Федеральное государственное бюджетное образовательное учреждение высшего образования
«РОССИЙСКАЯ АКАДЕМИЯ НАРОДНОГО ХОЗЯЙСТВА И ГОСУДАРСТВЕННОЙ СЛУЖБЫ ПРИ ПРЕЗИДЕНТЕ РОССИЙСКОЙ ФЕДЕРАЦИИ»

НИЖЕГОРОДСКИЙ ИНСТИТУТ УПРАВЛЕНИЯ – филиал РАНХиГС

Факультет управления

Кафедра информатики и информационных технологий
Специальность: 09.04.03 Прикладная информатика

Специализация: Информационные технологии в государственном и муниципальном управлении

Лабораторная работа № 1. Статическая и динамическая информационная модель. Основные конструкции языка XML.

Цель работы:

Изучение принципов построения статической и динамической информационной модели. Изучение основных конструкций языка XML и правил оформления XML-документов.

Результат:

Синтаксически правильные XML-документы для выбранной предметной области.

Задание для **самостоятельного** выполнения

Представьте в виде древовидной структуры данные выбранной по желанию предметной области. Примерами предметных областей могут служить, например, «Издательство» (выпускающее газеты и журналы), «Научная конференция», «Кафедра вуза», «Составление букетов», «Цех производства маш.деталей», «Школа», «Больница», «Налоговая инспекция», «Гараж», «Пожарная часть», «Частная клиника», «Магазин», «Администрация города» и «Войсковая часть». Студенты могут сами выбрать свои темы практической, с согласия преподавателя данной дисциплины.

Создайте XML-документ, использовав информационную модель, построенную в первой части лабораторной работы.

Методические указания

к *первой* части лабораторной работы

Моделирование данных и XML

Успех любого приложения XML зависит от того, насколько хорошо спроектированы фактически используемые документы XML: они должны быть способны не только нести информацию, которую люди передают друг другу сегодня, но и обладать достаточной гибкостью, чтобы учитывать будущие требования. Для этого необходимо рассмотреть следующие аспекты процесса проектирования:

Моделирование информации (понимание структуры и назначения информации, содержащейся в документах);

Проектирование документа (трансляция информационной модели в набор правил или схем для создания фактических документов);

Нотации схем (методы записи проекта документа, чтобы он был доступен как для обрабатывающего его программного обеспечения, так и для пользователя-человека).

Информационная модель — это описание используемой организацией информации, не зависящее от какой бы то ни было информационной технологии и определение таких моментов, как:

- каким образом она структурирована;
- что она означает;
- кому она принадлежит, и кто отвечает за ее своевременность и качество;
- откуда она берется и что происходит с ней в конце.

Моделирование информации имеет большое значение, потому что без модели нет информации, есть только данные. Информационная модель описывает назначение данных.

Любое информационное моделирование преследует две цели, которые не всегда бывает легко сочетать:

- *Получение абсолютно точных определений*
- *Эффективная коммуникация с пользователями*

Существуют два основных типа информационной модели: статическая и динамическая.

Статическая информационная модель

В *статической* модели описываются допустимые состояния системы, типы объектов в системе, их свойства и связи (вместе с этим определяются их имена). Достичь соглашения по именованию всех объектов очень важно, именно поэтому информационные модели XML иногда называют словарями.

При построении статической информационной модели необходимо пройти следующие этапы:

Этап 1. Идентификация понятий, присвоение им имен и их определение

Этап 2. Организация понятий в иерархию классов

Этап 3. Определение связей, множественности и ограничений

Этап 4. Добавление свойств для конкретизации деталей значений, связанных с объектами

Именование понятий

Для начала нужно составить список понятий, относящихся к системе. Иногда предлагают описать систему на бумаге и выделить все существительные. В любом случае этот этап, как правило, не представляет затруднений.

