FirstSpirit™

Your Content Integration Platform

UX-Bridge Whitepaper

Version	**1.2**
State	**RELEASED**
Date	**2013-02-19**
Department	Product Management
Author/ Authors	Andreas Knoor
Copyright	2012 e-Spirit AG
File name	UX-Bridge Whitepaper_DE

e-Spirit AG

Barcelonaweg 14
44269 Dortmund | Germany

T +49 231 . 477 77-0
F +49 231 . 477 77-499

info@e-Spirit.com
www.e-Spirit.com

e-Spirit AG

Table of contents

1 Introduction and vision

1.1 Websites and user experience

The requirements for modern websites have grown rapidly in recent years. This dynamic is increasingly related to the functionality of the website, which is more often being realized as an interactive communications medium with customers or prospects, as well as to the optimization of the editorial processes.

The requirements with regard to control of this customer communication are becoming more demanding, because consumer expectations are at a very high level. The demanding customer expects an excellent user experience. This term encompasses a multitude of aspects such as good usability, a target group-specific message and delivery of information at the right point in time in the correct granularity. In addition, the performance of a website is gaining in importance, in user acceptance as well as in aspects such as search engine ranking.

1.2 Hybrid architecture

In order to meet the increasingly stringent requirements, an adequate technical solution on the part of the Content Management System (CMS) is necessary. The wide spectrum of user experience requirements makes an unusually flexible architecture of the CMS essential. This particularly applies to aspects that represent the classic oppositional goals, such as:

- High dynamics of the content versus performant delivery of the website
- Stable system architecture versus integration of many different tools

In order to offer an optimum solution for the respective customer despite these trade-offs, FirstSpirit provides a particularly adaptable architecture, called "hybrid architecture".

The hybrid architecture of FirstSpirit makes it very easy, especially in integration-heavy scenarios, to establish a solution that fits exactly and which takes into account the specific properties and requirements of the website. Experience has shown that a one-size-fits-all architecture works only in very simple use cases. As the number and complexity of websites increases, however, the variety of third-party systems to be integrated also increases. Precisely in larger companies, this is a feature that is encountered frequently, because alongside the CMS component, additional systems such as e-commerce shops, enterprise portals, but also self-developed web applications are prescribed and are to be integrated. In such cases, the hybrid architecture of FirstSpirit enables a delicate balance between performance, stability, and maintainability. Most importantly, this balance is customized to the project.

With its hybrid architecture, FirstSpirit follows the rule of:

"Providing as much dynamics as necessary; pre-generating as much content as possible."

We understand a hybrid CMS to be a content management system which gives every customer the freedom to choose which elements of a website are fully dynamic and which elements are delivered already pre-generated.

Technically, the two variants mean:

- Fully dynamic: At the time the user accesses a website (request time), the editorial content is read out and put together live from one or more content repositories.

- Pre-generated: The editorial content is already physically contained in a website (for example, an HTML file) and can be delivered directly to the user.

A particularity of FirstSpirit's hybrid approach lies in the fact that the decision in favor of one variant or another can be established in a fine-grained manner for each element of an individual page. Concrete examples for this type of architecture are explained in Chapter 2.2.

1.3 UX-Bridge: Connecting Content and User Experience

With the trend toward more interactive websites, dynamic access to editorial content has a steadily increasing importance. Wherever pre-generation of content is not possible, the CMS content has to be accessed dynamically. Here, "dynamic" means that the content can change for each website user and at any point in time. Any information that is on the website longer than 1 to 2 minutes and is identical for all users (for example, an article in a news portal) is *not* considered dynamic here, rather as potentially pre-generated. Nevertheless, the dynamic scenarios are increasing, as personalized content delivery can make an important contribution to the user experience.

With the UX-Bridge module, FirstSpirit offers an infrastructure for the requirement of a dynamic content delivery platform. Consequently, the module expands the hybrid approach by adding a standard infrastructure for dynamic content delivery. As a rule of thumb for the two hybrid architecture variants, we can stipulate that:

- Pre-generated content delivery of editorial content:
  → FirstSpirit Standard Deployment

- Fully dynamic content delivery of editorial content:
  → UX-Bridge

It is important to note that these decisions are not global for the entire website, but can be made individually for every type of content element (see examples in Chapter 2.2).

When using UX-Bridge, it must be decided per project and per (fine-grained) use case whether the dynamization is functionally and technically necessary or logical (see the above discussion of the trade-off between dynamics and performance). Based on this decision, intelligent and use case-optimized solutions can be implemented very easily.

