Приложение 8 IDE Eclipse

Установка и запуск

Eclipse — сообщество открытого кода или open source, чьи проекты сфокусированы на создании открытой платформы для разработки, развертывания, управления приложениями с использованием различных фреймворков, инструментов и сред исполнения. Загрузить продукты Eclipse можно на официальном сайте http://eclipse.org/

Eclipse Java IDE for Web Developers — среда для разработки веб-приложений на Java. До загрузки IDE можно выбрать сборку для конкретной платформы. Ниже будет рассмотрена IDE из сборки Juno для операционной системы Windows (32-bit).

Для установки Eclipse Java IDE необходимо распаковать загруженный архив в ту директорию, в которой будет храниться IDE. Распакованная IDE занимает менее 300 Mб. Среда Eclipse используется для разработки приложений, поэтому предварительно необходимо установить Java Development Kit (JDK). Оптимальную версию можно загрузить с сайта http://oracle.com/ и следовать мастеру установки.

Если JDK установлен, то для запуска Eclipse Java IDE следует воспользоваться файлом eclipse.exe. При запуске файла на экране появится окно (Рис. 1.), где необходимо выбрать workspace — каталог, в котором будут храниться

Workspace Launcher		
Select a workspace		
Eclipse stores your projects in a folder called a workspace. Choose a workspace folder to use for this session.		
Workspace: C:\workspace		Browse
Use this as the default and do not ask again		
	ОК	Cancel

Рис. 1. Выбор директории хранения проектов

проекты. В случае использования одного **workspace**, необходимо установить значение по умолчанию (отметить check box **I**).

Создание и запуск проекта

При первом запуске проекта на экране появится вкладка *Welcome page* (Рис. 2), содержащая несколько ссылок. При закрытии или сворачивании вкладки *Welcome page*, среда будет выглядеть как на рис. 3.

Создать новый проект в Eclipse IDE можно несколькими способами:

— нажать Alt+Shift+N;

— выбрать в меню File пункт New.

После совершения одного из двух перечисленных действий необходимо выбрать тип проекта. Далее будут подробно рассмотрены два типа проектов: консольный и веб-проект.



Рис. 2. Welcome page



Рис. 3. Eclipse IDE

Создание, запуск и отладка консольного проекта

Для создания консольного проекта необходимо выбрать меню File — New — Project. В результате выбора откроется мастер создания проекта (Рис. 4.).

В Мастере создания проекта необходимо выбрать Java Project и нажать кнопку Next. Далее вводится имя проекта, выбирается JRE (Рис. 5.). Можно задать некоторые дополнительные параметры, нажимая кнопку Next, можно оставить все остальное по умолчанию и нажать кнопку Finish для создания проекта. В *Package explorer* слева появится только что созданный проект (Рис. 6.).

Чтобы создать пакет в новом проекте, необходимо нажать правой кнопкой мыши на название проекта в *Package Explorer* или выбрать в главном меню **File** — **New** — **Package**. Затем ввести имя пакета и нажать кнопку **Finish**. После этого в папке **src** появится пиктограмма пустого пакета с именем. С помощью **File** — **New** можно создать класс (**Class**), интерфейс (**Interface**), перечисление (**Enum**) или другой элемент проекта (Рис. 7.).

New Project				_0×
Select a wizard Create a Java project				
Wizards:				
Image: Stars Project Image: Stars Project	ing Ant Buildfile work			
(?)	< Back	Next >	Finish	Cancel

Рис. 4. Мастер создания проекта

New Java Project		
Create a Java Project Create a Java project in the workspace	ce or in an external location.	
Project name: SampleProject		
V Use default location		
Location: C:\Users\admin\Deskto	p\java1\SampleProject	Browse
JRE		
Output Use an execution environment	t JRE: JavaSE-1.7	•
Use a project specific JRE:	jre7	
O Use default JRE (currently 'jre7	")	Configure JREs
Project layout	auror and share film	
 Ose project folder as root for so Create separate folders for source 	urces and class files	Configure default
Working sets		
Add project to working sets		
Working sets:		▼ Select
? Kack	Next > Fin	ish Cancel

