JAVA-Орієнтовані технології

Лекція 9.2

доц. кафедри Інформатики
Сінельнікова Т.Ф.

Зміст

- Система Swing
  - Клас JApplet
  - Мітки Swing
  - Текстові поля
  - Кнопки
  - Клас JButton
  - Прапорці
  - Перемикачі
  - Поля зі списком
  - Панелі з кладками
  - Панелі прокрутки
  - Дерева
  - Таблиці
  - Інші можливості і майбутнє Swing-технології

Зміст

- Компоненти Java Beans
  - Переваги технології Java Beans
  - Інструментарій побудови додатків
  - Комплект розробника Bean-Компонентів
  - JAR-Файли
  - Інтроспекція
  - Проектні шаблони для властивостей
  - Проектні шаблони для подій
  - Створення нового Bean-компонента
  - Використання інтрефейсу BeanInfo
  - Обмежені властивості
  - Збереженість
  - Конфігуратор
  - Java Beans API
  - Майбутнє Bean-технології

Зміст

- SOAP-Технології
  - Що таке SOAP
  - Сутність SOAP
  - Що таке веб-сервіси
  - Механізм взаємодії клаєнта і сервера
  - WSDL файл
  - SOAP Envelope
  - Об'єктна модель SOAP Toolkit

Система Swing

- Swing API - Це набір класів, який забезпечує більш потужні та гнучкі компоненти, ніж AWT.

Система Swing

- На додаток до знайомих компонентів типу кнопок, прапорців і міток Swing постачає кілька цікавих додатків, включаючи панелі з вкладками, панелі з прокруткою, дерева і таблиці.
- Навіть знайомі компоненти, такі як кнопки, мають у Swing більше можливостей.
- Наприклад, з кнопкою можна пов'язати як зображення, так і текстову рядок.

Клас *JApplet*

Фундаментальним для Swing є клас JApplet, який розширює клас Applet. Аплети, які використовують Swing-Компоненти, повинні бути підкласами JApplet.

JApplet багатий функціональними можливостями, яких немає в Applet. Наприклад, JApplet підтримує різні "панелі", такі як панель змісту (Content pane), прозора ("скляний") панель (Glass pane) і коренева панель (Root pane).

При додаванні компонента до примірника JApplet не викликайте метод add о для *аплету.* Замість цього, викличте add о для *панелі змісту* JApplet-Об'єкта. Панель змісту може бути отримана за допомогою наступного методу:

- **Container getContentPane()**

У Swing значки інкапсульовані класом Imageicon, який малює значок з зображення. Нижче показано два його конструктора:

- **Imageicon (String filename)**
- **Imageicon (UKL *url)***

Перша форма використовує зображення у файлі з ім'ям filename, а друга форма - в ресурсі, розташованому по URL-Адресою url.

Мітки Swing

Мітки Swing - Екземпляри класу jLabel, який розширює JComponent. Він може відображати тексти та / або значки. Ось деякі з його конструкторів:

- **JLabel (Icon i)**
- **Label (String s)**
- **JLabel (String *s,* Icon i, int align)**

Тут sii- Текст і значок, що використовується для позначки. Параметр *align* визначає вирівнювання і має значення left, right або center.

Ці константи визначені в інтерфейсі SwingConstants, поряд з кількома іншими, використовуваними Swing-класами.

Мітки Swing

Значок і текст, пов'язаний з міткою, можна зчитувати і записувати наступними методами:

- **Icon getIcon ()**
- **String getText ()**
- **void setIcon (Icon i)**
- **void setText (String s)**

Тут ***i*** і ***s*** - Значок і текст, соответственно.

Текстові поля

Поле тексту Swing інкапсульовані класом JTextcomponent, який розширює JComponent. Він забезпечує функціональні можливості, які є загальними для текстових Swing-Компонентів.