1.4 Connection to the customer through interaction

Pure content delivery is insufficient to enable true interaction with the users of a website, even if it is personalized and done in a context-oriented manner. Rather, there are a multitude of use cases where information from the website has to flow back in the direction of content creation / CMS. Examples of use cases for this kind of return channel include user-generated content (UGC) or statistical data with regard to user behavior, which can be reused in a second step to optimize content delivery.

With the aid of UX-Bridge, you can store UGC content and read it back out via a standard path. At the same time, UX-Bridge can be used to allow certain information to flow back directly into the editing system from the website (for example, how many comments an article has received) in order to further process them there appropriately, such as for ranking lists.

1.5 Mobile First

The use of mobile end devices to display (web) content has been increasing exponentially for several years. It has now almost become a standard requirement to support a mobile presentation channel for smartphones or tablet PCs alongside a "normal" website. The "Mobile First" approach indeed goes a step further, and recommends that the mobile variant of a website be developed even before the conventional variant. The reasons are that, in the mobile channel, you have to restrict the scope and focus significantly more on the important points, in other words, the content that is relevant to the customer. Furthermore, alongside some technical limitations on mobile devices (for example, screen size), there are also advanced properties such as location tracking via GPS, gesture-based control, changing aspect ratios between portrait and landscape formats or offline capability. These have to be taken into account as soon as the initial design of a website, because they are very difficult to add later without a complete redesign.

By strictly separating content and layout, FirstSpirit makes it very easy to create content fragments that conform to mobile standards such as XML or HTML fragments.

UX-Bridge supplements this approach by adding flexible content delivery infrastructure, which optimally supports the requirements of displaying content for mobile devices. A specific example scenario is described in Chapter 2.2.6.

1.6 New requirements: big data, NoSQL, realtime analytics

Modern websites place ever higher requirements on the underlying technical infrastructure.

The quantities of data to be managed are increasing by leaps and bounds because, alongside classic editorial content, increasing amounts of user-generated content (UGC) and statistical data from the website users' behavior has to be managed and evaluated. Saving and managing such large quantities of data is also associated with the term "big data".

The increasing quantity of data is accompanied by a diversification of data types at the same time. Websites strongly driven by editorial content frequently consist of unstructured data, while website data from backend systems tends to be strongly structured (e.g. product information). User-generated content, such as comments or evaluations, is usually structured very simply and flat. Dependencies or relationships (who viewed what, who knows whom, etc.) reflect an additional type of user data. This "social graph" is a particularly important source of information for user interaction in online communities.
An additional use case for relationships between content elements is the so-called "semantic web". Contents are placed in relation to each other there by explicitly modeling connections, such as for improved search options.
For this reason, new types of data storage will play an ever-greater role in the future. These include NoSQL or graph databases.

Along with saving editorial and user-specific data, evaluating this data is becoming more and more important. The trend for data analysis is trending increasingly towards realtime reporting. Increasingly, methods from data warehousing and business intelligence are being used to arrive at the relevant analysis.

The mentioned constraints must be taken into account when drafting future-compatible architecture for a website:

- Constantly increasing data quantities
- Managing very different types of data
- Increasing requirements for evaluating data (reporting).

It is worth noting that classic content management repositories were not designed for these extensive requirements at all. From the trade-offs of the requirements, it is apparent that a "one size fits all" approach for the persistence of data is not feasible over the long-term (see Chapter 1.2).

FirstSpirit UX-Bridge therefore follows a new approach: The UX-Bridge architecture enables a flexible selection of data storage (persistence) for dynamic access, depending on the functional and technical requirements (see Chapter 2.1.3.2).

1.7 Cloud-ready solution

UX-Bridge was designed from the beginning so that the component can readily be used in the cloud without issue. Aspects such as

- Scaling and performance
- Failsafe reliability/failover
- Operating costs and total cost of ownership (TCO)

are good indicators for operating UX-Bridge in a cloud infrastructure such as Amazon. The same is also true for operating the complete website, and of course, for the FirstSpirit editing system as well. Like all architecture variants, this is an *optional* possibility which has to be checked for suitability in the respective customer and project situation.

