学生应知信息知识

Java知识手册（八）

狄登峰 主编

目　　录

J2EE 创建多媒体管理软件解决方案

新的事务模型的目的是逐渐使最终用户能够用标准化的和常见的软件管理、检索和操作存储的多媒体资源--例如相片、视频和行情资料。在利用现有的内部技术来降低成本和产生利润的时候，现有的媒体业务还用来实现访问它们的多媒体资源的标准化值。尽管在过去几年中存储量、处理能力和软件都有重大的发展，但是管理数字媒体资源仍然是一件代价相当高的事情。一些研究表明，大多数的多媒体文件是非结构化的资源；只有很少一部分存在于关系数据库和现有的应用程序中。结构化的缺乏使有效地访问和重新利用数字资源变得非常困难。

中间件平台--特别是应用程序服务器--总是处理数据资源的操作。在创建多媒体增强应用程序的过程中使用应用服务器好像是对这种技术固有强度的一种自然延伸。然而，和数字资源相关的大小、结构和元数据的基本的差异使你需要采用与 J2EE 平台创建的关系数据库和已有资源不同的方式来操作。本文将从现在可用的和正在开发这两个角度来探讨创建多媒体应用程序所需要的标准和技术。我还将讨论在存储、索引、访问和检索多媒体资源的过程中 J2EE 所起到的作用，以期把这个平台的领域扩展到数字资源领域。最后，我还将探讨 J2EE 平台必须解决的问题，以使用户可以最优化地使用多媒体资源。

三个特性区分和定义了一个多媒体资源。在多媒体资源和已有的相关数据之间最大的基本差别是媒体文件的大小。虽然压缩技术正在不断地改善，但是复杂的视频或者音频数据流仍然动辄以千兆字节计。虽然现在已经有了存储和管理极大数据流的数字内容管理系统，但是没有用于访问这些保存的资源的标准化应用程序编程接口或者机制。

还可以从结构上来区分多媒体资源和其他数据。一般来说，你可以把传统资源作为单独的组件来访问和使用。但是多媒体资源可能包含若干个元素，例如视频流、音频流、相关的字幕信息和其他数据集。维护这个结构是数字资源管理系统的一个基本要求。

最后，多媒体文件通常由二进制信息组成。因此，传统的查询、索引和检索文件的方法不适用于多媒体领域。为了应用程序能够成功地管理、检索并且操作一个多媒体组件，你必须维护数字资源和描述这种资源特征的元数据信息之间的关系。

诸如 JDBC 和 JCA 这样的现有的 J2EE 平台规范阐明了用于数据访问的协议，你可以遵循这些协议创建一个基于标准方法的程序来检索多媒体资源。新的标准还必须进一步增强定义的 J2EE 组件模型的多媒体能力。

获得多媒体和中间件平台之间最佳组合的方法主要在于你如何定义一个用于访问数字资源的存储抽象层。为了保持应用程序移植性，你必须利用或者扩展现有的标准来解决数字媒体存储特性，比如插入、更新或者查