Далее, и это иногда требует больше времени, надо описать типы объектов. Описание термина должно быть точным, чтобы не возникало разногласий по существу определения. Ценность моделирования в том и заключается, что оно предотвращает появление потенциальных источников непонимания.

По завершении работы, вероятно, получится длинный список типов объектов, и у некоторых будут длинные имена. Следует выбирать такие имена, которые занятые в этом бизнесе люди смогут корректно понять и интерпретировать: дело в том, что они не всегда будут сверяться с написанными определениями.

Помимо именования типов объектов стоит определить также, каким образом можно идентифицировать индивидуальные экземпляры. Возможно, в этом виде бизнеса существует код, который следует знать, или написать

его. Надо принимать во внимание, что в схемах кодирования в бизнесе часто обнаруживаются проблемы.

В конце этого этапа мы получим список типов тех объектов с именами и определениями, для которых достигнуто соглашение.

Таксономия

Таксономия — это термин, используемый в биологии для обозначения системы классификации; в информационном моделировании ее также называют иерархией типов (иногда ее называют еще онтологией). Перечислив и назвав типы объектов, их надо организовать в иерархическую схему классификации. Часто эти иерархические отношения возникают уже на этапе определения типов объектов.

Ключевой здесь является фраза, определяющая принадлежность (в английском языке — is или is kind of). Написав предложение вида "А есть разновидность В" или "Каждое А есть В", вы определили отношения подтипов в вашей таксономии.

Иногда эти действия называются тестом "is a" ("есть", "представляет собой"). Однако будьте внимательны, поскольку эта конструкция используется также и для описания отношений между конкретным экземпляром и его типом, безопаснее писать этот тест в форме "is a kind of" ("представляет собой разновидность").

Идентификация подтипов бывает, полезна при проектировании документа, что более важно, она помогает лучше понять определения типов объектов.

Если вы занимались объектно-ориентированным программированием, то понимаете, как определять иерархии типов. Но программисты часто рассматривают классы объектов, прежде всего в терминах модулей функциональности внутри системы, а не понятий, которые они представляют в окружающем мире. Тогда для обозначения типов объектов используются глаголы, а не существительные - что неверно.

Итак, этап 2 сводится к организации типов объектов в иерархию типов.

Поиск связей

После того как объекты названы, в статическом информационном моделировании надо определить *связи*, существующие между ними. Связи (на языке UML они называются ассоциациями) их можно показать, просто сформулировав их в виде обычных предложений или их можно показать графически в виде диаграммы. Для диаграмм, описывающих связи между объектами, существует большое количество нотаций, каждый может выбрать предпочтительную для себя. Диаграммы следует делать предельно простыми и интуитивно понятными, сосредоточившись на ключевых сообщениях и оставив подробности текстовым документам, которые проще обслуживать.

Существует некая информация, которую надо знать о каждой связи:

Множественность связи показывает, сколько объектов каждого типа принимает в ней участие.

Связи типа “один-ко-многим”: одна глава содержит много параграфов, один человек покупает много туристических поездок.

Связи типа “многие-ко-многим” также часто встречаются: один автор может написать несколько книг, но у книги также может быть несколько авторов.

Связи типа “один-к-одному”.

При моделировании информации для окончательного представления XML особенно важным типом связей являются связи включения. Множественность этих связей всегда бывает “один-ко-многим” и “один-к-одному”. Хотя четкого правила по поводу того, какие объекты образуют связи включения, не существует, можно иногда использовать правила обычного языка: глава содержит параграфы, курорт содержит отели, а отель содержит посетителей. В языке UML определено две формы связей включения. Первая — это *агрегации*, относительно свободное объединение объектов, позволяющее рассматривать их группу в течение некоторого времени как целое (например, туристическая группа, одни и те же люди могут в разное время входить в разные группы). Вторая форма — *композиция*. Это более строгая форма. Отдельные части целого не могут существовать независимо от него (например, комнаты в отеле не могут существовать независимо от отеля).

