Legacy Connectivity notes.

General.
Legacy connectivity refers to how Java components interact with a legacy system.
Typical properties of a legacy system are :

- developed in another programming language
- runs in an environment that doesn't support Java
- limited connectivity – e.g. a mainframe that only supports physical connections

Connectivity Options.
Generally, components can interact with software/services on a legacy system in the following ways :

- *in-process* – e.g. independent software processes collaborating
- *out-of-process (physical connection)* – e.g. serial IO over local cable / leased-line
- *out-of-process (virtual connection)* – e.g. over the network; serial IO over dialup; etc.
- *out-of-process (broker)* – e.g. via a middle-man such as an ORB, MQ, etc.

Sample Scenarios.

- XML News feed.
  News stories are stored in a MS-SQL server database. Servlets/JSPs (or the Java XML pack) and JDBC are used to render the stories as XML.

- Off-board server.
  A Java Socket server is produced to enable secure remote access to a mainframe by forwarding SSL requests sent on sockets to serial connections on the mainframe. Responses are transmitted back to the client via SSL. *N.B. this architecture would also be appropriate if "screen scraping" was the only connectivity option available.*

- Order processing.
  A component manufacturer wishes to sell it's products via a website. The existing order fulfilment system uses IBM MQ Series. Servlets/JSPs and JMS are used to produce an order request message from the website. The fulfilment system picks up website orders by consuming website order requests. When the order has been completed, the order fulfilment system produce an "order complete" event and publishes it to the messaging system. A MessageDrivenBean subscribes to the "order complete" event and notifies the user via email using JavaMail.

- Component reseller.
  An in-house CORBA development team require a complex content management system. An off-the-shelf system is available which uses EJB. As EJB uses JNDI and RMI-IIOP, the in-house team are able to integrate the content management system with their CORBA components.

- Installation program.
  A Java product has a platform specific installation program written in another programming language. E.g. a Windows specific installer "sniffs" the configuration details from the Windows registry and writes it to file; the Java product picks up the configuration on first run and configures the Java product accordingly.

- Credit card authorisation.
  A Java utility class authorizes credit cards via a JNI wrapper to a platform specific X25 IO card.

- Single sign-on.
  An organisation uses a directory product to store all relevant user details. The organisation insists that any web application must use the directory for authentication. The website developers use Servlets and JNDI to delegate the authorisation to the directory product.

Integration Options.

Low-level integration

Description	***Pros / Cons***
Custom protocols (over Java Sockets / Communications API)	***Pros:*** easy to develop; works well ***Cons:*** Java IO performance; impedance mismatch with Java binary streams and Java character encoding
File IO to a common specification (files are "local" - on the same machine or mapped network drive)	***Pros:*** simple and quick to develop ***Cons:*** clunky
Custom protocols (over HTTP, a.k.a. HTTP tunnelling)	***Pros:*** simple and quick to develop ***Cons:*** stateless; non-transactional
Screen scraping (over Java Sockets / Communication API)	***Pros:*** gain online access to mainframe ***Cons:*** extremely brittle, UI change == interface failure

Custom bridge.

Description	***Pros / Cons***
Java wrapper (e.g. Java socket server to mainframe serial IO; Java RMI with JNI wrapper)	***Pros:*** opens up access to other Java components ***Cons:*** requires an experienced developer; tedious coding
Windows bridge (e.g. MS Java SDK, Jawin)	***Pros:*** opens up Windows functionality ***Cons:*** non-portable code
Protocol bridge (e.g. COM/CORBA bridge)	***Pros:*** provides remote access to existing Windows code ***Cons:*** performance; won't necessarily pass through firewalls so only suitable for intranet use

Java standard protocols / APIs (***9 available***).

Description	***Pros / Cons***
JDBC (Java Database Connectivity)	***Pros:*** allows generic access to databases from any vendor
JNI (Java Native Interface)	***Pros:*** enables a Java wrapper to be layered around platform specific code ***Cons:*** clunky; non-portable code
Java Servlets / JSPs (Java Server Pages)	***Pros:*** enables fast and easy development of services for HTTP based clients
JMS (Java Message Service)	***Pros:*** messaging systems are available on a whole host of legacy platforms and JMS enables access to these from Java
RMI-IIOP / Java IDL (use RMI-IIOP to program to an RMI interface but allow access by CORBA clients; use IDL if your focus is CORBA with Java)	***Pros:*** enables Java/EJB and CORBA to interoperate ***Cons:*** RMI-IIOP loses some of the normal features of RMI (e.g. stub download, distributed garbage collection)
JNDI (Java Naming and Directory Interface)	***Pros:*** provides generic access to a host of naming/directory services (e.g. LDAP, COS Naming, Novell, DNS, file system, Windows registry)
Java XML Pack	***Pros:*** provides an XML toolkit to enable enhanced productivity
JCA (J2EE Connector Architecture)	***Pros:*** provides generic access to any EIS component ***Cons:*** early days
JDO (Java Data Objects)	***Pros:*** transparent object persistence ***Cons:*** early days

Open standards.

Description	***Pros / Cons***
IIOP (Internet Inter-ORB Protocol).	***Pros:*** enables components written in other languages / on different platforms to interoperate ***Cons:*** doesn't go through firewalls
SOAP (Simple Object Access Protocol) / XML	***Pros:*** similar to IIOP but uses XML over HTTP so does go through firewalls ***Cons:*** large overhead; uses non-transactional protocol for transport