Рис. 5. Создание Project



Рис. 6. Package Explorer

Java - Eclip File Edit Sour	ce Refactor Navigate Sea	arch Project Run Wi	ndow Help
Package E	▶ ☆ • • • • • • • • • • • • • • • • • •	`≷ ⊕ G • @] 7	<i>⊕ ₄</i> • ½ • ₩ • • •
@-so ⊡⊒\ JR	New Open in New Window		실 Java Project 다음 Project
	Open Type Hierarchy Show In	F4 Alt+Shift+W	Package
	Copy	Ctrl+C	C Interface C Enum
	Paste 🐹 Delete	Ctrl+V Delete	Annotation Source Folder
	Remove from Context Build Path	Ctrl+Alt+Shift+Down	K Java Working Set
	Source Refactor	Alt+Shift+S Alt+Shift+T	Hile UnitIed Text File
	≧ Import ⊿ Export		Task
5	Assign Working Sets	F5	Ctrl+N

Рис. 7. Создание нового элемента проекта

При создании класса необходимо как минимум ввести его имя. Так же в *Macmepe создания класса* можно добавить к классу интерфейсы, указать модификатор доступа, сообщить, что класс будет содержать метод **main()** и прочее (Рис. 8.).

Созданный класс будет выглядеть так (см. Рис. 9.).

Проект можно запустить нажатием сочетания клавиш Ctrl+F11 или кнопкой Run на панели инструментов (зеленый круг с белым треугольни-ком внутри).

IDE Eclipse

💭 New Java Class		<u>- 0 ×</u>
Java Class Create a new Java	class.	C
Source folder:	SampleProject/src	Browse
Package:	com.epam.sample.main	Browse
Enclosing type:		Browse
Name:	HelloWorld	
Modifiers:	⊙ public O default O private O protected □ abstract □ final □ static	
Superclass:	java.lang.Object	Browse
Interfaces:	\rm java.io.Serializable	Add
		Remove
Which method stubs	s would you like to create?	
	I public static void main(String[] args)	
	Constructors from superclass	
Do you want to add	comments? (Configure templates and default value <u>here</u>) Generate comments	
?	Finish	Cancel

Рис. 8. Создание нового класса

Java - Sample Project/src/com/epam/sam	pel/main/HelloWorld.java - Eclipse Project Run Window Helo					
3•88≥\$+0•0-4-×	₩ G•∅ ⊖∧*•₽⁄;	0 ¶ \$ •	49 • 65		955	🗌 🔡 🖓 Java EE 🕼 Java
Package Explorer 23 Package Explorer 23 Package Explorer 23 Package Explorer 24 Package Explore	Helloword java El package com.epam.sampel import java.io.Serieliz public class Helloworld public class Helloworld //- private static fina //- public static void // TODO Auto-ge } Problems El @ Javadoc @ Dec	.main; able; implements i l long seria; main(String[] merated metho Marabon	erializabl VersionUII args) { d stub	e { 2 = iL;		Task List 32 Task List 32 Trid Connect My/My 13 Connect to your task and ALM tools or greate a local task. Contine 32 Connect My/My 14 Connect My/My 15 Connect My/My
	0 tems Description ~	Resource	Path	Location	Туре	
< <u> </u>						

Рис. 9. Сгенерированный класс



Рис. 10. Результат запуска программы без аргументов

В папке *src* проекта может размещаться только java-код и файлы с расширением *properties*. Все остальные ресурсы проекта всегда должны находиться в проекте, но вне папки *src*.

При разработке может потребоваться временно или постоянно закомментировать участок кода. Выполняется указанное действия после выделения фрагмента кода нажатием сочетания клавиш **Ctrl**-/. Действия по снятию комментария с выделенного фрагмента осуществляется этим же сочетанием клавиш.

Код јаvа имеет для лучшего визуального восприятия лестничную структуру, вид которой можно изменить в настройках редактора. Но в процессе разработки она может быть нарушена программистом, тогда для придания коду форматированного вида следует нажать сочетание клавиш **Ctrl-Shift-F** или выбрать **Source — Format**.

Для передачи аргументов в метод **main()** необходимо выбрать пункт меню **Run** — **Run Configurations** и далее на вкладке **Arguments** в поле *Program arguments* указать параметры. Параметры разделяются пробелом (Рис. 11.).