询资源。

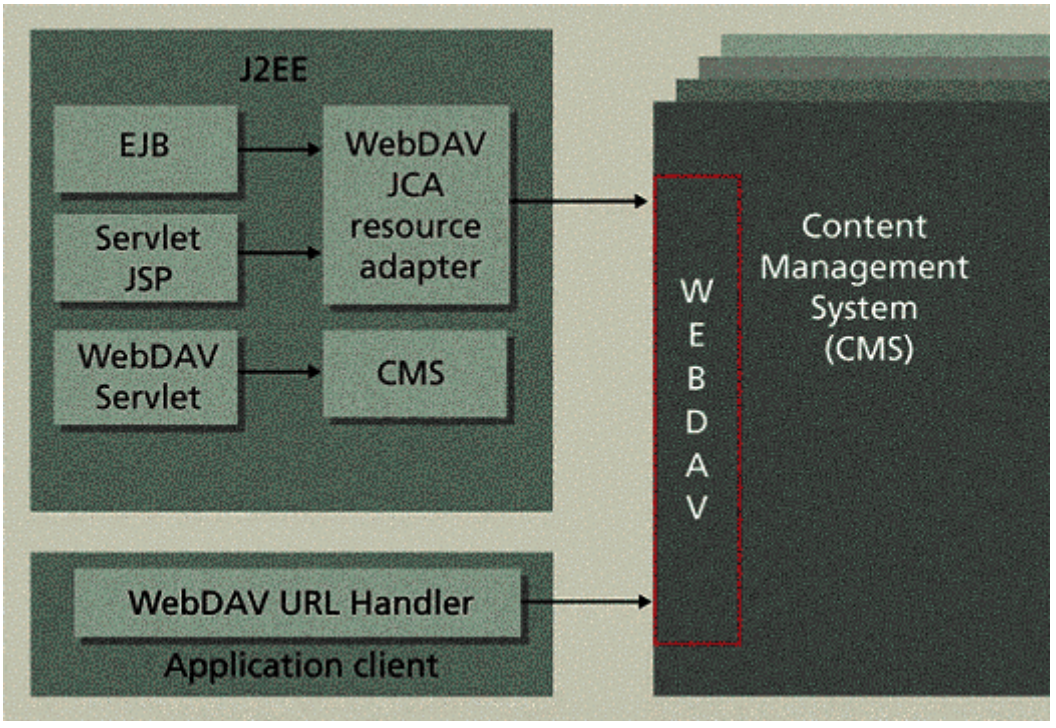

图1.定义一个存储抽象

WebDAV 规范是一个对 HTTP 进行扩展的协议，用于解决数字媒体存储大小、结构和元数据这三个方面（见图１）。它提供了跨 Internet 协议的分布式编辑和版本控制的能力，可以和现有的 HTTP 客户端交互操作。WebDAV 被使用在网络存储解决方案和 Web 服务器、许多编辑工具（包括微软公司出品的 Internet Explorer 浏览器、Apache Slide 客户端、Apple OS X Finder、Microsoft Office、和 Adobe 应用程序）和大部分的操作系统中。许多解决多媒体存储的内容管理产品支持 WebDAV。例如 Apache Slide 体系机构使用 WebDAV 作为客户端访问协议。Slide 提供一个抽象层，允许对机制类型的选择用

于所有它的存储，包括内容和元数据。这把内存存储、数据库存储、基于 XML 的存储等考虑进去。

惠普多媒体平台和 Apache Slide 工程利用 WebDAV 协议和所提供的关联的客户机和服务器应用程序编程接口来创建数字存储抽象功能。这种解决方案提供一个使用规格化、标准化和简单方式访问多后端内容管理程序的方法。这些平台提供了像 WebDAV servlet 这样的 Web 组件让开发者和任何 WebDAV 服务器接口，把许多 WebDAV 服务器整合到一个联合内容服务器中，或者创建基于请求信息的自定义解决方案。你可以使用 HP WebDAV servlet 和可以截取 WebDAV 请求和在存储和检索操作期间执行数据处理的 servlet 过滤器同时使用。有用的操作包括元数据和内容的提取、变换或者索引。

通过利用标准化 J2EE 组件，你可以创建一个可伸缩和容错的基于中间件的内容管理系统。例如，你可以联合 WebDAV servlet、相关的处理过滤器和 Apache Slide 来生成一个内容服务器，能够存储文件、这些文件附属的元数据属性和基于元数据属性的文件的搜索。这样一个系统在 J2EE 应用程序体系结构平台上执行，并且可以使用平台的性能、可伸缩性、安全和可移植性等特性。

客户端的存储器接口还可以利用 J2SE 和 J2EE 这两个版本的属性和设备。因为 URL 设置被构建进 J2SE 平台中，你可以在 Java 虚拟机中安装一个 WebDAV 协议处理程序来简化到 WebDAV 内容管理系统的客户接口。J2EE 组件可以潜在地利用 JCA 连接器实现来创建存储企业组件和应用程序。例如 HP 多媒体平台的

WebDAV 连接器访问遵从 WebDAV 协议的服务器作为企业资源：

```
ConnectionSpec spec;

ConnectionFactory factory;

WebDAVConnection conn;

factory
=(ConnectionFactory)ctx.lookup("java:comp/env/webdav/local" );

spec = new WebDAVConnectionSpec("/", "username",
"password" );

conn =
(WebDAVConnection)connectionFactory.getConnection

( spec );
```

发现和访问

一旦你已经创建一个基于标准 Java 的机制来用于存储和检索多媒体资源，你需要一个查询和发现存储的文件的方法来创建增强的多媒体应用程序。元数据的关键用途是改善信息的查询和检索。元数据实质上是关于信息的信息；它提供或者访问关于另一个信息资源的信息。在多媒体的上下文中，元数据简化了发现的访问数字内容的过程。

各种元数据标准分别在某些信息领域解决不同的问

题。Dublin Core 元数据标准被开发来提供一个描述文档，象 HTML 文档、PDF 文件和图像这样的资源的标准方法。它已经被扩展，现在库、档案和联机内容的发行者使用 Dublin Core 作为一种通用的元数据标准。