Найти подходящие имена для связей бывает нелегко. При записи связей полезно использовать полные фразы типа “отель расположен на курорте” или “человек — это автор книги”. Нам не надо использовать эти имена в тегах разметки XML, они присутствуют только в документации на систему.

Итак, мы определили связи, существующие между типами объектов в нашей модели.

Описание свойств

Типы объектов и связи формируют скелет статической информационной модели, свойства можно сравнить с плотью на костях. Свойства представляют собой простые значения, ассоциированные с объектами. У человека можно определить рост, вес, национальность и род занятий; отель имеет определенное количество комнат, этажность и ценовую категорию.

Не следует снова включать связи в список свойств объекта: расположение отеля не является его свойством, если мы уже промоделировали его как связь с курортом.

Главное, что нам надо знать о свойствах, - это тип их данных. Определен ли для них фиксированный диапазон значений, являются ли они числовыми, в каких единицах выражаются? Является ли свойство обязательным и есть ли у него значение по умолчанию?

В конце этапа 4 мы завершили формирование статической информационной модели: получили полное описание типов объектов в системе, их связей друг с другом и их свойств.

Динамическая информационная модель

Динамические модели описывают, что происходит с информацией: примерами таких моделей являются диаграммы рабочих процессов, потоков данных и жизненных циклов объектов. Динамические модели состоят примерно из таких утверждений: “Отделение патологии отправит результаты теста консультанту, отвечающему за пациента”. Динамические модели описывают процесс обмена информацией: данные отправляются из одного места в другое с конкретной целью.

Для построения динамической модели можно воспользоваться различными программными системами (например, диаграммы рабочих процессов и диаграммы потоков данных можно построить, используя программу BPWin).

Модели рабочих процессов

Модели рабочих процессов заостряют внимание на роли людей и организаций в выполнении работы, хранение и обработка информации играют в них вторичную роль. Модель процесса описывает, например, что будет с путешественником, если на отдыхе с ним произойдет несчастный случай. Она определяет, за какие действия отвечает локальный представитель на курорте, агент в стране, где находи курорт и главный офис. В результате становится ясно, кто должен отвечать за организацию медицинской помощи, за перевозку туриста домой и за информирование родственников. Она может описывать различные формы, заполняемые и пересылаемые между участниками, и вообще не привлекать компьютерные системы. Модель процесса обычно фокусирует внимание на ролях, обязанностях и задачах каждого действующего лица системы (actor), а workflow-модель имеет дело с документами, передаваемыми между действующими лицами. Модели потоков данных.

Модели потоков данных

Модели потоков данных очень напоминают предыдущий тип, но здесь основное внимание уделяется информационным, а не бизнес-системам. Эта модель описывает хранилища данных (data stores), где информация находится постоянно (это может быть база данных в компьютере или просто кабинет с папками), процессоры, манипулирующие с этими данными, и потоки данных, передающие данные от одного процессора другому. Она активно использует статическую информационную модель: последняя описывает, что означают такие концепции, как путешественник или отель, но ничего не сообщает о том, где содержится информация. Напротив, из модели потоков данных становится ясно, что информация о туристической поездке находится в базе данных покупок до завершения поездки и оплаты всех счетов, после чего резюме этой информации передается в маркетинговую информационную систему, а все остальное — в архив.

Объектные модели

Объектные модели содержат как динамический, так и статический компоненты. Динамическая или поведенческая часть определения объекта сосредоточена на том, что может делать или делал каждый объект, представляя для этого набор операций или методов, описывающих его действия.

Жизненные циклы объекта

Жизненные циклы объекта (на языке UML это называется линиями жизни объекта) также заостряют внимание на индивидуальных объектах, но придерживаются более целостного подхода. Они описывают, что происходит с объектом на протяжении его жизни: как он создается, какие события с ним происходят, как он реагирует на эти события и какие условия приводят в конце к его разрушению.