1.8 Content integration platform

Alongside dynamic delivery of editorial content to the website, transferring and forwarding content to third-party applications is an important implementation scenario for UX-Bridge. Examples of this use case include:

- Transferring content to a self-developed web application (such as a product configuration or a branch search feature)
- Transferring content to Cloud/SaaS applications that are intended to be integrated into the website and work on / with editorial content (such as an external recommendation engine or a search engine)

While the FirstSpirit editing system functions as a content-integration platform for the backend, UX-Bridge takes on this role on the live website.

2 Architecture description

UX-Bridge is an extension of the classic, pre-generated FirstSpirit architecture. In contrast to a "fully dynamic CMS", FirstSpirit assumes that the content is compiled in the editing system (backend) and (in many cases) is provided with a suitable target layout as part of what is termed "generation". The results are then, finished HTML pages, for example. Then the generated files are transferred (deployed) to the live system, such as a web or application server. FirstSpirit serves as the content integration layer for editorial content and various connected backend systems (see Figure 1).

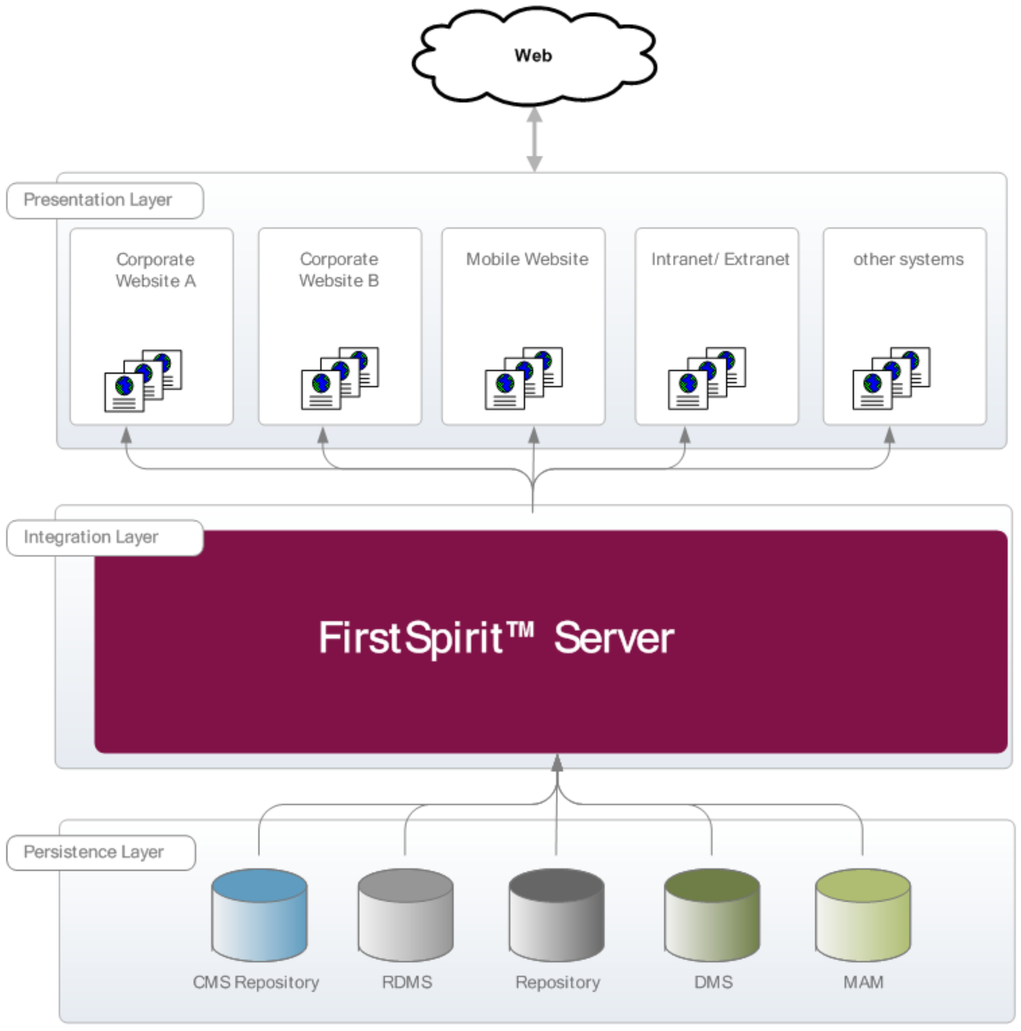


Figure 1: FirstSpirit architecture

2.1 Architecture overview of UX-Bridge

As part of the hybrid architecture, UX-Bridge expands FirstSpirit by adding dynamic content delivery using a standard medium. Other (UX-Bridge-free) alternatives to dynamic content delivery are, incidentally, also expressly possible. First, however, a review should be carried out to determine whether or not a standard-based approach such as UX-Bridge would be more beneficial.