Для отладки проект запускается кнопкой F11 клавиатуры или кнопкой Debug панели инструментов (кнопка с изображением жука). Точки остановки (breakpoint) ставятся двойным щелчком мыши на сером поле слева от кода класса. Точка остановки графически выглядит как маленький голубой шарик.

В режиме отладки среда переключается в следующий вид (Рис. 12.).

Create, manage, and run co	onfigurations
Run a Java application	U
Image: Second	Name: HelloWorld Ø Main @>= Arguments
Jin Task Context Test	Working directory:

Рис. 11. Передача параметров в таіп

		Quick Access	발 영 Java EE 륗 Java 錄 E
F Debug 12 생 Servers 🛞 💱 🌣 🗖 🗖	00+ Variables 🐹 🔍 Bi	reakpoints	E) 44 E
HelloWorld (Java Application) Generations and the second	Name p @ args		Value String[2] (id=17)
i Hellevitetd.main(2kmg] Inc. 15 J CAProgram Félet/Javai/jeP/bin/javav.exe (11.01.2013 10.3925)	x		, ,
HeleWorkjøve ☆			tine ∑ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □
Console 😸 🕢 Tasks BloWorld (Java Application) CtProgram Files/Java);re7\bin\javaw.xxx (31.01.2013 10:39:25)		■×後 Ba 5	

Рис. 12. *Debug*

Debug - SampleProject/src/com/epam/sample/main/HelloWorld.java - Eclipse				
File Edit Source Refactor Navigate Search Project Run	Run	Window Help		
	0 0 0	Toggle Breakpoint Toggle Line Breakpoint Toggle Method Breakpoint	Ctrl+Shift+B	
 Debug X & Servers HelloWorld [Java Application] Com.epam.sample.main.HelloWorld at localhost:49159 Thread [main] (Suscended (entry into method main) 		Toggle Watchpoint Skip All Breakpoints Remove All Breakpoints		
		Resume Suspend	F8	
	₹ 22	Disconnect Step Into	Ctrl+F2 F5	
<pre> HelloWorld.java ☆</pre>		Step Over Step Return Run to Line Use Step Filters	F6 F7 Ctrl+R Shift+F5	
<pre>> if (args.length == 0){ System.out.println("Hello, Eclipse! }else{ Sustem.out.println("Hello, Eclipse! }else</pre>	00 \$\$0	Run Debug	Ctrl+F11 F11	
}		Run History Run As Run Configurations	+ +	
Console Tasks HelloWorld [Java Application] C:\Program Files\Java\jre7\bin\javaw.		Debug History Debug As Debug Configurations	> >	
	0	External Tools	•	

Рис. 13. Управление отладкой

Для управления режимом отладки можно использовать пункты меню **Run**, кнопки на вкладке **Debug** или соответствующие быстрые сочетания клавиш на клавиатуре:

F8 — продолжить,
Ctrl+F12 — прервать выполнение программы,
F5 — выполнить инструкцию,
F6 — перейти к следующей инструкции,
Ctrl+R — перейти к строке.

Генерация методов

Среда Eclipse предоставляет много дополнительных возможностей по генерации кода, что существенно сокращает рутинные действия при программировании.

Пусть создан класс Student с полями int id и String lastName. Определение инкапсуляции требует, чтобы поля объявлялись как private, а доступ к ним осуществлялся через методы get*Имя()* и set*Имя()*. Для получения доступа к меню генерации этих методов следует нажать правую кнопку мыши и в открывшемся меню выбрать Source — Generate Getters and Setters. В открытом виде будет представлено меню выбора (Рис. 14.). С его помощью можно выбрать необходимые для генерации методы, отметив их в окне *Select*. Выбрать

Generate Getters and Setters	
Select getters and setters to create:	
 ↓ id ↓ o getId() ○ setId(int) ↓ ♥ □ lastName ♥ ○ getLastName() ♥ ○ setLastName(String) 	Select <u>A</u> ll <u>D</u> eselect All Select <u>G</u> etters Select Setters
Allow setters for final fields (remove 'final' modifier from fields if necess	j sary)
Insertion point:	
After 'lastName'	•
Sort by:	
Access modifier	
⊚ public	
final synchronized	
Generate method <u>c</u> omments	
The format of the getters/setters may be configured on the Code Template	<u>es</u> preference page.
i 3 of 4 selected.	
Э ОК	Cancel

Рис. 14. Генерация getters & setters

место размещения в классе в выпадающем списке *Insertion point*. Задать последовательность размещения в коде в выпадающем списке *Sort by*, а также определить модификаторы доступа для создаваемых методов.

