트가 포함된 인바운드 대기열을 빌드합니다. 스크립트는 세 가지 기능(메시지 수신, 메시지 본문 구문 분석 및 라우팅)을 지원합니다.

메시지 수신

스크립트의 메시지 수신 섹션은 다음 기능을 지원합니다.

- MQ, HTTP/S 또는 JMS를 비롯한 지원되는 프로토콜을 통해 소스에서 메시지를 가져옵니다.
- 메시지 본문을 구문 분석하여 메시지 유형, 메시지 ID, 메시지 소스를 알 수 있습니다.
- 메시지 유형, 메시지 ID, 전송자 ID 및 날짜/시간이 포함된 레코드를 메시지 카탈로그에서 작성합니다.
- 수신확인 워크플로우에 대한 이벤트를 트리거합니다. 수신확인 워크플로우 기능에 대해서는 아래를 참조하십시오.

메시지 본문 구문 분석

스크립트의 메시지 본문 구문 분석 섹션은 다음 기능을 지원합니다.

- 소스 대 목적지 맵 이름 및 대상 카탈로그 이름이 매개변수로 포함됩니다.
- 항목 세트를 제공하기 위해 각 소스 대 목적지 맵 이름 및 대상 카탈로그 이름의 메시지 본문을 구문 분석합니다.

메시지 라우팅

스크립트의 라우팅 섹션은 다음 기능을 지원합니다.

대상 카탈로그에서 항목 세트의 항목을 추가/수정/삭제합니다.

- 기술 비즈니스 프로세스 분석자는 수신확인 메시지를 전송할 수 있도록 수신확인 워크플로우를 설정합니다. 워크플로우에는 다음의 기능이 있습니다.
- 수신확인 워크플로우의 단계에서 메시지 카탈로그로부터 협업 영역으로 새 레코드를 체크 아웃합니다.
- 수신확인 워크플로우의 다음 단계에서는 메시지 ID, 전송자 ID, 날짜/시간 및 메시지 소스별 필수 명령(예: 수신됨)이 포함된 수신확인 메시지를 메시지 소스로 전송합니다.
- 수신확인 워크플로우의 다음 단계에서는 메시지 카탈로그 레코드를 체크 인합니다.

런타임

설정이 올바로 구성되었으면 다음 런타임 이벤트가 발생합니다.

1. 대기열이 대기열 스크립트의 메시지 수신 섹션을 통해 메시지를 수신합니다.

2. 대기열 스크립트의 메시지 수신 섹션은 메시지 본문을 구문 분석하여 메시지 유형, 메시지 ID 및 전송자 ID를 확인합니다.

3. 대기열 스크립트의 메시지 수신 섹션은 메시지 카탈로그에서 메시지 유형, 메시지 ID, 전송자 ID 및 날짜/시간이 포함된 레코드를 작성합니다.

4. 대기열 스크립트의 메시지 수신 섹션은 수신확인 워크플로우의 단계에서 메시지 카탈로그로부터 협업 영역으로 새 레코드를 체크 아웃합니다.

5. 수신확인 워크플로우의 다음 단계에서는 메시지 ID, 전송자 ID, 날짜/시간 및 메시지 소스별 필수 명령(예: 수신됨)이 포함된 수신확인 메시지를 메시지 소스로 전송합니다.

6. 수신확인 워크플로우의 다음 단계에서는 메시지 카탈로그 레코드를 체크 인합니다.

7. 대기열 스크립트의 메시지 본문 구문 분석 섹션은 항목 세트를 제공하기 위해 각 맵 이름 및 대상의 메시지 본문을 구문 분석합니다(아래에 언급된 새 스크립트 연산 사용).

8. 대기열 스크립트의 라우팅 섹션은 카탈로그 추가/수정/삭제를 위한 기존 스크립트 연산을 사용하여 대상 카탈로그에서 항목 세트의 항목을 추가/수정/삭제합니다.