Dublin Core 标准描述了适用于在很宽的资源范围内的描述性元素的小集合。这些元素包括象标题、创建者、主题、日期、格式和语言等属性。即使还没有元数据表达机制被普遍接受,但是 Dublin Core 项目已经采用了资源描述框架(RDF)。RDF 提供了一个描述和交换元数据的方法。这那些框架支持元数据与支持用于语义、语法和结构的标准协定机制的互操作性。RDF 不强制用于不同资源描述共同体的语义。取而代之的是，它提供了用于这些团体来根据需要定义新的元数据元素的方法。RDF 通常使用 XML 作为一种元数据交换和处理的机制。XML 的使用促进了元数据元素集合之间的互操作性，以及在完全不同的团体之间的扩展名和元数据的再使用。此外，RDF 简化了词汇的发布，不仅能使词汇被人阅读而且可以很容易地被应用程序处理。

列表 1：

```
＜?xml version=’1.0’?＞

＜rdf:RDF xmlns:rdf=’http://www.w3.org/1999/

02/22-rdf-syntax-ns#’

xmlns:rdfs= ’http://www.w3.org/2000/01/
```

```
rdf-schema#’
xmlns:hp=’http://www.hpmiddleware.com/xml/
namespaces/metadata-java-1.0#’>
<rdf:Property rdf:about="http://
www.hpmiddleware.com/rdf/maiden/1.0#name">
<rdfs:isDefinedBy rdf:resource="http://
www.hpmiddleware.com/rdf/maiden/1.0#" />
<hp:datatype>java.lang.String</hp:datatype>
</rdf:Property>
<rdf:Property rdf:about="http://
www.hpmiddleware.com/rdf/maiden/1.0#title">
<rdfs:isDefinedBy rdf:resource="http://
www.hpmiddleware.com/rdf/maiden/1.0#" />
<hp:datatype>java.lang.String</hp:datatype>
</rdf:Property>
</rdf:RDF>
```

列表 1 是描述两个元数据属性的 RDF 文档示例：name 和 title。＜rdf:Property＞节点定义了元数据资源。rdf:about 属性标识了元数据资源。(你总是通过一个 URI 描述一种资源。)＜rdfs:isDefinedBy＞元素指出定义了相关元数据资源的资源。这两个元素共同描述了元数据资源的语义。在列表 1 中?maiden/1.0#域名空间定义了 name 元数据属性。＜hp:datatype＞元素定义了元数据资源的本地 Java 类型。在列表 1 中，name 元数据资源的本地 Java 类型是 java.lang.String。

许多容器模型已经被提出来用于访问各种可能存在的用于描述日益不同的资源的元数据集合。这样一个提议是 Warwick Framework。Warwick Framework 提出了一个容器体系机构的概念模型，在其中你可以部署多个元数据集合，例如你已经习惯于用于创建企业应用程序来使用地现有的应用程序组件。这个框架的特定实现必须提供一个用于处理容器和它的元数据包的实际方法。