В результате выбора осуществленного в рис. 14 будет сгенерирован код в виде (см. Рис. 15).

Обычной практикой для классов, хранящих информацию представляется автоматическая генерация ме-



Рис. 15. Созданный код getters & setters

тодов equals(), hashCode() и toString(). Для доступа к меню генерации первых двух методов после щелчка правой кнопкой мыши в окне класса в появившемся меню следует выбрать Source — Generate hachCode() and equals(). Задать необходимые поля (по умолчанию используются все поля класса), место расположения методов в коде класса и необходимость добавления комментариев.

Generate hashCode() and equals()	
Select the fields to include in the hashCode() and equals() methods:	
♥ o id ♥ o lastName	Select <u>A</u> ll
Insertion point:	
After 'getId()'	
Generate method <u>c</u> omments	
Use 'instanceof' to compare types	
i 2 of 2 selected.	
? ОК	Cancel

Рис. 16. Генерация equal & hashCode

В результате по правилам языка Java будет сгенерирован код в виде:



Рис. 17. Код сгенерированных методов equal & hashCode

Задать правила генерации метода toString() возможно, выбрав в том же меню Source — Generate toString().

В отличие от обычно неизменяемого после генерации кода методов equals() и hashCode(), код метода toString() часто подвергается изменениям после генерации в соответствии с потребностями разработчика.

Eclipse предоставляет удобный механизм генерации переопределяемых методов суперкласса. Пусть разработан класс GroupStudent, наследующий класс ArrayList в виде:

```
package com.epam.sample.main;
import java.util.ArrayList;
public class GroupStudent extends ArrayList<Student> { }
```

При разработке функциональности класса потребовалось переопределить методы добавления студентов в экземпляр разрабатываемого класса. Для решения этой задачи следует выбрать **Override/Implement Methods** в строке меню **Source** и в открывшемся окне развернуть список доступных для переопределения методов класса **ArrayList**. После чего выбрать необходимые методы из представленного списка.

В результате будет сгенерирован синтаксически корректный код с использованием функциональности переопределяемого метода (Рис. 20.). Разработчику остается только внести в код необходимые ему изменения.

Generate toString()	
Select fields and methods to include in the toString() method:	
 ✓ • Fields ✓ • id ✓ • lastName ✓ • Methods ● getlastName() ● getld() ● hashCode() ● Inherited methods ④^{NF} getClass() ● toString() 	Select All Qeselect All Up Down Soft
Insertion point:	J
After 'equals(Object)'	•
Generate method comments	
Generated code	
String format: <> Default template>	Edit
Code style: String concatenation	Configure
Skip <u>n</u> ull values	
List contents of arrays instead of using native toString()	
Limit number of items in arrays/collections/maps to 10 束	
i 2 of 2 fields and 0 of 5 methods selected.	
ОК ОК	Cancel

Рис. 18. Генерация toString

Override/Implen	nent Methods		
Select methods to	override or implement:	<u>ः</u>	Select <u>A</u> ll
	yList <student></student>	^	Deselect All
♥ ♥	add(int, Student)		
V 0	add(Student)	=	
V 🛛	addAll(Collection extends Student)		
V 🛛	addAll(int, Collection extends Student))	
	clear()		
	clone()		
	contains(Object)		
	ensureCapacity(int)		
	get(int)		
	indexOf(Object)	*	
Insertion point:			
First member			•
Cenerate metho	od <u>c</u> omments		
The format of the	method stubs may be configured on the <u>Co</u>	ode Template	s preference page.
i 4 of 31 selecte	d.		
?		ОК	Cancel