Один з його підкласів - JTextFieid, дозволяє редагувати один рядок тексту. Ось деякі з його конструкторів

- **JTextFieid ()**
- **JTextFieid (int *cols)***
- **JTextFieid (String s, int *cols)***
- **JTextFieid (String s)**

Тут s - Рядок, який буде представлена; *cols* - число позицій в текстовому полі.

Кнопки

Кнопки Swing мають властивості, яких не можна знайти в класі Button, визначеному в AWT.

Кнопки Swing - Це підкласи класу AbstractButton, який розширює JComponent.

AbstractButton містить багато методів, які дозволяють управляти поведінкою кнопок, прапорців і перемикачів.

Наприклад, можна визначати різні піктограми для відображення компонента, коли він віджатий (Disabled), натиснутий (Pressed), або обраний (Selected).

Деяку піктограму можна використовувати як значок *"Наїзду"* (Rollover), який відображається, коли курсор миші встановлений поверх цього компонента ("наїхав" на нього).

- **void setDisabledIcon (Icon *dx)***
- **void setPressedIзon (Icon *pi)***
- **void setSelectedIcon (Icon *si)***
- **void setRolloverIcon (Icon *rj)***

Тут ***di***, ***pi***, ***si*** і ***ri*** - Піктограми, які потрібно використовувати для цих різних станів.

Кнопки

Текст, пов'язаний з кнопкою, можна читати і записувати за допомогою таких методів:

- **String getText ()**
- **void setText (String s)**

Тут s - Текст, який потрібно зв'язати з кнопкою.

При натисканні кнопки конкретні підкласи AbstractButton генерують action-Події.

Блоки прослуховування реєструють і скасовують реєстрацію для цих подій за допомогою таких методів:

- **void addActionListener (ActionListener *al*)**
- **void removeActionListener (ActionListener *al)***

Тут ***al*** - Блок прослуховування подій дії.

Клас *JButton*

Клас JButton забезпечує функціональні можливості кнопки. JButton дозволяє пов'язати з кнопкою зображення, рядок або і те й інше. Деякі з його конструкторів:

- **JButton (Icon i)**
- **JButton (String s)**
- **JButton (String s, Icon i)**

Тут ***s*** і ***i*** - Рядок і зображення, які використовуються для кнопки.

Прапорці

Клас JCheckBox, який забезпечує функціональні можливості прапорця, є конкретною реалізацією класу AbstractButton.

Деякі з його конструкторів:

- **JCheckBox (Icon i*)***
- **JCheckBox(Icon i, boolean *state)***
- **JCheckBox (String s)**
- **JCheckBox (String s, boolean *state)***
- **JCheckBox (String s, Icon i)**
- **JCheckBox (String *s,* Icon *z,* boolean state)**

Тут використовуються наступні параметри:

- ***i*** - Зображення для кнопки,
- ***s*** - Текст.

Якщо **state** - True, прапорець спочатку вибраний. В іншому випадку - ні.

Перемикачі

Перемикачі підтримуються класом JRadioButton, який є конкретною реалізацією класу AbstractButton. Деякі з його конструкторів:

- **JRadioButton (Icon i)**
- **JRadioButton (Icon z, boolean *state)***
- **JRadioButton (String s)**
- **JRadioButton (String s, boolean state)**
- **JRadioButton (String s, Icon i.)**
- **JRadioButton (String s, Icon f, boolean state)**

де використовуються параметри:

- ***i*** - Зображення для кнопки;
- ***s*** - Текст.

Якщо **state** -true, кнопка спочатку вибрана. Інакше - ні.

Поля зі списком

Swing забезпечує *комбіноване поле* (Combo box) - Комбінацію текстового поля та списку, через клас JComboBox, який розширює JComponent.

Комбіноване поле зазвичай відображає один вхід (елемент) списку. Однак воно може також відображати і розкривається список, який дає можливість користувачу вибирати різні входи. Ви також можете ввести (з клавіатури) своє значення елементу списку в текстове поле. Нижче показано два конструктора JComboBox:

- **JComboBox ()**
- **JComboBox (Vector v)**

Тут **v** - Вектор, який ініціалізує комбіноване поле.