Unlike the usual scenario, in the UX-Bridge architecture, those content elements that are to be read out dynamically via a web application are not stored on the web server as finished (HTML) files. Instead, what is called a live repository takes the relevant data from FirstSpirit and forwards it to the dynamic web applications (see Figure 2).

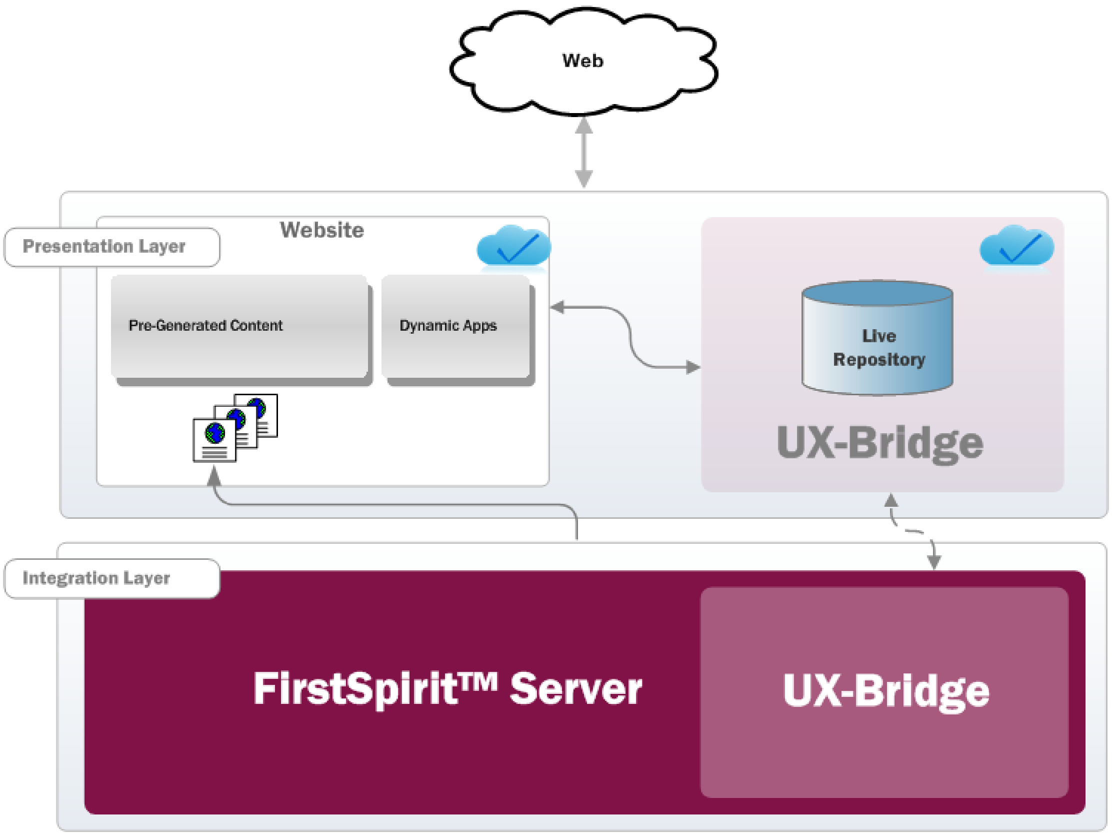
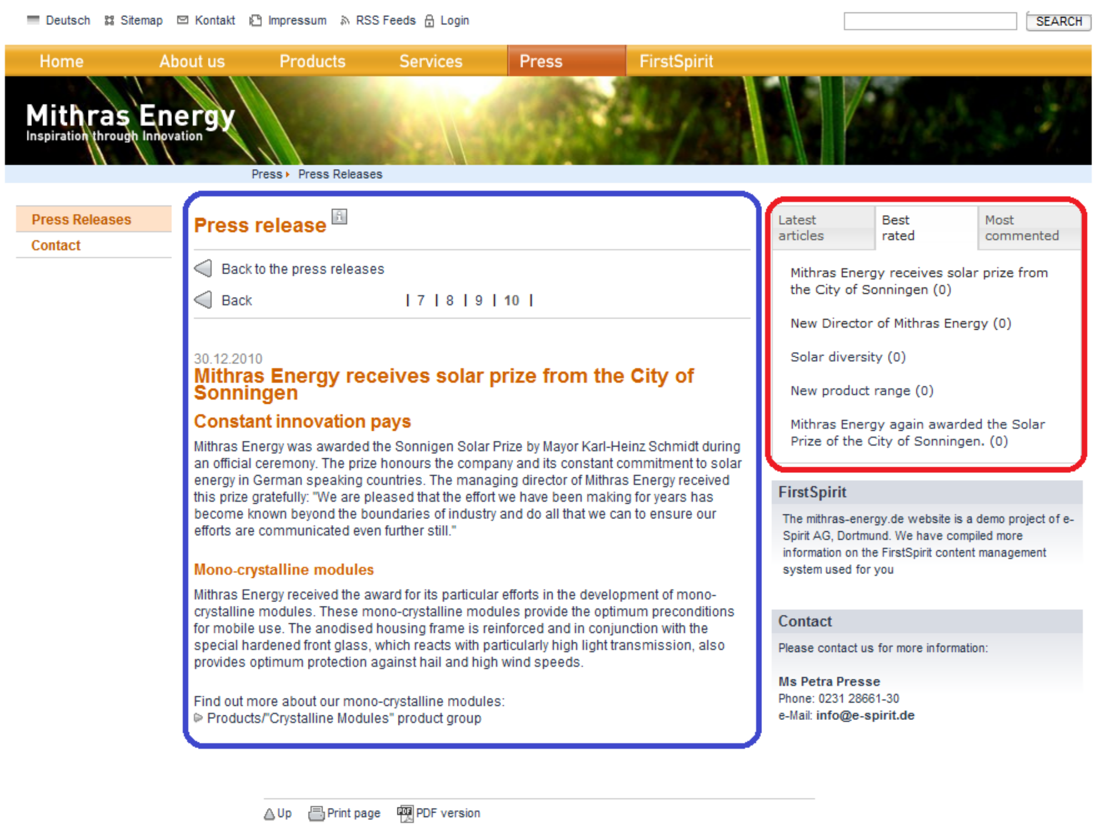