Рис. 19. Генерация переопределяемых методов

```
public class GroupStudent extends ArrayList<Student>{
   @Override
   public void add(int arg0, Student arg1) {
       // TODO Auto-generated method stub
       super.add(arg0, arg1);
   3
   @Override
   public boolean add(Student arg0) {
       // TODO Auto-generated method stub
       return super.add(arg0);
   }
   @Override
   public boolean addAll(Collection<? extends Student> arg0) {
       // TODO Auto-generated method stub
       return super.addAll(arg0);
   }
   @Override
   public boolean addAll(int arg0, Collection<? extends Student> arg1) {
       // TODO Auto-generated method stub
       return super.addAll(arg0, arg1);
   }
```

Рис. 20. Созданный код переопределяемых методов

Если же класс не наследует, а агрегирует свойства другого класса, как например:

Тогда для реализации необходимых разработчику методов с использованием имен и, возможно, функциональности агрегированного класса следует в меню **Source** выбрать **Generate Delegate Methods**. Последующие действия аналогичны действиям при генерации переопределенного кода.

Генерация конструкторов

Важную роль в коде практически любого класса выполняют конструкторы. При генерации конструкторов следует принимать во внимание особенности процесса создания экземпляра класса, для которого предназначен тот или иной конструктор. Основное назначение конструктора — инициализация полей. Для генерации конструктора на примере класса **Student** с полями **id** и **lastName** следует выбрать **Source** — **Generate Constructor using fields**.

В предложенном окне следует выбрать поля, которые будут использованы в качестве параметров создаваемого конструктора. Результатом будет один

Generate Constructor using Fields	- • ×
Select super constructor to invoke:	
(Object()	•
Select fields to initialize:	
V o id V o lastName	Select <u>All</u> <u>D</u> eselect All <u>Up</u> Down
Insertion point:	
After 'lastName'	-
Access modifier	
Generate constructor <u>c</u> omments	
Omit call to default constructor super()	
The format of the constructors may be configured on the <u>Code Template</u>	s preference page.
i 2 of 2 selected.	
ОК ОК	Cancel

Рис. 21. Генерация конструктора на основе полей класса

конструктор. Если необходим конструктор с другим набором параметров, операцию создания следует повторить.

Если класс наследует другой класс, то при разработке его конструкторов может потребоваться обращение к конструкторам суперкласса. На примере класса GroupStudent, выбрав в списке Source — Generate Constructors from Superclass, будет получено окно, в котором будет представлен список конструкторов суперкласса, доступных для использования в конструкторе подкласса. Выбрав необходимый набор и нажав кнопку OK, будет сгенерировано столько конструкторов, сколько было отмечено.

Generate Constructors from Superclass	
Select constructors to implement:	
Image: Construction of the second state of the second s	Select <u>A</u> ll
Insertion point:	
First member	-
Access modifier © public © protected © d <u>e</u> fault © pri <u>v</u> ate	
Generate constructor <u>c</u> omments	
<u> O</u> mit call to default constructor super()	
The format of the constructors may be configured on the <u>Code Templat</u>	es preference page.
i 3 of 3 selected.	
ОК	Cancel

Рис. 22. Генерация конструкторов с использованием суперкласса

Code assist

Java редактор предоставляет возможность разработчику создавать код быстрее, предоставляя ему возможности *code assist*. Пусть в методе **main()** необходимо создать код, использующий возможности класса **System**. Достаточно набрать в редакторе

System.

Сделать небольшую паузу, и автоматически откроется окно code assist с предложением выбора статических полей и методов класса **System** (Рис. 23.). Как только выбор будет осуществлен вертикальным перемещением по списку или набором на клавиатуре начальных символов необходимого метода или поля с последующим выбором из списка, результаты выбора появятся в редакторе кода. Вызов code assist можно также осуществить нажатием клавиш **Ctrl-Space**.

public class H	elloWorld {	
public sta	<pre>stic void main(String[] args) {</pre>	
System	3.	
}	o ^S class : Class < java.lang.System >	*
}	SF err : PrintStream - System	
·	SF in : InputStream - System	E
	SF out : PrintStream - System	
	S arraycopy(Object arg0, int arg1, Object arg2, int	arg3, int a
	S clearProperty(String arg0) : String - System	
	S console() : Console - System	
	S currentTimeMillis() : long - System	
	Sexit(int arg0) : void - System	
	^S gc(): void - System	
	Setenv() : Map <string,string> - System</string,string>	-
	< [III]	•
	Press 'Ctrl+Space' to show Templ	ate Proposals

Рис. 23. *Code assist*

Шаблоны

Eclipse поддерживает короткий набор части кода с возможностью генерации части конструкции. Для этого применяется набор так называемых templates, созданных в самой среде.