Елементи додаються до списку виборів за допомогою методу addItem (), чия сигнатура має вигляд:

- **void addItem (Object *obj)***

Тут ***obj*** **-** об'єкт, який буде додано до комбінованого полю.

Панелі з вкладками

Панель з вкладками (Tabbed pane) - Компонент, який з'являється як група папок в CAB-файлі (File cabinet). Кожна папка має заголовок. Коли користувач вибирає папку, її вміст стає видимим. Тільки одна з папок може бути обрана одночасно.

Панель з вкладками інкапсульовані класом JTabbedPane, який розширює IComponent. Ми будемо використовувати його замовчуваний конструктор. Вкладки визначаються за допомогою наступного методу:

- **void addTab (String *str,* Component *camp)***

Тут ***str*** - Заголовок вкладки; *camp* - Компонент, який повинен бути доданий у вкладку. Як правило, додаються об'єкти класу jpanei або його підкласів.

Загальна процедура використання в аплеті панелі з вкладками:

- Створити об'єкт JTabbedPane.
- Викликати addTab () для додавання вкладки в панель. (Аргументи цього методу визначають заголовок вкладки і компонента, котрий вона містить)
- Повторити крок 2 для кожної вкладки.
- Додати панель з вкладками в панель змісту аплету.

Панелі прокрутки

Панель прокрутки (Scroll pane) - Компонент, який представляє прямокутну область, в якій компонент може бути переглянутий. У разі необхідності в панель можна додати горизонтальну і/або вертикальну смуги прокрутки.

Панелі прокрутки реалізовані в Swing класом JScroiiPane, який розширює JComponent. Ось деякі з його конструкторів:

- **JScrollPane (Component comp)**
- **JScroiiPane (int *vsb,* int *hsb)***
- **JScrollPane (Component *camp,* int *vsb,* int *hsb)***

Тут comp - компонент, який буде додано в панель прокрутки; ***vsb*** і ***hsb*** — int-Константи, які визначаються; коли потрібно показувати вертикальні і горизонтальні смуги прокрутки в панелі прокрутки. Ці константи визначені інтерфейсом scrollPaneConstants.

Дерева

Дерево (Tree) - Компонент, який представляє *ієрархічний* вид даних. Користувач має можливість розгорнути або згорнути індивідуальні піддерева в цьому показі. Дерева реалізовані в Swing класом JTree, який розширює jcomponent. Ось деякі з його конструкторів:

- **JTree (Hashtable At)**
- **JTree (Object *obj)***
- **JTree (TreeNode tn)**
- **JTree (Vector v)**

Перша форма створює дерево, в якому дочірньої вершиною є кожен елемент хеш-таблиці ***ht.*** У другій формі дочірньої вершиною є кожен елемент масиву ***obj****.* У третій формі параметр ***tn*** вказує кореневий вузол дерева. Нарешті, остання форма використовує елементи векторного параметра **v** як дочірні вершини.

Дерева

Коли вузол розгортається або згортається, об'єкт JTree генерує події. Методи AddTreeExpansionListener() І removeTreeExpansionListener() Дозволяють блокам прослуховування реєструвати або скасовувати реєстрацію для цих повідомлень. Сигнатури цих методів:

- **void addTreeExpansionListener (TreeExpansionListener *tel)***
- **void removeTreeExpansionListener (TreeExpansionListener *tel)***

Тут **tel -** Об'єкт блоку прослуховування.