Жизненные циклы объекта очень полезны для тестирования завершенности модели. Часто наблюдается тенденция к акцентированию внимания только на некоторых событиях за счет остальных. Пока мы не определим, каким образом каждый объект попадает в систему и как он удаляется из нее, полного понимания не будет.

Варианты использования

Варианты использования (use cases) анализируют выполнение специфических задач пользователя (например, человек, купивший туристическую путёвку, отменяет свой заказ). Вариант использования напоминает модель процесса, но в общем случае заостряет внимание на деятельности одного конкретного пользователя.

Варианты использования могут быть полезны как на этапе моделирования деловой активности, так и при описании внутреннего поведения информационных систем. Одна из опасностей заключается в смешении двух уровней. Лучше этого не делать, поскольку они представляют интерес для различных аудиторий.

Представленный в виде варианта использования диалог пользовательского интерфейса описывает, какой информацией пользователь обмениваются с системой, а не то, как она представлена на экране. Это естественным образом приводит к реализации XML, в которой информационное содержание отделено от особенностей представления.

Диаграммы взаимодействия объектов

Диаграммы взаимодействия объектов позволяют проанализировать обмен сообщениями между объектами на более тонком уровне детализации, чем модель потока данных.

Диаграммы взаимодействия объектов неоценимы, если требуется описать взаимодействие между различными системами. Они позволяют определить, какая информация содержится в каком сообщении. Поскольку эти сообщение написаны на языке XML, диаграммы взаимодействия

объектов дают нам контекст, требуемый для начала проектирования структуры XML каждого индивидуального сообщения.

Пример статической информационной модели

Информационную модель данных, которая затем будет трансформирована в XML-документ, можно представить в виде *древовидной* структуры, которая начинается с *корня* и заканчивается *листьями*.

XML-документ должен содержать *корневой элемент*, который является *родительским* для всех остальных элементов. Любой элемент (кроме находящихся на самом нижнем уровне дерева) может иметь вложенные элементы (*дочерние элементы*).

С помощью терминов *родитель*, *дочерний* и *потомок* описываются отношения между элементами в дереве XML документа. *Родители* содержат *дочерние элементы*, а дочерние элементы одного уровня называются *потомками* (*братьями* или *сестрами*).

На рисунке представлен пример статической информационной модели, описывающей отношения между элементами в предметной области «Книжный магазин».