Демонстрация использования шаблонов на примере циклов. В методе набирается слово for и нажимаются клавиши Ctrl-Space. Появляется меню с предложением выбрать один из четырех циклов языка или классов, начинающихся на символы, набранные в редакторе (Рис. 24.). В правой части окна располагается пример кода, который будет сгенерирован после нажатия кнопки Enter. После того, как код будет создан, шаблон предоставляет возможность изменения имени переменной цикла.

Список всех доступных для применения шаблонов можно получить, выбрав Window — Preferences — Java — Editor — Templates.

Разрешено создавать собственные шаблоны. Создание шаблона начинается с нажатия кнопки **New** (Puc. 26.).

Шаблоны, в том числе и стандартные, разрешено редактировать. Сохранить и перенести авторский набор шаблонов на другую версию Eclipse можно, применяя кнопки **Export/Import** в окне, представленном на рис. 25.

Рефакторинг

Созданный код может потребовать исправления. Чаще всего разработчику необходимо переименовать пакет, класс, метод, поле, переменную или константу. Причем требующее изменения понятие, например метод, может быть



Рис. 24. Вызов шаблона



Рис. 25. Список доступных шаблонов

🔘 New Temp	late		
<u>N</u> ame:		<u>C</u> ontext: Java	▼ ✓ Auto <u>m</u> atically insert
Description:			
<u>P</u> attern:			•
	1		*
	•		
	Insert Variable		
🔇 Templat	e name cannot be empty.		
?			OK Cancel

Рис. 26. Меню создания шаблона

многократно использовано в коде. В обычной ситуации разработчику придется перебирать все классы, в которых осуществлялся вызов такого метода и исправлять его имя, что может оказаться весьма трудоемкой задачей.

Eclipse позволяет изменить имя класса, метода, поля или переменной, распространяя изменения на все обращения к изменяемому понятию, в том числе и из других классов и пакетов приложения.

Чтобы осуществить изменение имени, следует двойным щелчком левой клавиши мыши выделить имя, требующее изменения, после чего правой кнопкой вызвать список, выбрать в нем Source — Refactor — Rename, ввести новое имя и нажать Enter для фиксации изменений в приложении.

При рефакторинге метода может понадобиться не только изменение его имени, но и корректировка всей сигнатуры. Для этого следует выделить метод и выбрать **Source — Refactor — Change Method signature** в результате чего, например, для метода **void setLastName(String lastName)** будет выведена форма редактирования. В этом окне кроме имени можно изменить тип возвращаемого методом значения, спецификатор доступа к методу. Список аргументов метода может быть увеличен добавлением новых аргументов или сокращен при необходимости. Имена аргументов также можно изменять. Предварительный вид сигнатуры будет отображаться в строке *Method signature preview*. Изменения коснутся также всех переопределенных версий метода во всех подклассах.

Рефакторинг предоставляет много других возможностей при работе с классом и его составляющими.

Change Metho	d Signature		
A <u>c</u> cess modifier:	Return <u>t</u> ype:	Method <u>n</u> am	e:
public	void	setLastNam	e
Parameters Ex	ceptions		
Туре	Name	Default value	Add
String	lastName	-	<u>E</u> dit
			Re <u>m</u> ove
			Up
			Down
Eeep original r Mark as de Method signature public void s	nethod as delegate to o precated : preview: etLastName(String	changed method	
i Change the si	gnature of the selected	I method and all its overrid	ding methods.
	Prev	/ie <u>w</u> > OK	Cancel

Рис. 27. Рефакторинг сигнатуры метода

Подключение библиотек

При разработке проектов кроме стандартных библиотек Java необходимы библиотеки сторонних разработчиков, например: Log4J, JUnit, Xerces и прочие. Чтобы воспользоваться функциональностью классов из этих библиотек, их необходимо подключить к проекту. Сначала требуется загрузить jar-архив библиотеки с определенного интернет-ресурса и разместить его на диске, желательно в директории, находящейся внутри основной директории проекта, хотя в общем случае это необязательно. Для включения библиотеки в ресурсы проекта необходимо щелкнуть правой кнопкой мыши по проекту и из выпавнего списка выбрать **Build Path** — **Configure Build Path**.

