© Copyright 2004, Sun Microsystems, Inc.

Exception Handling in J2EE Systems

Stephen A. Stelting
Sun Services
Sun Microsystems, Inc.

© Copyright 2004, Sun Microsystems, Inc.

Objectives and Payoff

Objectives:

- Explain the exception models of key J2EE APIs
- Present best practices for global exception handling in J2EE systems
- Describe challenges of J2EE exception handling

Payoff:

By the end of this presentation, you'll be familiar with the exception models of each tier. You'll also have an idea of how to use exceptions in a full J2EE application.

© Copyright 2004, Sun Microsystems, Inc.

Why Is J2EE Exception Handling Hard?

- J2EE code is run within different containers, possibly on different servers
- Component exception models are not mutually compatible
- You can't use a blanket exception handling strategy... nor would you want to!

J2EE systems require developers to be more savvy about an application's exception model

© Copyright 2004, Sun Microsystems, Inc.

How Do You Use Exceptions in J2EE?

The key to effectively managing exceptions in J2EE lies in understanding:

1) What exceptions exist for each tier
2) How, when and by whom they can be produced
3) How they are used and what the result of an exception will be
4) How they affect the overall J2EE application

© Copyright 2004, Sun Microsystems, Inc.

The J2EE Exception Model

JSP

<% %>

Servlet

SB

EB

MDB

Client Tier

Client-specific errors

Web Tier **EJB Tier**

Adapt the EIS error model

Manage application exceptions

Convert exceptions for the Client tier

EIS Tier

System-specific errors

© Copyright 2004, Sun Microsystems, Inc.

Web Tier – Exception Model

- There are two types of errors in the Web Tier:
  1) **HTTP Errors (400-500 series response)**

     Communicate problems to Web clients
  2) **Java Exceptions**

     Communicate problems to the Web container
- Exceptions can be generated within the tier or passed back from other tiers
- If you want exceptions to be passed on to a client, you must convert them to HTTP errors

© Copyright 2004, Sun Microsystems, Inc.

Types of Web Components

Servlet specification:

- Servlets
- Filters
- Listeners

JSP specification:

- JSPs
- Tag Handlers (custom tag libraries)

© Copyright 2004, Sun Microsystems, Inc.

Exceptions for the Web Tier

Servlet specification:

- javax.servlet.ServletException
- javax.servlet.UnavailableException

JSP specification:

- javax.servlet.jsp.JspException
- javax.servlet.jsp.JspTagException (JSP2.0)
- javax.servlet.jsp.SkipPageException (JSP2.0)
- javax.servlet.jsp.el.ELException (JSP2.0)
- javax.servlet.jsp.el.ELParseException (JSP2.0)

Lifecycle of Servlets and Filters

javax.servlet.Filter:

```
public void init(FilterConfig c) throws ServletException

public void doFilter(ServletRequest req,
        ServletRespose rsp, FilterChain ch)
        throws java.io.IOException, ServletException

public void destroy()
```

javax.servlet.Servlet:

```
public void init(ServletConfig c) throws ServletException

public void service(ServletRequest req, ServletResponse rsp)
        throws java.io.IOException, ServletException

public void destroy()
```

© Copyright 2004, Sun Microsystems, Inc.

Servlet and Filter: Initialization

- The init method
  - Called by the container to “set up” the component
  - Must complete successfully before the component can be used
- Exceptions
  - Servlets and Filters can throw a ServletException or UnavailableException
  - Signals the container that initialization has failed

© Copyright 2004, Sun Microsystems, Inc.

Initialization: Exceptions

- ServletException – Container releases the object; it can immediately try to create a new Web component
- UnavailableException (no wait time) – Same as ServletException
- UnavailableException (wait time) – Container releases the object; it must wait for the specified time before creating a new component

© Copyright 2004, Sun Microsystems, Inc.

- The destroy method is never called, since initialization did not complete
- Client calls during component unavailability trigger an HTTP 500 error response
- Unchecked exceptions are wrapped in a ServletException

© Copyright 2004, Sun Microsystems, Inc.

Servlet and Filter: Service Method

- The service, do*<u>Zzz</u>*, or doFilter methods
  - Called to handle a client request
  - Run by worker threads in the container
- Exceptions
  - Servlets and Filters can throw java.io.IOException, ServletException or UnavailableException
  - Signals the container that the component cannot complete this method normally, or that it can no longer handle **<u>any</u>** client requests

© Copyright 2004, Sun Microsystems, Inc.