Дерева

Щоб транслювати клацання миші на певній точці дерева в гілку (шлях) дерева використовується метод GetPathForLocation (). Його сигнатура:

- **TreePath getPathForLocation (int *x,* int y)**

Тут ***x*** і ***y*** — координати покажчика миші, де виконано клацання. Значення, що повертається - об'єкт TreePath, який інкапсулює інформацію щодо вузла дерева, вибраного користувачем.
Клас TreePath інкапсулює інформацію про шлях до специфічного вузлу дерева. Він забезпечує кілька конструкторів і методів.
Інтерфейс TreeNode оголошує методи, які отримують інформацію щодо вузла дерева. Наприклад, можливо отримати посилання до батьківського вузла або перерахуванню дочірніх вузлів.
Інтерфейс MutablereeNode розширює TreeNode. Він оголошує методи, які можуть вставляти і видаляти дочірні вузли або змінювати батьківський вузол.
Клас DefaultMutableTreeNode реалізує інтерфейс MutabieTreeNode. Він представляє вузол вдереві. Нижче показаний один з його конструкторів:

- **DefaultMutableTreeNode (Object** *obj)*

Тут ***obj*** — Об'єкт, який буде включений в цей вузол дерева. Новий вузол дерева не має батьківського або дочірнього вузла.

Дерева

Щоб створювати ієрархію вузлів дерева, можна використовувати метод add () класу DefaultMutableTreeNode. Його сигнатура:

- **void add (MutabieTreeNode child)**

Тут ***child*** — Змінний вузол дерева, який повинен бути доданий як дочірній до поточного вузла.

Події розширення дерева (Tree expansion events) описані класом TreeExpansionEvent У пакеті javax.swing.event. Метод getPath() цього класу повертає об'єкт TreePath, який описує шлях до зміненого вузлу. Його сигнатура:

- **TreePath getPath()**

Інтерфейс TreeExpansionListener забезпечує наступні два методи:

- **void treeCollapsed {TreeExpansionEvent tee)**
- **void (TreeExpansionEvent tee)**

Тут ***tee*** — Подія розширення дерева. Перший метод викликається, коли піддерево згортається, а другий - коли піддерево стає видимим (розгортається).

Дерева

Кроки алгоритму створення аплету з деревом такі:

- Створити Об'єкт JTree.
- Створити об'єкт JScrollPans. (Аргументи конструктора визначають дерево і установку вертикальних і горизонтальних смуг прокрутки.)
- Додати дерево до панелі прокрутки.
- Додати панель прокрутки до панелі змісту аплету.

Таблиці

Таблиця (Table) - Компонент, який відображає рядки і стовпці даних. Для зміни розмірів стовпців можна переміщати курсором їх межі. Можна також перетягувати стовпці в нову позицію.

Таблиці реалізовані класом JTable, який розширює JComponent. Ось один з його конструкторів:

- **JTable (Object *data*[ ] [], Object *colHeads*[ ])**

Тут **data** - Двомірний масив інформації, яка буде представлена у формі таблиці; ***colHeads*** - Одномірний масив з заголовками стовпців.

Кроки алгоритму для створення таблиці в аплеті такі:

- Створити об'єкт JTable.
- Створити об'єкт JScrollPane. (Аргументи конструктора визначають таблицю і установку для вертикальних і горизонтальних смуг прокрутки.)
- Додати таблицю в панель прокрутки.
- Додати панель прокрутки в панель змісту аплету.

Інші можливості і майбутнє Swing-Технології

Як говорилося раніше, Swing - Це велика система. Вона має ще дуже багато властивостей.

Наприклад, Swing забезпечує інструментальні панелі (Toolbars), підказки кнопкових команд (Tooltips), і прогрес-смужки (Progress bars).

Компоненти Swing можуть також мати специфічним pluf1 -Властивістю, яке "Pluggable look-and-feel" означає, що до Swing-Компоненту можна підключити інший вигляд і поведінку. Причому це може бути зроблено динамічно. Ви можете навіть проектувати ваш власний вигляд і поведінку.