WebSphere Product Center 인바운드/아웃바운드 대기열 작성

WebSphere Product Center 인바운드 및 아웃바운드 대기열은 대기열 콘솔을 사용하여 작성됩니다. 대기열이 작성되기 전에 스크립트 콘솔에서 트리거 스크립트를 작성해야 합니다. 트리거 스크립트는 새 대기열 화면의 드롭 다운 트리거 스크립트 경로 필드에 표시됩니다.

1. 대기열 콘솔에서 **새로 작성**을 클릭하십시오.

2. 대기열 세부사항 화면에서 대기열 이름, 설명을 입력하고 프로토콜 및 트리거 스크립트 경로를 선택하십시오. 유형이 "메시지 대기열 프로세서"인 스크립트 콘솔에 트리거 스크립트가 작성됩니다.

3. **저장**을 클릭하십시오.

메시징 스크립트 연산

WebSphere Product Center 스크립트 연산은 각 스크립트 연산의 인수에 기능을 정의하여 WebSphere Product Center 스크립트 응용프로그램을 작성하는 유연한 방법을 제공합니다. 이 절에 식별된 다음 스크립트는 MQ 또는 JMS를 사용하여 WebSphere Product Center에서 지원하는 메시징 기능을 지원하는 데 사용됩니다. 이러한 방법을 통해 외부 대기열에서 메시지 가져오기 및 내보내기가 가능합니다.

참고: 이 절에 나열된 스크립트 연산은 변경될 수도 있습니다. 최신 스크립트 연산은 스크립트 샌드박스를 참조하십시오.

MQ 스크립트 연산

스크립트 응용프로그램을 작성할 때 스크립트 연산 mqGetQueueMgr은 MQQueueManager에 핸들러를 리턴하며 mqDisconnect를 호출하여 핸들을 해제하기 전에 이 핸들을 통해 여러 MQ 연산을 작성할 수 있습니다.

mqGetQueueMgr

- 프로토타입: MQQueueManager mqGetQueueMgr(String hostname, String port, String channel, String queueMgrName)
- 설명: 제공된 등록 정보로 새 MQ 대기열 관리자를 작성하고 리턴합니다.

mqDisconnect

- void MQQueueManager::mqDisconnect()
- 제공된 대기열 관리자에서 연결을 끊습니다.

mqSendTextMsg

- 프로토타입: MQMessage MQQueueManager::mqSendTextMsg(String msgText, String queueName, String queueOpenOptions, String messagePutOptions)
- 설명: queueName 위의 String msgText에 제공된 메시지를 전송합니

다. MQMessage를 리턴합니다.

참고: mqSendReply를 사용하여 제공된 메시지에 응답을 전송하려고 하면 mqSendTextMsg가 사용된 경우 오류가 리턴됩니다. 이를 방지하려면 mqSendTextMsgWithReply를 사용하십시오.

mqSendTextMsgWithReply

- 프로토타입: MQMessage MQQueueManager::mqSendTextMsgWithReply(String msgText, String queueName, String replyQueueName, String queueOpenOptions, String messagePutOptions)
- 설명: queueName 위의 String msgText에 제공된 메시지를 전송합니다. 응답 대기열이 지정됩니다. MQMessage 오브젝트를 리턴합니다.

mqGetTextFromMsg

- 프로토타입: String mqGetTextFromMsg(MQMessage mqMessage)
- 설명: 헤더를 포함하여 MQMessage의 전체 내용이 들어 있는 문자열을 리턴합니다.

mqGetReceivedMsg

- 프로토타입: MQMessage MQQueueManager::mqGetReceivedMsg(String queueName, String queueOpenOptions, String messageGetOptions)
- 설명: queueName으로부터 메시지를 수신합니다. 메시지를 MQMessage 또는 널로 리턴합니다.