Service Method: Exceptions

- IOException or ServletException – Problem with this request only; the container will "clean up"
- UnavailableException (no wait time) – Component cannot process any requests; container calls the destroy method and removes the component
- UnavailableException (wait time) – Component cannot process any requests for the wait time; container can either:
  - Return an HTTP 503 error until the wait time is over
  - Call destroy and permanently remove the component

© Copyright 2004, Sun Microsystems, Inc.

Service Method: Container Response

- If an individual request fails, the server returns an HTTP 500 error
- If a component is temporarily unavailable, the server returns an HTTP 503 error during the wait period
- If a component is permanently unavailable, the server returns an HTTP 404 error
- Unchecked exceptions will cause a request to fail; the server returns an HTTP 500 error
- If output has already been committed, HTTP errors will not appear, but CAN truncate the response!

© Copyright 2004, Sun Microsystems, Inc.

Servlet and Filter: Destroy Method

- The destroy method
  - Called by the container to "clean up" the component before it is destroyed
  - Typically called as a result of server or application shutdown by the administrator
- Exceptions
  - Neither component declares checked exceptions in this method
  - **Tomcat**: Unchecked exceptions are wrapped in a ServletException and written to the server log

© Copyright 2004, Sun Microsystems, Inc.

Servlet Event Handlers

Listeners called at specific times for a Web app:

ServletContext listeners

- ServletContextAttributeListener: Called when ServletContext attributes are modified
- ServletContextListener: Called for Web application startup and shutdown

ServletRequest listeners

- ServletRequestAttributeListener: Called when ServletRequest attributes are modified
- ServletRequestListener: Called when a ServletRequest is created or destroyed

© Copyright 2004, Sun Microsystems, Inc.

Servlet Event Handlers *(Continued)*

HttpSession listeners

- HttpSessionActivationListener: Called during session activation/passivation
- HttpSessionAttributeListener: Called when HttpSession attributes are modified
- HttpSessionBindingListener: Called when values are bound to or unbound from an HttpSession
- HttpSessionListener: Called when an HttpSession is created or destroyed

© Copyright 2004, Sun Microsystems, Inc.

Event Handlers and Exceptions

- None of the listeners declare exceptions
- Unchecked exceptions can be sent to an error page using Deployment Descriptor entries
- If there is no error page mapping, the container generates an HTTP 500 error; no more listeners will be called for the event

Some unchecked exceptions can halt a Web app

Example: An unchecked exception of a ServletContextListener can prevent a Web application from being initialized

Deployment Descriptor error-page

The web.xml file allows you to configure global handling:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE web-app PUBLIC "-//Sun Microsystems,
      Inc.//DTD Web Application 2.3//EN"
      "http://java.sun.com/dtd/web-app_2_3.dtd">
<web-app>
  <!-- Servlet definitions, mappings, etc. -->
  <error-page>
    <exception-type>
        java.lang.NullPointerException
    </exception-type>
    <location>/err/nullPtr.jsp</location>
  </error-page>
  <error-page>
    <error-code>404</error-code>
    <location>/err/notFound.html</location>
  </error-page>
</web-app>
```

© Copyright 2004, Sun Microsystems, Inc.

Deployment Descriptor *(Continued)*

For error page mappings, the container will:

- Pass the request and response objects to the handler
- Disable the setStatus method
- Set standard attributes

  javax.servlet.error.status_code

  javax.servlet.error.exception_type

  javax.servlet.error.message

  javax.servlet.error.exception

  javax.servlet.error.request_uri

  javax.servlet.error.servlet_name

Colorado Software Summit

© Copyright 2004, Sun Microsystems, Inc.

Container exception handling process:

1) Check to see if the exception matches an error-page
   - Forward if there's a match
2) Check to see if the wrapped exception of a ServletException matches an error-page
   - Forward if there's a match
3) Use default handling (HTTP 500 for service method)

- Error pages can be invoked by sendError for HTTP 400 and 500-series response codes
- Not used for RequestDispatcher or doFilter calls

© Copyright 2004, Sun Microsystems, Inc.

JavaServer Pages

- Because JSPs are dynamically generated, errors can occur during:
  - Translation
  - Compilation
  - Runtime
- Default handling is "built into" a JSP
- Can specify an handler for the JSP with the page directive

© Copyright 2004, Sun Microsystems, Inc.

JSP Translation and Compile Errors

Translation errors

- Caused by incorrect use of the JSP specification
- Generates an HTTP 500 response with info about the location of the mistake

Compilation errors

- Caused by incorrect Java code in the JSP
- Generates an HTTP 500 response with a compiler error in the stack trace

- Tomcat: For both errors, more detailed information is available in the server logs

© Copyright 2004, Sun Microsystems, Inc.

JavaServer Pages: Structure

JSPs have a built-in handling structure:

```
public void _jspService(HttpServletRequest req, HttpServletResponse rsp)
    throws java.io.IOException, ServletException {
  /* ++ PLACEHOLDER FOR VARIABLE DEFINITIONS  ++ */
  try {
    /* ++ CONVERTED JSP WOULD GO HERE  ++ */
  } catch (Throwable t) {
   if (!(t instanceof javax.servlet.jsp.SkipPageException)){
     out = _jspx_out;
     if (out != null && out.getBufferSize() != 0)
       out.clearBuffer();
     if (pageContext != null) pageContext.handlePageException(t);
   }
  } finally {
   if (_jspxFactory != null) _jspxFactory.releasePageContext(pageContext);
  }
}
```

© Copyright 2004, Sun Microsystems, Inc.

JSP Runtime Handling Structure

- Any Throwable is passed to a global catch block
- For every Throwable except SkipPageException, the JSP:
  1) Clears its output buffer
  2) Calls the PageContext method handlePageException
  3) Releases its PageContext (finally block)
- The PageContext handler method forwards to:
  - Its error page, if one is defined
  - The Web container's handler, if no error page is defined

© Copyright 2004, Sun Microsystems, Inc.

JSP Error Pages

- Defined in the page directive:

```
<%@ page errorPage="err/handleErr.jsp" %>
<jsp:directive.page errorPage="err/HandleErr" />
```

- The PageContext will preferentially forward Throwables to the errorPage

© Copyright 2004, Sun Microsystems, Inc.

JSP Expression Language (EL)

- New to JSP 2.0
- Allows you to use a theoretically simple expression language in your JSPs
- Exceptions are used as part of handling framework when EL is interpreted
- “It is expected that many JSP containers will use additional mechanisms to parse EL expressions and report their errors…” (JSP.14.2)

© Copyright 2004, Sun Microsystems, Inc.

JSP Tag Handlers

- Defined by interfaces in javax.servlet.jsp.tagext
- Standard Methods
  - Tag: doStartTag, doEndTag
  - BodyTag: doInitBody
  - SimpleTag: doTag
  - IterationTag: doAfterBody
- Every method declares the JspException
- SimpleTag's doTag method also declares IOException, SkipPageException

© Copyright 2004, Sun Microsystems, Inc.

Tag Handlers *(Continued)*

- Exceptions are propagated to the calling JSP
- SkipPageException is only thrown by a simple tag handler to halt evaluation of a page
- Tag handlers can implement the TryCatchFinally interface to deal with exceptions:
  - doCatch (declares Throwable)
  - doFinally

© Copyright 2004, Sun Microsystems, Inc.

The EJB Tier

- **Very** complex exception model
- Results from the fact that EJBs have:
  - Different calling options (local *vs*. remote)
  - Complicated lifecycle (an EJB is not always the same as the data it represents)
  - Complex container services (caching, transactions)
- It's useful to think of the different exceptions in groups or categories

© Copyright 2004, Sun Microsystems, Inc.

Application Exceptions

The EJB specification defines application and system exceptions

Application Exceptions:

- Defined in the throws clause of an EJB's interface
- Does not include java.rmi.RemoteException
- Signals clients of unacceptable application-level conditions
- Clients can usually recover from these exceptions

© Copyright 2004, Sun Microsystems, Inc.

System Exceptions

System exceptions:

- Represent problems that prevent successful method completion
- Can occur because the exception or error is:
  - Unexpected
  - Expected, but the container doesn't know how to recover
- A RemoteException can be used to send a system exception to a client

© Copyright 2004, Sun Microsystems, Inc.

EJB Clients – Communication

- Session beans and entity beans can have local and remote communication interfaces
- Exceptions indicate
  - Communication problems (RemoteException)
  - System exceptions (represented in a standard way)

Local EJBs: Methods in the interfaces declare javax.ejb.EJBException (a RuntimeException)

Remote EJBs: Methods in the interfaces declare java.rmi.RemoteException

© Copyright 2004, Sun Microsystems, Inc.

Session Bean: Application Exceptions

Application exceptions for Session Beans

- Lifecycle exceptions
  - create: CreateException
  - remove: RemoveException
- Business exceptions
  - Business methods

Note: The create and remove methods are **not** directly called by clients for Stateless SBs!

© Copyright 2004, Sun Microsystems, Inc.

Entity Bean: Application Exceptions

Application exceptions for Entity Beans

- Lifecycle exceptions
  - create: CreateException, DuplicateKeyException
  - remove: RemoveException
  - finder methods: FinderException, ObjectNotFoundException
- Business exceptions
  - Business, home and finder methods

© Copyright 2004, Sun Microsystems, Inc.

Impact of Exceptions on Client

- Session Beans

  Problem: A client could call a method on a bean that no longer exists (through a timeout or error)

  Result: The container throws

  - NoSuchObjectException (remote)
  - NoSuchObjectLocalException (local, runtime)

- Entity Beans

  Problem: An entity bean could be removed by a client, then called by another client

  Result: The container throws NoSuchEntityException (runtime)

Server-side Exceptions

Bean provider responsibilities:

- Application Exceptions
  - Define exceptions for the Bean and its interfaces
  - Throw for appropriate problems in business logic
  - Ensure that the instance is in an appropriate state
  - Mark the transaction for rollback
- System Exceptions
  - Throw for system-level exceptions or errors
  - Understand the impact of exceptions on your code

© Copyright 2004, Sun Microsystems, Inc.

Application Exceptions

Business exceptions

- Session Beans, Entity Beans
  - Business methods
  - Entity bean only: finder and home methods
- Message-driven Beans: some messaging types

Session Beans: Lifecycle exceptions

- ejbCreate: CreateException
- ejbRemove: RemoveException

© Copyright 2004, Sun Microsystems, Inc.

Entity Beans: Lifecycle Exceptions

- ejbCreate = SQL INSERT

  CreateException, DuplicateKeyException

- ejbRemove = SQL DELETE

  RemoveException

- findBy*Zzz* = SQL SELECT (primary key)

  FinderException, ObjectNotFoundException

© Copyright 2004, Sun Microsystems, Inc.

Container Behavior for Exceptions

Application exceptions

- Re-throw the exception to the client (or the resource adapter for a MDB)

System exceptions

Client-called:

- Log the exception
- Discard the instance
- Throw EJBException (local) or RemoteException (remote)

Container-called:

- Log the exception
- Discard the instance