В появившемся окне следует выбрать закладку Libraries и нажать кнопку Add External JARs (Рис. 28.). С помощью формы файловой системы найти необходимый jar-архив, например, log4j-[*версия*].jar и нажать кнопку **Open**. После чего указанная библиотека появится в списке подключенных библиотек.

Библиотека Log4J будет видна при открытии папки *Referenced Libraries*.

Теперь классы библиотеки можно использовать в разрабатываемом проекте.

Properties for SampleProject		
type filter text	Java Build Path	⇔ • ⇔ • •
 Resource Builders Java Build Path Java Code Style Java Compiler Java Editor Javadoc Location Project Facets Project References Run/Debug Settings Task Repository Task Tags Validation WikiText 	Source Projects Libraries Order and Export JARs and class folders on the build path:	Add JARs Add External JARs Add Variable Add Libr <u>a</u> ry Add Class Folder Add External Class Folder Edit <u>E</u> dit <u>R</u> emove <u>M</u> igrate JAR File
?		OK Cancel

Рис. 28. Подключение внешней библиотеки



Рис. 29. Проект с подключенной библиотекой

Подключение plugin-ов

IDE Eclipse представляет собой настраиваемую систему, возможности которой можно наращивать и улучшать. Добавление новых возможностей производится за счет подключения plugin-ов. Plugin представляет собой некую небольшую

IDE Eclipse

🙆 Add Re	pository	
<u>N</u> ame:		L <u>o</u> cal
Location:	http://	Archive
?		OK Cancel
?		OK Cancel

Рис. 30. Проект с подключенной библиотекой

программу, для подключения которой следует выбрать **Help** — **Install New Software** и в появившейся форме нажать кнопку **Add** (Рис. 30.). В поле *Name* указать имя плагина, в поле *Location* адрес URL его загрузки и запустить процесс. Затем плагин будет закачан на диск разработчика и подключен, после чего сразу становится доступным для использования.

Существуют еще три способа подключения плагинов, если они уже находятся на диске разработчика. Если плагин представлен в виде архива, то следует на форме нажать кнопку **Archive** и выбрать искомый архив с плагином. Если плагин представлен в виде набора папок с файлами, то следует выбрать **Local** и указать на корневую папку плагина. Два последних способа могут не сработать, так как плагины, разработанные к разным версиям Eclipse, могут быть несовместимы с представленной версией. Есть еще один кустарный способ подключения плагинов прямым копированием в папку plugins самого IDE Eclipse. Несовместимость версий в этом случае может привести к тому, что плагин может быть вообще не представлен в функционале IDE или работать некорректно.

Совместимость версии плагина с версией Eclipse следует проверять на сайте разработчика плагина или на eclipse.org.

Импорт/экспорт проектов

В процессе разработки проект может переноситься с одного компьютера на другой. Для переноса проекта следует правым кликом на проекте вызвать его меню и выбрать **Export** — **Java** — **JAR file** — **Next** (Рис. 31.). В открывшейся форме обязательно пометить check box с надписью *Export Java source files and resources*. Иначе архив, полученный в результате экспорта, не будет содержать файлы с исходным кодом проекта. В поле *JAR file* следует указать месторасположение и имя jar-архива и нажать кнопку **Finish**. После чего можно забрать созданный архив и переносить на другой компьютер.

Экспорт можно выполнить, и в директорию на диске для этого необходимо выбрать Export — General –File System — Next. В поле *To directory* открывшейся

JAR Export	
JAR File Specification	8
Define which resources should be exported into th	ne JAR.
Select the resources to <u>e</u> xport:	
 ▲ □ SampleProject ▷ ♥ Src □ ▷ .settings 	 ✓ X .classpath ✓ X .project
Export generated class files and resources Export all output folders for checked projects Export Java gource files and resources Export refactorings for checked projects. <u>Select</u>	t refactorings
Select the export destination:	
JAR file:	Browse
Options: Compress the contents of the JAR file Add directory entries Qverwrite existing files without warning	
? < <u>B</u> ack	Next > Einish Cancel

Рис. 31. Экспорт проекта

формы ввести путь, по которому после выполнения экспорта можно будет забрать директорию с файлами проекта.