  참고: 메시지를 검색하면 메시지가 대기열에서 제거됩니다. 메시지 ID를 지정하지 않으면 대기열의 첫 번째 메시지를 확보합니다.

mqSendReply

- 프로토타입: MQMessage MQQueueManager::mqSendReply (MQMessage receivedMsg, String msgText, String passedInQueueOpenOptions, String passedInMessagePutOptions)
- 설명: 성공 여부를 표시하지 않고 제공된 메시지에 응답을 전송합니다.

mqSendReplyWithStatus

- 프로토타입: MQMessage MQQueueManager::mqSendReplyWithStatus (MQMessage receivedMsg, String msgText, String status, String passedInQueueOpenOptions, String passedInMessagePutOptions)
- 설명: 제공된 상태를 표시하도록 피드백 필드를 설정하여 제공된 메시지에 응답을 전송합니다. 상태는 SUCCESS, FAIL, VALCHANGE, VALDUPES, MULTIPLE_HITS, FAIL_RETRIEVE_BY_CONTENT, BO_DOES_NOT_EXIST, UNABLE_TO_LOGIN, APP_RESPONSE_TIMEOUT, NONE 중 하나이어야 합니다(대문자 또는 소문자).

참고: 하나의 상태 값만 사용할 수 있습니다.

mqGetXMLMessageContent

- 프로토타입: String mqGetXMLMessageContent(String orgXmlMsg)
- 설명: XML 문서를 가져오기 위한 입력 문자열의 시작 부분에서 가비지를 버립니다. 보다 정확하게 말하자면 다음과 같이 활동합니다. 입력 문자열의 형식이 A + B이고 여기서 B는 유효한 XML 문서이고 A는 문자열(비어 있을 수 있음)인 경우 이 연산은 B를 리턴합니다. 그렇지 않으면 널을 리턴합니다.

  참고: 수신 메시지를 구문 분석할 때 이 방법을 사용하십시오.

mqGetResponseToMsg

- 프로토타입: MQMessage MQQueueManager::mqGetResponseToMsg (MQMessage outgoingMessage, String queueOptions, String messageOptions)
- 설명: 제공된 대기열의 제공된 메시지에서 응답을 받습니다.

mqGetMessageDiagnostics

- 프로토타입: String mqGetMessageDiagnostics(MQMessage message)
- 설명: 제공된 메시지에 대한 진단 정보가 포함된 문자열을 리턴합니다.

mqGetMessageId

- 프로토타입: String MQMessage::mqGetMessageId()
- 설명: 제공된 메시지의 ID를 16진수가 포함된 문자열로 리턴합니다.

mqGetReceivedMsgByMessageID

- 프로토타입: MQMessage MQQueueManager::mqGetReceivedMsgByMessageID(String queueName, String messageId, String passedInQueueOpenOptions, String passedInMessageGetOptions)
- 설명: 제공된 대기열에서 제공된 메시지 ID를 사용하여 메시지를 찾습니다. 이 ID는 16진수가 포함된 문자열에 전달됩니다. 제공된 대기열에 이러한 메시지가 없으면 널을 리턴합니다.

JMS 스크립트 연산

스크립트 응용프로그램을 작성할 때 스크립트 연산 jmsGetConnectionFactory는 QueueConnectionFactory에 핸들러를 리턴하며 jmsDisconnect를 호출하여 핸들을 해제하기 전에 이 핸들을 통해 여러 JMS 연산을 작성할 수 있습니다.

jmsGetContext

- 프로토타입: Context jmsGetContext(String url, String jndiFactory)

- 설명: JMS 컨텍스트를 작성합니다.

jmsGetConnectionFactory

- 프로토타입: QueueConnectionFactory Context::jmsGetConnectionFactory(String jmsFactory)
- 설명: 지정된 컨텍스트를 사용하여 jms 연결 팩토리를 작성하고 리턴합니다.

jmsGetMQConnectionFactory

- 프로토타입: QueueConnectionFactory jmsGetMQConnectionFactory (String mqQueueManager, String mqHostname, String mqChannel, Integer mqPort)
- 설명: MQ 대기열과 통신할 수 있도록 JMS 연결 팩토리를 작성하고 리턴합니다. MQ 연결 팩토리를 가져올 경우에는 컨텍스트가 필요하지 않지만 다른 JMS 대기열에 연결할 경우에는 컨텍스트가 필요하다는 점을 참고하십시오.

jmsGetQueueByName

- 프로토타입: javax.jms.Queue jmsGetQueueByName(Context ctx, String name)
- 설명: 제공된 JNDI 이름 및 컨텍스트의 javax.jms.Queue 오브젝트를 리턴합니다.