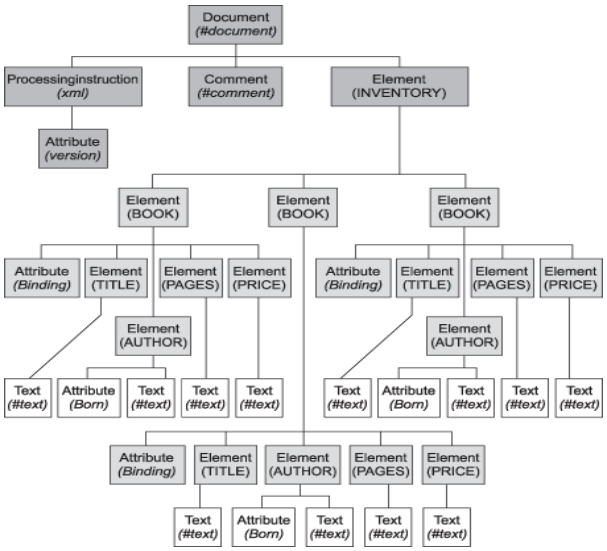

РИСУНОК 1

```
<?xml version=”1.0” encoding=”windows-1251” ?>

<!—Имя файла: Inventory Dom.xml →
<INVENTORY>
  <BOOK Binding=”mass market paperback”>
    <TITLE>The Adventures of Huckleberry Finn</TITLE>
    <AUTHOR Born=”1835”>Mark Twain</AUTHOR>
    <PAGES>298</PAGES>
    <PRICE>$5.49</PRICE>
  </BOOK>
  <BOOK Binding=”trade paperback”>
    <TITLE>The Marble Faun</TITLE>
```

```
    <AUTHOR Born=”1804”>Nathaniel Hawthorne</AUTHOR>
    <PAGES>473</PAGES>
    <PRICE>$10.95</PRICE>
  </BOOK>
  <BOOK Binding=”hardcover”>
    <TITLE>Moby-Dick</TITLE>
    <AUTHOR Born=”1819”>Herman Melville</AUTHOR>
    <PAGES>724</PAGES>
    <PRICE>$9.95</PRICE>
  </BOOK>
</INVENTORY>
```

Пример листинга программы.

Методические указания ко ***второй*** части лабораторной работы

Синтаксис XML

Для ограничения тегов в разметке XML, так же как и в HTML, используются угловые скобки: тег начинается со знака "меньше" (<) и завершается знаком "больше" (>). Но необходимо помнить, что в отличие от HTML вся разметка XML чувствительна к регистру символов, это касается как имен тегов, так и значений атрибутов.

Имена

В языке XML все имена должны начинаться с буквы, символа нижнего подчеркивания (_) или двоеточия (:) и продолжаться только допустимыми для имен символами, а именно: они могут содержать только буквы, входящие в секцию букв кодировки Unicode, арабские цифры, дефисы, знаки подчеркивания, точки и двоеточия. Однако имена не могут начинаться со строки *xml* в любом регистре. Имена, начинающиеся с этих символов, зарезервированы для использования консорциумом W3C.

Структура XML- документа

Любой XML-документ состоит из следующий частей:

- Необязательный пролог
- Тело документа
- Необязательный эпилог, следующий за деревом элементов

Рассмотрим каждую из частей более подробно.

Пролог

Пролог состоит из нескольких частей:

Необязательное *объявление XML* (XML Declaration).

Объявление заключено между символами <?...?> и содержит:

пометку xml и номер версии (*version*) спецификации XML;

указание на кодировку символов (*encoding*), в которой написан документ (по умолчанию encoding="UTF-8");

параметр standalone, который может принимать значения "yes" или "no" (по умолчанию standalone="yes"). Значение "yes" показывает, что в документе содержатся все требуемые декларации элементов, а "no" - что нужны внешние определения DTD.

Все это вместе может выглядеть следующим образом:

<?xml version ="1.0" encoding-"windows-1251" standalone="yes"?>

Важно отметить, что в объявлении XML только атрибут *version* является обязательным, все остальные атрибуты могут быть опущены и, следовательно, принимать значения по умолчанию. Также нужно помнить, что все эти атрибуты следует указывать только в приведенном выше порядке.

Комментарии

Инструкции по обработке.

Назначение инструкций по обработке — сообщить информацию, передаваемую XML-процессором приложению. Инструкция по обработке имеет следующую общую форму записи:

<? Кому инструкция ?>

Здесь *Кому* есть имя приложения, которому адресована инструкция. Допускается любое имя при соблюдении следующих правил:

имя должно начинаться с буквы или символа подчеркивания (_), после чего могут следовать или не следовать другие буквы, цифры, точки (.), тире (-) или символы подчеркивания (_);

имя «xml», в любом сочетании строчных или прописных букв, зарезервировано («xml» строчными буквами используется в объявлении XML-документа, которое представляет собой разновидность инструкции по обработке).

Инструкция есть информация, передаваемая приложению. Она может состоять из любой последовательности символов, за исключением пары ?>, зарезервированной для обозначения окончания инструкции по обработке.

Например, следующая инструкция по обработке предписывает браузеру использовать CSS-таблицу из файла styles.css:

<?xml-stylesheet type="text/css" href=" styles.css"?>

Символы пустых пространств.