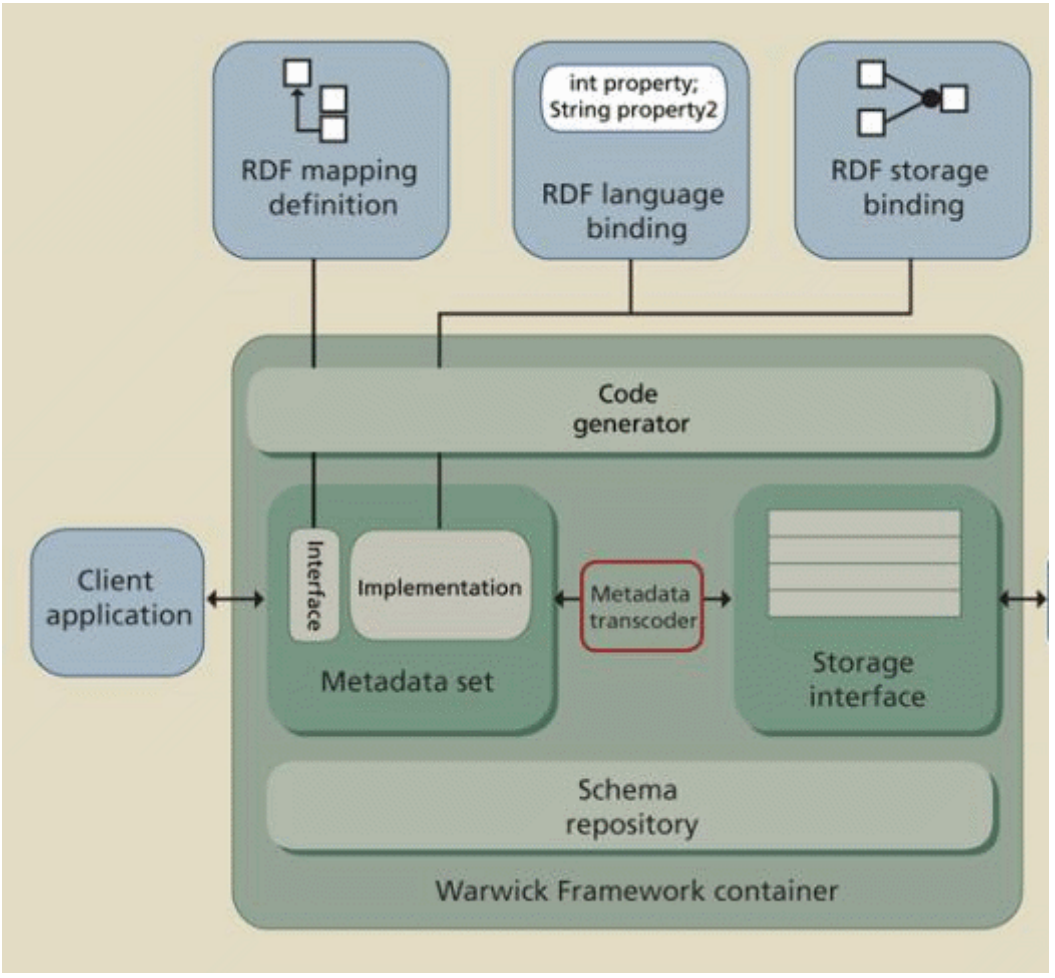

图 2、实现 Warwick Framework 容器

图 2 显示了 Warwick Framework 容器的典型的 Java 实现。这个框架使用了一种普通的方法描述了元数据集合中的属性之间的关系，这样你就可以部署你可以在不同平台上实现的不同容器的描述。此外，你使用一种本地类型绑定描述（特别是对于一种程序设计语言）来生成实现绑定的代码。一个存储绑定允许在多个元数据集合中的属性来映射到存储的单一值（canonical 属性）。

存储绑定的一部分定义了所需的代码转换来转换存储中编码的属性值为用于指定的元数据集合的固有的编码。

一个客户应用程序使用 Warwick Framework 实现来调用容器自己、保存属性值的对象和部署到容器上的任何元数据集合。一个 Java 名称和目录接口(JNDI)查找可以获得到容器的引用。一个配置的存储驱动程序管理属性值的持续性。一个低级的存储层提供了元数据属性的本体类型绑定。面向客户端的元数据集合应用编程接口不仅使用合适的本地类型而且使用用于元数据集合的合适的编码来传递属性值。

列表 2：

```
package metadataset;

import com.hp.mw.richmedia.metadata.*;

import com.hp.mw.richmedia.metadata.spi.TranscodingMetadataProper

public interface MaidenAppSample extends

   com.hp.mw.richmedia.metadata.MetadataSet {

public static final TranscodingMetadataProperty

   NAME_METADATA_PROPERTY =

new
TranscodingMetadataProperty("http://www.hpmiddleware.com/rdf/maiden/1
```

public static final TranscodingMetadataProperty

TITLE_METADATA_PROPERTY =

new
TranscodingMetadataProperty("http://www.hpmiddleware.com/rdf/maiden/1

public void addName(java.lang.String name) throws MetadataExcepti

public PropertyValueCollection getNameCollection() throws Metadatal

public void setName(java.lang.String name)
CardinalityConstraintException,

MetadataException;

public java.lang.String getFirstName() throws MetadataException;

public void addTitle(java.lang.String title) throws MetadataExcepti

public PropertyValueCollection getTitleCollection() throws Metadata

public void setTitle(java.lang.String title
CardinalityConstraintException,

MetadataException;

public java.lang.String getFirstTitle() throws MetadataException;

}

列表 2 说明了一个 Warwick Framework 生成的面向客户端的应用编程接口的示例。列表 3 中的 JSP 代码说明了你如何通过 JNDI 查找、检索元数据集合对象和查询用于 name 元数据属性的值的对象来访问 Warwick Framework。