Таким чином, Swing-Підхід до GUI-Компонентам міг би колись у майбутньому замінити AWT-Класи.
Swing - Тільки одна частина бібліотеки класів Java Foundation Classes **(JFC).** Можна використовувати і інші частини JFC.
Java 2D API (Система поліпшеного управління графікою) забезпечує розширені можливості роботи з формами, текстом та зображеннями. Drag-and-Drop API (Пакет java.awt.and) підтримує обмін інформацією між Java- і не Java-Програмами.

Компоненти Java Beans

Java Bean — Компонент є програмним компонентом, який розроблений так, щоб багаторазово використовуватися в різних середовищах.

Переваги технології Java Beans

Архітектура програмних компонентів забезпечує стандартні механізми для роботи з програмними будівельними блоками.

Нижче перераховані деякі з специфічних вигод, які забезпечує технологія Java для розробника компонентів.

- Bean-Компонент отримує всі вигоди від Java-Парадигми "писати - одного разу, виконувати - де завгодно".
- Властивості, події та методи Bean-Компонента, які пред'являються
- інструменту побудови додатків, можуть бути керованими.
- Bean-Компонент може бути спроектований для правильної роботи в
- різних мовних регіонах, що робить його корисним для глобальних
- ринків.
- Для допомоги в конфігуруванні Bean-Компонента можливе використання допоміжного програмного забезпечення. Це програмне забезпечення необхідно тільки тоді, коли для того компонента встановлюються параметри часу розробки. Його не потрібно включати в середу часу виконання.
- Параметри настройки конфігурації Bean-Компонента можуть бути збережені в постійній пам'яті і відновлені у більш пізній час.
- Bean-Компонент можна реєструвати, щоб приймати події від
- інших об'єктів, і він може генерувати події, які посилаються
- до інших об'єктів.

Інструментарій побудови додатків

При роботі з компонентами Java Beans більшість розробників використовує *інструментарій побудови додатків.* Це утиліта, яка дає можливість конфігурувати набір Bean-Компонентів, з'єднувати їх разом і створювати працююче додаток. Її головні можливості:

- Забезпечення палітри, яка перераховує всі доступні Beап-компоненти. Коли розроблений або куплений додатковий Bean-Компонент, його можна додати до палітри.
- Відображення робочого листа, який дозволяє проектувальникові розташовувати Bean-Компонент в графічному інтерфейсі користувача. Проектувальник може перетягнути Bean-Компонент з палітри в цей робочий лист.
- Використання спеціальних редакторів і настроювачів, що дозволяють конфігурувати Bean-Компонент. Це механізм, за допомогою якого поведінка Bean-Компонента може бути адаптовано до специфічної середовищі.
- Застосування спеціальних команд, які дозволяють проектувальнику за
- прашівать стан і поведінку Bean-Компонента. Ця інформація автоматично стає доступною, коли Bean-Компонент додається до палітри.
- Зв'язування Bean-Компонентів. Це означає, що події, згенеровані одним компонентом відображаються у виклики методів інших компонентів.
- Збереження Bean-Компонентів у постійній області пам'яті, коли їх
- колекція налаштована і пов'язана. Ця інформація може використовуватися в більш пізній час для відновлення стану програми.

Комплект розробника Bean-Компонентів

Комплект розробника Bean-Компонентів (BDK,Bean Developer Kit), Доступний на сайті JavaSoft, Є простим прикладом інструментарію, який дає можливість створювати, конфігурувати і підключати набір Bean-Компонентів.

Існує також набір зразків Bean-Компонентів з їх вихідним кодом.

JAR-Файли

Файл JAR дозволяє ефективно розгортати набір класів і пов'язаних з ними ресурсів.

Наприклад, розробник може побудувати мультимедійне додаток, що використовує різні звукові та графічні файли.

Набір Bean-Компонентів може контролювати, як і коли представляти цю інформацію. Всі ці частини можуть бути поміщені в один JAR-Файл.

JAR-Технологія набагато спрощує процес постачання і встановлення програмного забезпечення.