© Copyright 2004, Sun Microsystems, Inc.

EJB Exceptions and Transactions

- By itself, the EJB exception model is complex
- Transactions further increase this complexity
- New considerations:
  - CMT *vs*. BMT
  - Local *vs*. remote invocation
  - Caller's transaction *vs*. EJB-owned transaction
  - Exceptions due to mismatched transaction model
- In addition, there's a difference between rollback and transaction-related exceptions!

EIS Tier Exceptions

- The EIS tier represents enterprise resources accessed by the J2EE application... DBMS, ERP systems, legacy applications
- As such, there needs to be some way of adapting errors for use by the J2EE system
- This is managed by the technologies used to communicate with the EIS tier – Connectors

© Copyright 2004, Sun Microsystems, Inc.

EIS Tier – Connectors

- Connector APIs provide a way to manage distributed communication with EIS resources
- Can reside at any tier – Client, Web or EJB (although EJB tier is most common)
- Principal technologies:
  - JDBC, The Java Database Connectivity API
  - Java Messaging Service (JMS)
  - J2EE Connector Architecture

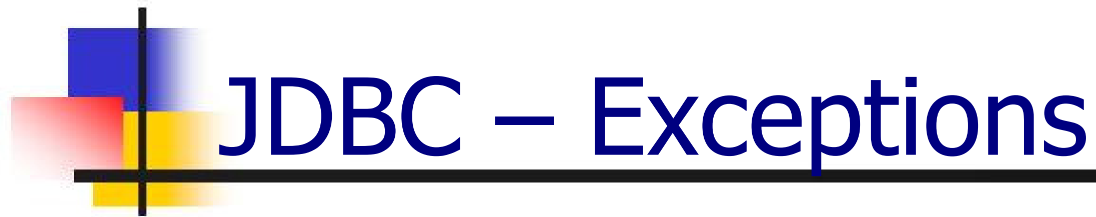
© Copyright 2004, Sun Microsystems, Inc.

Exceptions in Connectors

- Connector architectures are based on:
  - Interface-based API
  - Plug-in adapters (SPI)
  - Adapters for a resource (possibly)
- There are lots of failure points for communication:

  Caller ⇔ API ⇔ Adapter ⇔ Channel ⇔ Resource
- Many connectors define a “base” exception class
- API/Adapter is responsible for creating the exception

© Copyright 2004, Sun Microsystems, Inc.



JDBC – Exceptions

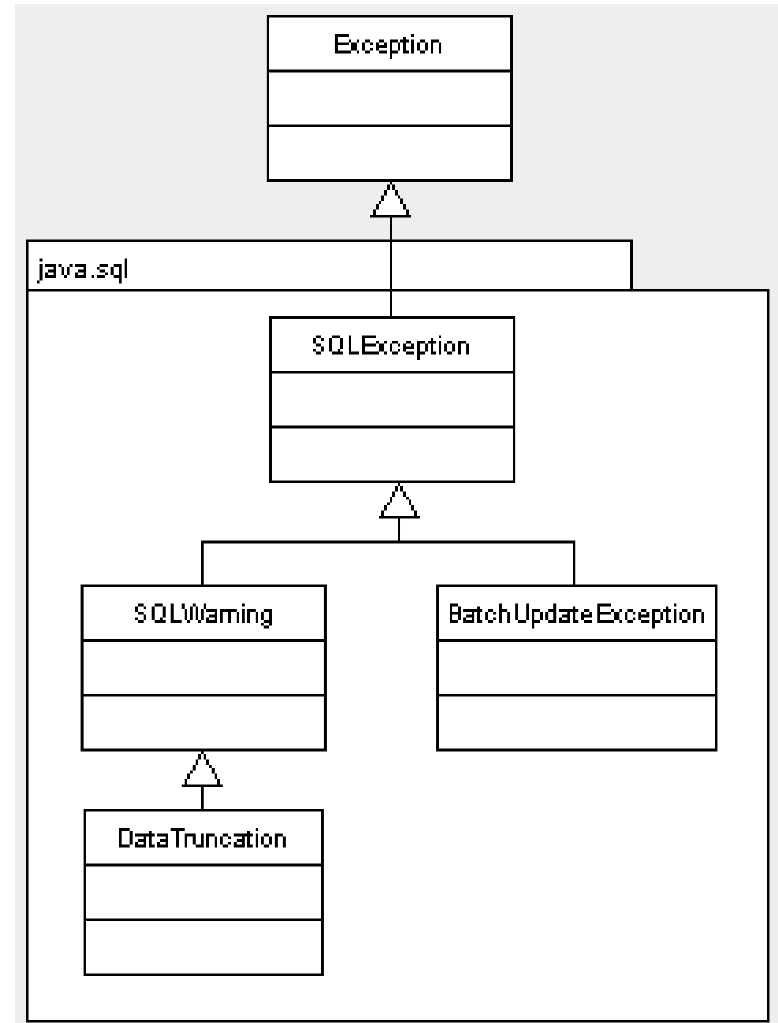
- JDBC enables communication with DBMS systems (usually relational)
- Base exception class: java.sql.SQLException
- Can be thrown at any time from getConnection to the close method call
- In addition to standard exception behavior, supports error code, SQL state (XOPEN/SQL99), exception chaining

© Copyright 2004, Sun Microsystems, Inc.

JDBC – Exceptions

Other exception types:

- SQLWarning
  Database warnings
- DataTruncation
  Truncation of data during
  DBMS reads or writes
- BatchUpdateException
  Errors during batch operations



Colorado Software Summit

© Copyright 2004, Sun Microsystems, Inc.

JMS – Exceptions

- Java Message Service (JMS) supports communication with messaging systems
- Primary exception: javax.jms.JMSException
- Like JDBC, supports exception chaining
- JMSException (or subclasses) can be thrown throughout the use of the API – from creating a connection to closing it at end of lifecycle

© Copyright 2004, Sun Microsystems, Inc.

J2EE Connector Architecture

- Provides communication with enterprise systems (DBMS, ERP, legacy apps)
- Exceptions:
  - javax.resource.ResourceException
  - javax.resource.NotSupportedException
  - javax.resource.cci.ResourceWarning
- Like the previous APIs, methods throw the ResourceException throughout the use of the API to access the enterprise resource

© Copyright 2004, Sun Microsystems, Inc.

J2EE – Special Considerations

Beyond the exception-producing methods in the APIs, there are some standard considerations for a few of the tiers:

- EIS Integration Tier: Connection management