jmsGetQueueConnection

- 프로토타입: QueueConnection QueueConnectionFactory::jmsGetQueueConnection()
- 설명: 제공된 연결 팩토리의 JMS 대기열 연결을 리턴합니다.

jmsGetQueueSession

- 프로토타입: QueueSession QueueConnection::jmsGetQueueSession()
- 설명: 제공된 연결 팩토리의 JMS 대기열 연결을 리턴합니다.

jmsDisconnect

- 프로토타입: void QueueSession::jmsDisconnect(QueueConnection qcon)
- 설명: 제공된 대기열 관리자에서 연결을 끊습니다.

jmsCreateTextMsg

- 프로토타입: Message QueueSession::jmsCreateTextMsg(String msgText)
- 설명: 제공된 텍스트와 함께 QueueSession 정보를 사용하여 새 JMS TextMessage를 작성합니다.

jmsSendMsg

- 프로토타입: Message QueueSession::jmsSendMsg(Message msg, String queueName[, HashMap properties, Message messageToReplyTo])

- 설명: 이름이 queueName인 대기열을 통해 메시지 MSG를 전송하고 MSG 또는 널을 리턴합니다. MESSAGETOREPLYTO가 제공된 경우 대기열에 대한 응답 및 메시지 ID를 여기에서 읽습니다. PROPERTIES는 문자열 키에서 문자열 값으로의 맵핑입니다. 세 개의 특수 키 "WPC_REPLY_TO_QUEUE", "WPC_COPY_CORRELATION_ID_BYTES" 및 "WPC_COPY_CORRELATION_ID"가 있습니다. WPC_REPLY_TO_QUEUE가 제공된 경우 QUEUENAME 또는 제공된 MESSAGETOREPLYTO의 리플라이투 대기열을 대체합니다. MESSAGETOREPLYTO의 리플라이투 대기열은 QUEUENAME을 대체합니다. "WPC_COPY_CORRELATION_ID" 및 "WPC_COPY_CORRELATION_ID_BYTES"는 상관 ID를 MESSAGETOREPLYTO에서 MSG로 계속 복사합니다. 둘 다 제공할 수 있습니다. 해당 값은 부울이어야 합니다(위에 설명된 문자열과 반대임).

jmsReceiveMsgFromQueue

- 프로토타입: JMSMessage QueueSession::jmsReceiveMsgFromQueue (javax.jms.Queue queue, Integer timeout[, String messageSelector, JMSMessage messageToReceiveReplyFor])
- 설명: JMS 메시지를 수신합니다. TIMEOUT 밀리초 이후 제한시간에 도달합니다. INBOUNDQUEUE가 널이 아닌 경우 이 대기열을 확인합니다. INBOUNDQUEUE가 널이고 MESSAGETORECEIVEREPLYFOR가 널이 아닌 경우 MESSAGETORECEIVEREPLYFOR의 리플라이투 필드에 정의된 대기열을 확인합니다. INBOUNDQUEUE가 널이고 MESSAGETORECEIVEREPLYFOR가 널인 경우 AustinException을 발생시킵니다. 이제 사용할 대기열을 알았습니다. MESSAGESELECTOR 및 MESSAGETORECEIVEREPLYFOR가 모두 널인 경우 해당 대기열에서 첫 번째 메시지를 선택합니다. 그렇지 않으면 MESSAGESELECTOR 및 MESSAGETORECEIVEREPLYFOR에서 정의하는 모든 조건을 충족시키는 대기열(있는 경우)에서 첫 번째 메시지를 선택합니다. MESSAGETORECEIVEREPLYFOR가 널이 아닌 경우 상관 ID가 MESSAGETORECEIVEREPLYFOR의 메시지 ID와 일치하지 않는 메시지를 거부합니다. MESSAGESELECTOR가 널이 아닌 경우 messageSelector에 정의된 조건을 충족시키지 않는 메시지를 거부합니다. 적합한 메시지를 찾을 수 없는 경우 널을 리턴합니다.