Крім того, елементи в JAR-Файлі стиснуті, що робить завантаження файлу подібного роду набагато швидше, ніж роздільну завантаження декількох нестиснутих файлів.

З індивідуальними елементами в JAR-Файлі можна також зв'язати *цифрові підписи* (Сигнатури). Вони дозволяють споживачу переконатися, що дані елементи були створені певною організацією або індивідуумом.

Інтроспекція

Інтроспекція - це процес аналізу Bean-Компонента для визначення його можливостей.

Це суттєва властивість Java Beans API, тому що воно дозволяє інструменту побудови додатків надати розробнику програм інформацію про компоненті.

Існують два способи, за допомогою яких розробник Bean-Компонента може вказувати, які з його властивостей, подій і методів повинні бути показані інструментом розробника додатків.

- У першому способі використовуються прості угоди про імена. Вони дозволяють механізмам інтроспекції виводити інформацію про Bean-Компоненті.
- У другому способі створюється додатковий клас, який явно постачає цю інформацію.

Проектні шаблони для властивостей

Властивість - це підмножина стану Bean-Компонента. Значення, призначені властивостями, визначають поведінку і зовнішній вигляд цього компонента.

Прості властивості
Просте властивість має одиночне значення. Воно може бути ідентифіковано наступними *проектними шаблонами* (Design patterns), де N- Ім'я властивості, а T - Його тип:

- **public T getN () ;**
- **public void setN (T *arg) ;***

Властивість *читання / запис* (Read / write) використовує обидва цих методи для доступу до своїх значень. Властивість *тільки для читання* (Read-only) використовує тільки метод get, а властивість *тільки для запису* (Write-only) - Тільки метод set.

Проектні шаблони для властивостей

Булеві властивості

Булево властивість має значення true або false. Воно може бути ідентифіковано наступними проектними шаблонами, де n- Ім'я властивості:

- **public boolean isN ();**
- **public boolean getN ();**
- **public void setN (boolean value)**;

Щоб витягти значення булевого властивості, можна використовувати перший або другий шаблон. Однак, якщо клас використовує обидва методи, застосовується перший шаблон.

Проектні шаблони для властивостей

Індексовані властивості

Індексовані властивість складається з множинних значень. Воно може бути ідентифіковано наступними проектними шаблонами, де N - Ім'я властивості, а T - Його тип:

- **public T getN (int *index);***
- **public void setN (int *index,* T value);**
- **public T [] getN ();**
- **public void setN (T values[]);**

Проектні шаблони для подій

Bean-Компоненти використовують модель делегування подій. Вони можуть генерувати події і посилати їх іншим об'єктам. Події можна ідентифікувати наступними проектними шаблонами, де T - Тип події:

- **public void addTListener (TListener *eventListener)*;**
- **public void addTListener (TListener *eventListener)***
  throws TooManyListeners;
- **public void removeTListener (TListener *eventListener);***

Перший шаблон вказує, що Bean-Компонент може поширювати подія багатьом слухачам.

Другий шаблон вказує, що Bean-Компонент може передавати подія тільки одному слухачеві.

Третій шаблон використовується блоком прослуховування, коли він більше не бажає приймати певний тип повідомлень про події від Bean-Компонента.

Створення нового Bean-Компонента

Алгоритм створення нового Bean-Компонента виглядає так:

- Створити каталог для нового Beап-компонента.
- Створити вихідний Java-Файл (або кілька файлів).
- Компілювати вихідні файли.
- Створити файл опису.
- Генерувати JAR-Файл.
- Запустити BDK.
- Протестувати програму.
- Наступні розділи обговорюють кожен з перелічених кроків цього алгоритму в деталях.

Використання інтерфейсу *BeanInfo*

Цей інтерфейс визначає кілька методів, включаючи такі:

- **PropertyDescriptor[] getPropertyDescriptors ()**
- **EventSetDescriptor[] getEventSetDescriptors ()**
- **MethodDescriptor [ ]getMethodDescriptors()**

Вони повертають масиви об'єктів, які забезпечують інформацію про властивості, події і методи Bean-Компонента.