Необязательное *объявление типа документа*, `DTD` (Document Type Declaration), которое заключено между символами `<!DOCTYPE...>` и может занимать несколько строк. В этой части объявляются теги, использованные в документе, или приводится ссылка на файл, в котором записаны такие объявления.

После объявления типа документа также могут следовать комментарии, команды обработки и символы пустых пространств.

Поскольку все эти части необязательны, пролог может быть опущен.

Тело документа

Тело документа состоит из одного или больше *элементов*. В правильно оформленном XML-документе элементы формируют простое

иерархическое дерево, в котором обязательно присутствует корневой элемент (root element), в который вложены все остальные элементы документа. Имена элементов должны быть уникальны в пределах документа. Имя корневого элемента считается именем всего документа и указывается во второй части пролога после слова Doctype.

Элемент начинается открывающим тегом, затем идет необязательное содержимое элемента, после чего записывается закрывающий тег (в отличие от HTML, наличие закрывающего тега обязательно, исключением являются элементы без содержания, так называемые *пустые элементы*, которые могут быть записаны в сокращенной форме: <имя_элемента/>). В качестве содержимого элемента могут выступать:

Другие элементы

Символьные данные

Ссылки на символы

Для того чтобы вставить в текст документа некоторый символ, который, например не присутствует в раскладке клавиатуры либо может быть неправильно истолкован анализатором, используют ссылки на символы. Ссылка на символ обязательно начинается со знака "&" (амперсанта) и заканчивается точкой с запятой. Ссылки на символы записываются в следующем виде:

&# код_символа_в_Unicode;

Код символа можно записать и в шестнадцатеричном виде. В этом случае перед ним ставится символ "x":

&#xШестнадцатеричный_код_символа;

Ссылки на сущности

Ссылки на сущности позволяют включать любые строковые константы в содержание элементов или значение атрибутов. Ссылки на сущности, как и ссылки на символы, начинающиеся с амперсанта, после которого идет имя сущности и заканчивающиеся точкой с запятой:

&имя_сущности;

Ссылки на сущности указывают программе-анализатору подставить вместо них строку символов, заранее заданную в определении типа документа (DTD).

Комментарии

Если надо вставить в текст документа комментарий либо сделать какой-то фрагмент "невидимым" для программы-анализатора, то его оформляют следующим образом:

<!--…текст комментария…-->

Разделы **CDATA**

Секция CDATA используется, для того чтобы задать область документа, которую при разборе анализатор будет рассматривать как простой текст, игнорируя любые инструкции и специальные символы.

Программа-анализатор не разбивает секцию CDATA на элементы, а считает ее просто набором символов. В отличие от комментариев, содержание данной секции не игнорируется, а передается без изменений на выход программы анализатора, благодаря чему его можно использовать в приложении.

Секция CDATA начинается со строки <![CDATA[, после которой записывается содержимое секции. Завершается секция двумя закрывающими квадратными скобками и знаком "меньше":

<![CDATA[содержание секции]]>

Инструкции по обработке

Инструкции по обработке содержат указания программе-анализатору документа XML. Инструкции по обработке заключаются между символами <? и ?>. Сразу за начальным вопросительным знаком записывается имя программного модуля, которому предназначена инструкция. Затем, через пробел, идет сама инструкция, передаваемая программному модулю. Сама инструкция это обычная строка, которая не должна содержать набор символов "?>", означающий конец инструкции. Примером инструкции по обработке может служить строка объявления XML:

<?xml version="1.0" encoding="windows-1251"?>

Эта инструкция предназначена программе, обрабатывающей документ XML. Инструкция передает ей номер версии и кодировку, в которой записан документ.