jmsGetTextFromMsg

- 프로토타입: String Message::jmsGetTextFromMsg()
- 설명: 헤더를 포함하여 JMS 메시지의 전체 내용이 들어 있는 문자열을 리턴합니다.

jmsGetMessageID

- 프로토타입: String Message::getJMSMessageID()
- 설명: JMS 메시지 ID가 들어 있는 문자열을 리턴합니다.

jmsGetMessageCorrelationID

- 프로토타입: String Message::getJMSMessageCorrelationID()
- 설명: JMS 메시지에 대한 상관 ID가 들어 있는 문자열을 리턴합니다.

jmsGetMessageProperties

- 프로토타입: HashMap Message::getJMSMessageProperties()
- 설명: 문자열 등록 정보 이름에서 해당 우선순위의 문자열 값으로의 해시맵을 리턴합니다.

jmsSendMsgToQueue

- 프로토타입: JMSMessage QueueSession::jmsSendMsgToQueue (JMSMessage msg, javax.jms.Queue outboundQueue [, HashMap properties, JMSMessage messageToReplyTo,])
- 설명: 메시지 MSG를 전송하고 MSG 또는 널을 리턴합니다. OUTBOUNDQUEUE가 널이 아니면 OUTBOUNDQUEUE에서 지정하는 대기열로 메시지를 전송합니다. OUTBOUNDQUEUE가 널인 경우 MESSAGETOREPLYTO가 제공되었으면 MESSAGETOREPLYTO의 리플라이투 큐로 MSG를 전송합니다. OUTBOUNDQUEUE가 널이고 MESSAGETOREPLYTO가 제공되지 않은 경우 AustinException을 발생시킵니다. MESSAGETOREPLYTO가 제공된 경우 여기에서 메시지 ID를 읽습니다. PROPERTIES는 문자열 키에서 문자열 값으로의 맵핑입니다. 하나의 특수(비JMS) 키 WPC_INCOMING_REPLY_QUEUE가 있습니다. WPC_INCOMING_REPLY_QUEUE는 외부 응용프로그램이 이 메시지에 대한 응답을 전송해야 하는 javax.jms.Queue 오브젝트를 표시합니다.

jmsSetMessageText

- 프로토타입: void Message::setJMSMessageText(String msgText)
- 설명: JMS TextMessage에 대해 제공된 텍스트를 설정합니다. JMS TextMessage 유형만이 지원됩니다.

웹 서비스

새 웹 서비스 작성

협업 관리자 > **웹 서비스** > **새 웹 서비스** 메뉴 경로를 사용하십시오. "웹 서비스 세부사항" 화면이 표시됩니다.

해당 정보를 다음 필드에 입력하십시오.

웹 서비스 이름	웹 서비스 이름을 입력하십시오. 이 이름은 SOAP 서비스의 URL 일부가 됩니다. 이름에는 공백이 포함될 수 없습니다. 예:
웹 서비	웹 서비스 설명을 입력하십시오.

스 설명	
프로토콜	웹 서비스에 사용하는 프로토콜. 현재 SOAP over HTTP가 유일하게 지원되는 프로토콜입니다. 기본값은 "SOAP_HTTP"입니다.
URL	서비스에 액세스할 수 있는 URL을 제공합니다. 이 필드는 웹 서비스를 저장한 후에 자동으로 채워집니다.
WSDL URL	웹 서비스의 WSDL에 액세스할 수 있는 URL. 이 필드는 웹 서비스를 저장한 후에 자동으로 채워집니다.
WSDL	이 서비스의 WSDL을 입력하십시오. WSDL 문서는 인터페이스, URL 및 서비스 프로토콜을 XML 형식으로 설명합니다. 이 문서를 수동으로 입력해야 하지만 당사는 아래 WSDL 문서 샘플을 제공합니다. 웹 서비스가 저장되도록 올바른 XML을 입력해야 합니다.
구현 스크립트	이 서비스를 구현하는 Trigo 스크립트를 입력하십시오. 서비스의 수신 매개변수를 배열 변수 "soapParams"에서 사용할 수있습니다. 서비스의 리턴 값은 String이 되어야 하며 "out" 작성기 변수에 기록될 수 있습니다. SOAP 결함을 리턴하려면 결함 코드를 "soapFaultCode" 작성기 변수에 기록하고 결함 메시지를 "soapFaultMsg" 작성기 변수에 기록하십시오. 구현 스크립트 샘플이 아래 제공되어 있습니다.
요청 저장?	이 항목을 선택하면 Trigo가 모든 수신 요청의 매개변수를 docstore에 저장합니다. 이 매개변수는 트랜잭션 콘솔에서 사용할 수 있습니다.
응답 저장?	이 항목을 선택하면 Trigo가 모든 응답 내용을 docstore에 저장합니다. 이 내용은 트랜잭션 콘솔에서 사용할 수 있습니다.
전개됨	이 항목을 선택하면 서비스가 전개됩니다. 그렇지 않으면 이 서비스를 사용할 수 없습니다.