- EJB Tier: Object and data cache management
- Web Tier: I/O, threading

© Copyright 2004, Sun Microsystems, Inc.

J2EE – Standard Challenges

EIS Integration Tier:

- Need for exception handling and conversion
  - Especially evident when exceptions are a catch-all (JDBC) or defined by a flat exception model (JTA)

EJB Tier:

- RemoteException (declared for EJB1.0 specification)
- Container-specific semantics of methods with an unspecified transaction context (ref. 17.6.5)

Web Tier:

- Exception handling/conversion for callers

© Copyright 2004, Sun Microsystems, Inc.

J2EE – Standard Issues

Issues related to exceptions include:

- Communication overhead between containers
- Support for different kinds of clients
- Implications of unchecked exceptions
- Application recovery from a server crash

Colorado Software Summit

© Copyright 2004, Sun Microsystems, Inc.

J2EE – Validation

A good practice is to validate information as early as possible in the flow of J2EE communication to conserve bandwidth and reduce exceptions:

- Client tier: field-level
- Web tier: object-level, business operations
- EJB tier: entities/components, business sequence
- EIS tier: “strict” validation (data across multiple caller address spaces)

© Copyright 2004, Sun Microsystems, Inc.

J2EE Exception Handling: EIS

- Connectors: Since they can occur anywhere within the J2EE system, a generalized handling strategy is often preferred
  - Log problems
  - Convert to a general "resource not available" exception
  - Report problems to the clients, or let the consumers query for EIS resource availability

Filter for incorrect/inaccurate data before sending to EIS

© Copyright 2004, Sun Microsystems, Inc.

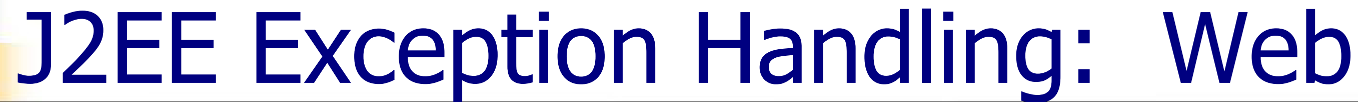
J2EE Exception Handling: EJB

- EJB Tier: Different approaches tend to be preferred based on the type of problem
  - Resource-based errors & communication problems: log problem, report unavailability

    Consider secondary caching if possible
  - Lifecycle errors: log and report, possibly attempt to "reinitialize" EJBs if possible (local references)
  - Business errors: report problem to caller

Client should check for EJB resource availability after prolonged inactivity

© Copyright 2004, Sun Microsystems, Inc.



J2EE Exception Handling: Web

- Web Tier:
  - Lifecycle problems: determine whether the cause is global or local
    - Global: Log and report "resource" unavailability through ServletException
    - Local: Report problem to caller (consider logging)
  - "Web-ize" format of exceptions for easier client use

© Copyright 2004, Sun Microsystems, Inc.

Summary

In this session, we have:

- Discussed exceptions in the principal J2EE APIs
  - Web Tier
  - EJB Tier
  - EIS Integration Tier
- Presented common challenges in dealing with exceptions
- Described best practices for exception handling in J2EE

© Copyright 2004, Sun Microsystems, Inc.

Questions

? ?

© Copyright 2004, Sun Microsystems, Inc.

For More Information...

- More information is available in my book, "Robust Java" [Sep. 2004, Prentice Hall]
- Also contains
  - exception handling best practices
  - testing strategies
  - lots of other good stuff
- Companion website:
  http://www.talk-about-tech.com