DTP: Overview and Direction

John Graham
DTP PMC Chair
Sybase, Inc.

Introduction: Project

- Eclipse Data Tools Platform Project (DTP)
- Top level project
- Proposed in February, 2005
- Contributors
  - Sybase
  - IBM
  - Actuate

Introduction: Speaker

- John Graham (john.graham@sybase.com)
- DTP PMC Chair
- Senior Staff Software Engineer, Sybase
- Building products on Eclipse since version one
- Participated in Eclipse community from early on
- Academic background: natural and formal language theory
  - Thesis on formal computational properties of natural language
  - Worked on database, XML, EAI technologies
- Have lead DTP for Sybase since project proposal

DTP Release History

- First release, 0.7, around EclipseCon 2006
- Second release, 0.9, with Callisto (29-June)
- Maintenance release, 0.9.1, with Callisto (29-September)

DTP Release Horizon

- DTP 1.0 scheduled for end of 2006
- DTP 1.5 planned for ‘Europa’ (2007 June)

DTP Approach

- Conservative
- Emphasis on getting frameworks in place
- Secondary emphasis on tools
- Examples around relational databases
  - Apache Derby exemplary clients
  - SQL Development Tools project
- Active community encouragement through Enablement project
- Future: Leverage foundation
  - More tools (still extensible)
  - More contributions to Enablement beyond relational

DTP Use Cases

So, what can it do for me?

Use Cases: Model Base

- Provides a *model driven design* for relational database technologies
- Based on EMF
- Currently have models for
  - SQL (based on SQL:99/03 specifications)
  - Database Definition
  - SQL Query (including XML support)
- Use these if you want to interact with these areas through their problem domain, not that of the underlying provider (eg. JDBC)

Use Cases: Connectivity

- Centralize definition of data source connector (driver) components to one place in Eclipse platform instance
- Centralize connection definition and creation to one place in Eclipse platform instance
- Allow tools to access a know, abstracted set of *connections*
- Don't force the users to keep defining connections for every tool requiring access to a data source!
- Tools should share connection definitions, since this allows for increased integration and ease of use
- Currently, this functionality is repeated and scattered across Eclipse.org projects (and, of course, projects outside Eclipse.org)

Use Cases: Connectivity ODA

- Open Data Access (ODA)
- Migrated from BIRT
- Provides framework, runtime and design tool support
- Abstraction over notion of *data type*
- Write and contribute ODA drivers
- Enablement project's XML driver is an example from BIRT

Use Cases: SQL Development Tools

- SQL Editor framework for writing SQL editors
- SQL Results view framework
- Example tools
  - SQL editor
  - Results view, with query history
- Support for Apache Derby at the moment
- More support coming in Enablement project
- SQL Query Parser
  - Based on SQL Query model
  - Extensible SQL parser

DTP Direction

What does the future promise?

Enablement Contributions

- Provide support for specific data sources
- For relational databases
  - Sybase
  - IBM
- For XML ODA, Actuate/BIRT
- Others pending…

Visual SQL Query Builder

- Enhanced editor support for building SQL queries (DML)
- Increased graphical support for building SQL queries
- Seeding contribution from IBM, participation from
  - Sybase
  - Actuate
  - Others?
- “Tech Preview” drop to be included with DTP 1.0
- Build out frameworks and tools for DTP 1.5
- Large, interesting, demanding new component for DTP

DTP Challenges

What makes it hard?

Incumbent Technologies

- Components for databases are popular
  - Eclipse based
  - Open source other than Eclipse
  - Commercial
- Switching costs
  - What does DTP provides above and beyond these incumbents?
  - How much will it cost to make the switch?
- Adoption curve
  - Still in early stages
  - Hasn't penetrated majority yet
  - Lacking network effects currently

Database Tooling Cultural Attitude

- It has been done…
- Or at least done well enough for more everyone
- The basic problems are solved
- The really hard problems, if any, are for academics or specialized vendor products
- NO!
  - Are you satisfied with database tooling in Eclipse?
  - Is it as easy and full-featured as, at least, the JDT is for Java?
  - What about large databases? Streams? RFID? Query optimization?
  - It is *not* a solved, completed problem!

Range of Contributions

- General Eclipse.org characteristic: Product Preservation
  - Organizations want to leverage Eclipse-based OSS for their products
  - Avoid having to rewrite/rewire as much as possible
- Results in
  - Duplication of effort
  - Inflexibility to change based on technology advances
- Have to balance existing adopter with potential adopter concerns

Product Preservation: Striking a balance

- Need a diverse, vocal, involved community
- Any open source project is strong only by the involvement of its community
- Participation
  - Raise requirements
  - Register bugs
  - Contribute
- But, most importantly: stick to it
- The presence of a vocal, involved and committed community pulls against the Product Preservation tendency

Thank You!

Questions?