구현 스크립트 및 WSDL 문서 샘플

다음 설계 방식의 구현 스크립트 및 WSDL 문서 샘플은 수신 SOAP 요청의 매개변수 수를 검사합니다. 매개변수가 정확히 네 개인 경우 이 매개변수를 나열하는 문자열을 리턴합니다. 매개변수가 네 개를 초과하거나 네 개 미만인 경우 SOAP 결함이 발생합니다.

구현 스크립트

```
nParams = soapParams.size();
if (nParams != 4)
  {
    soapFaultCode.writeln("WRONG_NUM_PARAMS");
    soapFaultMsg.writeln("Wrong number of parameters.  This service require
  }
else
  {
    out.writeln("Success.");
    for (i = 0; i < nParams; i++)
    {
      out.writeln("Parameter " + (i + 1) + " is " + soapParams[i]);
    }
  }
```

WSDL

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<wsdl:definitions targetNamespace="http://my.trigo-instance.com/soap/servic
                  xmlns="http://schemas.xmlsoap.org/wsdl/"
                  xmlns:apachesoap="http://xml.apache.org/xml-soap"
                  xmlns:impl="http://my.trigo-instance.com/soap/services/Ch
                  xmlns:intf="http://my.trigo-instance.com/soap/services/Ch
                  xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
                  xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
                  xmlns:wsdlsoap="http://schemas.xmlsoap.org/wsdl/soap/"
                  xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <wsdl:message name="invokeRequest">
    <wsdl:part name="param1" type="xsd:string"/>
    <wsdl:part name="param2" type="xsd:string"/>
    <wsdl:part name="param3" type="xsd:string"/>
    <wsdl:part name="param4" type="xsd:string"/>
  </wsdl:message>
  <wsdl:message name="invokeResponse">
    <wsdl:part name="invokeReturn" type="xsd:string"/>
  </wsdl:message>
  <wsdl:portType name="CheckParams">
    <wsdl:operation name="invoke" parameterOrder="param1 param2 param3 para
      <wsdl:input message="intf:invokeRequest" name="invokeRequest"/>
      <wsdl:output message="intf:invokeResponse" name="invokeResponse"/>
    </wsdl:operation>
  </wsdl:portType>
  <wsdl:binding name="CheckParamsSoapBinding" type="intf:CheckParams">
    <wsdlsoap:binding style="rpc" transport="http://schemas.xmlsoap.org/soa
    <wsdl:operation name="invoke">
      <wsdlsoap:operation soapAction=""/>
      <wsdl:input name="invokeRequest">
        <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encod
                       namespace="http://DefaultNamespace"
                       use="encoded"/>
      </wsdl:input>
      <wsdl:output name="invokeResponse">
        <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encod
                       namespace="http://my.trigo-instance.com/soap/service
                       use="encoded"/>
      </wsdl:output>
    </wsdl:operation>
  </wsdl:binding>
  <wsdl:service name="CheckParamsService">
    <wsdl:port binding="intf:CheckParamsSoapBinding"
               name="CheckParams">
      <wsdlsoap:address location="http://my.trigo-instance.com/soap/service
    </wsdl:port>
  </wsdl:service>
</wsdl:definitions>
```