Атрибуты

Открывающие теги либо теги пустых элементов в XML могут содержать атрибуты, представляющие собой пару *имя=значение*. В одном открывающем теге разрешается использовать только один экземпляр имени атрибута. Атрибуты могут содержать ссылки на объекты, ссылки на символы, текстовые символы. В отличие от языка HTML, в XML значения атрибутов обязательно надо заключать в апострофы ('), либо в кавычки ("). Таким образом, атрибут может быть записан в одном из двух форматов:

имя_атрибута="значение_атрибута"

имя_атрибута='значение_атрибута'

Атрибуты используются для того, чтобы связать некоторую информацию с элементом, а не просто включить ее в содержание последнего. Однозначного ответа на вопрос «Что лучше выбрать – элемент или атрибут?» не существует. Каждый выбирает то, что ему больше нравится. Атрибуты удобно использовать для описания простых значений или для указания типа элемента. Например, мы можем ввести в открывающий тег <city> атрибут type (который может принимать одно из значений: город, поселок, деревня). Тогда данный тег может выглядеть следующим образом:

<city type="город"> Новосибирск </city>

Эпилог

В эпилог XML могут входить комментарии, инструкции по обработке и/или пустое пространство.

В языке XML не определен индикатор конца документа, большинство приложений для этой цели используют завершающий тег корневого элемента документа. Таким образом, встретив завершающий тег корневого элемента, скорее всего, приложение закончит обработку документа, и эпилог обработан не будет.

Правильно оформленные и валидные документы

В общем случае XML- документы должны удовлетворять следующим требованиям:

Любой XML-документ должен всегда начинаться с инструкции <?xml?>, внутри которой также можно задавать номер версии языка, номер кодовой страницы и другие параметры, необходимые программе-анализатору в процессе разбора документа.

Теги (метаданные) в документе выделяются символами <> (угловые скобки). Название тега должно начинаться сразу после угловой скобки (пробелы между открывающей угловой скобкой и названием тега недопустимы). Каждый открывающий тэг, определяющий некоторую область данных в документе, обязательно должен иметь своего закрывающего «напарника», нельзя опускать закрывающие тэги.

В XML учитывается регистр символов.

Все значения атрибутов, используемых в определении тэгов, должны быть заключены в кавычки.

Вложенность тэгов в XML строго контролируется, поэтому необходимо следить за порядком следования открывающих и закрывающих тэгов.

Вся информация, располагающаяся между начальным и конечными тэгами, рассматривается в XML как данные, и поэтому учитываются все символы форматирования (т.е. пробелы, переводы строк, табуляции не игнорируются, как в HTML).

Если XML-документ не нарушает приведенные правила, то он называется *формально-правильным* и все анализаторы, предназначенные для разбора XML- документов, смогут работать с ним корректно.

Пример XML-документа

Ниже представлена структура XML-документа по приведенной в первой части данной лабораторной работы статической информационной модели предметной области «Расписание занятий».

```
<?xml version="1.0"?>
<timetable>
    <day dayOfWeek="Monday">
```

```
        <lesson type="practical">
                <timeFrom>08.00</timeFrom>
                <timeTo>09.30</timeTo>
                <subject>Deutsch</subject>
                <teacher>Borisova</teacher>
                <room>216</room>
        </lesson>
        <lesson type="lecture">
                <timeFrom>09.40</timeFrom>
                <timeTo>11.10</timeTo>
                <subject>SAP Administration</subject>
                <teacher>Egorov</teacher>
                <room>384</room>
        </lesson>
        <lesson  type="practical">
                <timeFrom>11.20</timeFrom>
                <timeTo>12.50</timeTo>
                <subject>SAP Administration</subject>
                <teacher>Petrov</teacher>
                <room>384</room>
        </lesson>
</day>
</timetable>
```

Прежде чем Internet Explorer отобразит XML-документ, его встроенный синтаксический XML-анализатор (parser) просмотрит содержимое документа. Если он обнаружит ошибку, Internet Explorer отобразит страницу с сообщением об ошибке, не предпринимая попытки отобразить документ.

Internet Explorer использует для отображения документа имеющуюся по умолчанию таблицу стилей. Именно поэтому в описании ошибки упоминается использование таблицы стилей XSL.