SimpieBeaninfo - Це клас, який забезпечує задані за замовчуванням реалізації інтерфейсу Beaninfo, включаючи тільки що перераховані три методи. Ви можете розширити цей клас і перевизначити один з них (або декілька).

Обмежені властивості

Bean-Компонент, який має *обмежене* (Constrained) властивість, генерує подія, коли здійснюється спроба змінити значення цієї властивості.

Подія має тип PropertyChangeEvent. Воно посилається об'єктах, які попередньо зареєстрували інтерес в прийомі таких повідомлень.

Дані об'єкти мають здатність відхиляти запропоновану зміну.

Ця можливість дозволяє Bean-Компоненту працювати по-різному, узгоджуючи з середовищем часу виконання.

Збереженість

Збереженість (Persistence) - Це здатність зберігати Bean-Компонент в незалежній пам'яті і відновлювати його в більш пізній час.

Особливо важливо запам'ятовувати параметри конфігурації.

Конфігуратор

Розробник може змінювати властивості Bean-Компонента у вікні **Properties BDK.**

Однак, це не кращий інтерфейс користувача для комплексного компонента з великим числом взаємодіючих властивостей.

Тому Bean-Розробник може створювати спеціалізований *конфігуратор* (Customizer), який допомагає іншому розробнику конфігурувати дане програмне забезпечення.

Конфігуратор може забезпечити покрокове керівництво процесом, якому потрібно дотримуватися, щоб використовувати компонент в певному контексті.

Він може також забезпечити і оперативну (Online) документацію.

Розробник повинен забезпечити більшу гнучкість Bean-Компонентів, щоб розробити конфігуратор, який може диференціювати його продукт в ринковому просторі.

Java Beans API

Функціональні можливості Java Bean-Компонентів забезпечуються набором класів та інтерфейсів в пакеті java.beans.

Майбутнє Bean-Технології

Технологія Java Beans знаходиться на вістрі Java-Програмування, а створення компонентного програмного забезпечення буде важливою частиною більшості робіт Java-Програмістів в найближчому майбутньому.

Крім того, Bean-Компоненти формують додаток до ActiveX - Архітектурі програмних компонентів фірми Microsoft.

Програми типу Internet Explorer, Microsoft Office iVisual Basic можуть служити контейнерами для цих компонентів.

SOAP-Технології

SOAP покликані вирішити проблеми крос-платформного взаємодії додатків.

Що таке SOAP

В даний час використовуються технології віддаленого виклику методів (DCOM, CORBA / IIOP і RMI) досить складні в налаштуванні та організації взаємодії.

Це тягне за собою проблеми в експлуатації та функціонуванні розподілених систем (проблеми безпеки, транспорт через брандмауери тощо).

Існуючі проблеми успішно вирішені створенням SOAP (Simple Object Access Protocol), простого протоколу, заснованого на XML, для обміну повідомленнями в розподілених середовищах (WWW).

Він призначений для створення веб-сервісів і віддаленого виклику методів.

SOAP можна використовувати з різними транспортними протоколами, включаючи HTTP, SMTP і т.д.

Сутність SOAP

Фактично SOAP є новою іпостассю парадигми RPC (віддаленого виклику процедур), яка на відміну від своїх попередниць реалізована на основі загальноприйнятих стандартів HTTP і XML.

Протокол HTTP 1.1 цілком вписується в парадигму RPC. У ньому присутній природна модель запитів / відповідей.

Він дозволяє підтримувати активні сполуки, при цьому забезпечуючи можливість переносних незалежного подання об'єктних посилань.