列表 3：

```
<%@ page ?>

<% Hashtable env = new Hashtable();

env.put( Context.URL_PKG_PREFIXES,

"com.hp.mw.richmedia" );

InitialContext ctx = new InitialContext( env );

MetadataContainer container =

( MetadataContainer ) ctx.lookup(

"metadata:/metadata-container.xml" );

MetadataQuery query = container.createQuery(

new URIQuerySpec( "http://

www.hpmiddleware.com/maiden/1.0/

testresource" ) );
```

```
Collection c = container.query( query );

Iterator itor = c.iterator();

MetaObject mo = ( MetaObject ) itor.next();

MaidenAppSample sampleMetadataSet =

( MaidenAppSample )

container.getMetadataSet( mo,

"http://www.hpmiddleware.com/rdf/

maidenapp/sample/1.0#" );

PropertyValueCollection pvc =

sampleMetadataSet.getNameCollection();

out.println( "<br><h3>Values for the name

property?</h3><br>" );

itor = pvc.iterator();

while ( itor.hasNext() )

out.println( ( String ) itor.next() + "<br>" =;

%>
```

我已经细节说明的存储组件和元数据组件简化了基于标准、多媒体增强的应用程序的创建。这些组件提供了使用你熟悉 J2EE 开发机制来存储、查询和检索多媒体元素的能力。因此，你不仅可以把现有的和新的 J2EE 应用程序中的上下文的多媒体内容提供给最终用户，而且可以利用这个平台来创建支持新媒体组件的生成的应用程序。

例如，你可以通过集合媒体组件到同步多媒体集成语言(Synchronized Multimedia Integration Language，SMIL)来组装一个促进媒体表现得交互创建的基于 Web 的应用程序。(SMIL 是一个提供根据时间线计划音频、视频、文本和图形文件的标注语言。)另一个例子是一个简化提取、审查和最终发布多媒体内容的门户。这两个应用程序可以通过利用标准 J2EE 组件,例如 JSP、servlets 和 Enterprise Java Beans 来在多媒体生命周期中确定关键阶段。

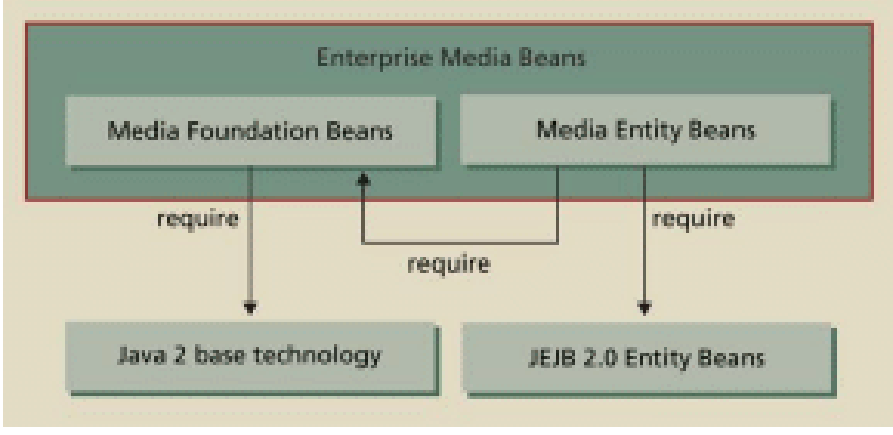

图3.EMB 组件和从属物

除了利用现有的组件之外，Java 开发者团体还在探索如何扩展现有的 Java 平台的范围来在多媒体资源生成中担任一个更重要的角色。JCP 的许多创新能解决现有的 Java 平台中的一些多媒体处理的不足。

Enterprise Media Bean

Enterprise Media Bean (EMB) Java 规范请求解决了多媒体数据的操作和应用。EMB 规范工作推出了一个基于组件的体系机构，用于把多媒体数据无缝集成到基于 J2EE 程序设计模型的应用程序中。（参见图３）。这个规范提出了两个组件类型。第一个组件 Media Foundation Bean 提供了一些基本的服务（比如头标的提取）来提供一个标准和轻量级方法来在 J2EE 应用程序中包含基本的多媒体相关特性。相反，第二组件 Media Entity Bean 依靠 EJB 实体 bean 并且因此用于基于 EJB 体系结构的应用程序。这些 Media Entity Bean 基本上是持久的、远程的和易变的模型数据的实体 bean。因此，它们扩展了标准 EJB 事务和安全模型到媒体数据中，提供了一个服务器端协议处理程序（例如流服务器）的抽象。

此外，JCP 调查访问内容存储的标准方法。现在，客户应用程序必须使用一个供应商的专有的应用编程接口来与特定的内容容器相互作用。Content Repository for Java Technology 应用编程接口将集中于事务读/写访问二进制内容（流操作）、文本内容、全文搜索、过滤、监视、版本控制和处理结构化内容。

除现有的 JCP 创新来扩展 J2EE 平台之外，补充的