Для импорта проекта в Eclipse следует сначала создать новый java проект, например, с именем *SampleProjectCopy*. После чего выбрать **File** — **Import** — **General** — **Archive file** (Рис. 32.). Далее выбрать *SampleProjectCopy/src* в поле *Into folder*. В поле *From archive file* выбрать jar-архив импортируемого проекта и нажать кнопку **Finish**.

Все исходные коды импортируемого проекта окажутся в папке *src* проекта *SampleProjectCopy*. Существуют и другие способы импорта.

💽 Import	- • X
Archive file Source file is not a valid Zip file.	
From archive file:	Browse
Filter Types Select All Deselect All Into folder: SampleProjectCopy/src Overwrite existing resources without warning	Browse
() (Back Next > Einish	Cancel

Рис. 32. Импорт проекта

Создание, запуск и отладка веб-проекта

Перед созданием динамического веб-проекта необходимо настроить сервер java-приложений. Для этого в меню **Window** — **Preferences** выбирается в дереве слева **Server** — **Runtime Environments**. Чтобы добавить application server, например, Apache Tomcat необходимо нажать кнопку **Add**...

После нажатия кнопки Finish в списке серверов появится только что добавленный сервер. Теперь необходимо выбрать меню File — New — Dynamic web project, задать имя проекта, выбрать сервер. Можно задать некоторые дополнительные параметры, нажимая кнопку Next, можно оставить все остальное по умолчанию и нажать кнопку Finish для создания проекта (Рис. 35.). Созданный проект появится в Project Explorer (Рис. 36.).

Далее следует нажать Next, и на следующей форме также нажать Next. На полученной форме конфигурирования веб-модуля выбрать *Generate web. xml deployment descriptor*. Наличие файла web.xml необходимо для задания важных параметров инициализации веб-приложения.

Для завершения создания веб-проекта следует нажать **Finish** и в закладке *Project Explorer* появится проект.

O New Server Runtime Environment	
New Server Runtime Environment Define a new server runtime environment	I
Downloa Select the type of runtime environment:	d additional server adapters
type filter text	
Apache Apache Tomost v3.2 Apache Tomost v4.0 Apache Tomost v4.0 Apache Tomost v4.1 Apache Tomost v5.0 Apache Tomost v5.0 Apache Tomost v5.0 Apache Tomost v7.0 Create a new local server	EE 5 and 6 Web modules.
	Finish Cancel

Рис. 33. Выбор сервера для добавления



Рис. 34. Определение директории сервера

New Dynamic Web Project		
ynamic Web Project		1
Create a standalone Dynamic Web project or add it to a new or ex	isting Enterprise Applic	ation.
Project name: Samola Dunamio Project		
rioject name. Joanpie Synamerioject		
Project location		
Vise default location		d
Location: C:\Program Files\Java\SampleDynamicProject1		Browse
Target runtime		
Apache Tomcat v7.0		New Runtime
Dunamic web module version		
13.0		
Configuration		
Default Configuration for Apache Tomcat v7.0		Modify
A good starting point for working with Apache Tomcat $v7.0$ runtii installed to add new functionality to the project.	me. Additional facets ca	an later be
EAR membership		
Add project to an EAR		
EAR project name: EAR	<u></u>	New Project
Working sets		
Add project to working sets		
Working sets:	*	Select
2	11	
2 Angle New	t > Einish	Cancel

Рис. 35. Создание нового веб-проекта

New Dynamic Web Project	
Web Module Configure web mo	odule settings.
Context <u>r</u> oot:	SampleDynamicProject
Content directory:	WebContent
Cenerate web.xn	nl deployment descriptor
?	< <u>B</u> ack Next > <u>Finish</u> Cancel

Рис. 36. Создание нового веб-проекта



Рис. 37. Вид веб-проекта в Project Explorer

Для запуска веб-проекта необходимо нажать правой кнопкой мыши на название проекта в **Project Explorer** и выбрать пункт **Run As** — **Run On Server**. Для отладки: **Debug As** — **Debug On Server**. Управление процессом отладки такое же, как в консольном проекте.