Сутність SOAP

Другий елемент зв'язки - XML - виступає тут як універсального засобу представлення параметрів виклику RPC. На відміну від існуючих протоколів (CDR для CORBA або NDR для DCOM) він володіє цілим рядом переваг. По-перше, для нього вже є досить багато реалізацій системного ПО, зокрема модулів розбору XML-документів для всіх платформ і середовищ. По-друге, XML - текстовий мову, що значно спрощує розробку на його основі власних програм. Фактично з цією роботою може впоратися навіть не дуже кваліфікований програміст. По-третє, гнучкість і розширюваність XML роблять його привабливим при реалізації та супроводі довгострокових проектів.

Однак для SOAP важливий не XML сам по собі, а формат XML-повідомлень, що використовуються для передачі запитів і відповідей на них. Відповідно до специфікації SOAP кожне таке повідомлення складається з двох частин:

- конверта (envelop) - в ньому містяться заголовки запитів і відповідей, у яких зазначене супровідна інформація (наприклад, маршрут прямування, кінцевий і проміжний одержувачі тощо);
- тіла (body) - куди містяться атрибути виклику (запитувані методи, передані їм параметри і т. д.).

Що таке веб-сервіси

Веб-сервіси - це функціональність і дані, надані для використання зовнішніми додатками, які працюють з сервісами за допомогою стандартних протоколів і форматів даних.

Веб-сервіси повністю незалежні від мови і платформи реалізації.

Технологія веб-сервісів є наріжним каменем програмної моделі Microsoft. NET.

Механізм взаємодії клієнта і сервера

1. Клієнтський додаток створює екземпляр об'єкту SOAPClient
2. SOAPClient читає файли опису методів веб-сервісу (WSDL і Web Services Meta Language - WSML). Ці файли можуть зберігатися і на клієнті.
3. Клієнтський додаток, використовуючи можливості пізнього зв'язування методів об'єкта SOAPClient, викликає метод сервісу. SOAPClient формує пакет запиту (SOAP Envelope) і відправляє на сервер. Можливе використання будь-якого транспортного протоколу, але, як правило, використовується HTTP.
4. Пакет приймає серверний додаток Listener (може являти собою ISAPI додаток або ASP сторінку), створює об'єкт SOAPServer і передає йому пакет запиту
5. SOAPServer читає опис веб-сервісу, завантажує опис і пакет запиту в XML DOM дерева
6. SOAPServer викликає метод об'єкта / програми, що реалізовує сервіс
7. Результати виконання методу або опис помилки конвертуються об'єктом SOAPServer в пакет відповіді і відправляються клієнту
8. Об'єкт SOAPClient проводить розбір прийнятого пакета і повертає клієнтського додатку результати роботи сервісу або опис виниклої помилки.

WSDL файл

WSDL файл це документ у форматі XML, що описує методи, що надаються веб-сервісом.

Також параметри методів, їх типи, назви та місцезнаходження Listener сервісу.

SOAP Toolkit автоматично генерує цей документ.

SOAP Envelope

SOAP Envelope (Пакет) - XML документ, який містить в собі запит / відповідь на виконання методу.

Зручніше за все розглядати його як поштовий конверт, в який вкладена інформація.

Тег Envelope повинен бути кореневим елементом пакета.

Елемент Header не обов'язковий, а Body повинен бути присутнім і бути прямим нащадком елемента Envelope.

У випадку помилки виконання методу сервер формує пакет, який містить в тегу Body елемент Fault, який містить докладний опис помилки.

Об'єктна модель SOAP Toolkit

Об'єктна модель SOAP Toolkit дає можливість працювати з об'єктами низькорівневого API:

- SoapConnector - Забезпечує роботу з транспортним протоколом для обміну SOAP пакетами
- SoapConnectorFactory - Забезпечує метод створення коннектора для транспортного протоколу, зазначеного в WSDL файлі (тег)
- SoapReader - Читає SOAP повідомлення і будує XML DOM дерева
- SoapSerializer - Містить методи створення SOAP повідомлення
- IsoapTypeMapper, SoapTypeMapperFactory - Інтерфейси, дозволяють працювати зі складними типами даних

Дякую за увагу!