Figure 2: Pre-generation in combination with UX-Bridge

It is important that both the approaches of pre-generated files (left side of the graphic) and live repository (right side of the graphic) are usually used in combination with each other (see Figure 3).

In this way, in a news portal, the normal press release can be pre-generated entirely statically (as a finished HTML fragment), while on the same page, a "Top Rating

Widget" in the sidebar shows the press releases that have been rated the best "live" (bordered in red in the illustration).



Press release

Back to the press releases

Back | 7 | 8 | 9 | 10 |

30.12.2010

Mithras Energy receives solar prize from the City of Sonningen

Constant innovation pays

Mithras Energy was awarded the Sonnigen Solar Prize by Mayor Karl-Heinz Schmidt during an official ceremony. The prize honours the company and its constant commitment to solar energy in German speaking countries. The managing director of Mithras Energy received this prize gratefully: "We are pleased that the effort we have been making for years has become known beyond the boundaries of industry and do all that we can to ensure our efforts are communicated even further still."

Mono-crystalline modules

Mithras Energy received the award for its particular efforts in the development of mono-crystalline modules. These mono-crystalline modules provide the optimum preconditions for mobile use. The anodised housing frame is reinforced and in conjunction with the special hardened front glass, which reacts with particularly high light transmission, also provides optimum protection against hail and high wind speeds.

Find out more about our mono-crystalline modules:
▷ Products/"Crystalline Modules" product group

Mithras Energy receives solar prize from the City of Sonningen (0)

New Director of Mithras Energy (0)

Solar diversity (0)

New product range (0)

Mithras Energy again awarded the Solar Prize of the City of Sonningen. (0)

FirstSpirit

The mithras-energy.de website is a demo project of e-Spirit AG, Dortmund. We have compiled more information on the FirstSpirit content management system used for you

Contact

Please contact us for more information:

Ms Petra Presse
Phone: 0231 28661-30
e-Mail: **info@e-spirit.de**

Figure 3: Pre-generation and dynamics

The hybrid architecture of FirstSpirit enables any desired mixture of such pre-generated and completely dynamic content.

2.1.1 Update content

To stay with the news scenario described above, then the website will be updated via the following steps:

1. An editor writes a new press release, which is then released via a workflow.

2. The last workflow step consists of the deployment of the press release, which can be further subdivided:

   a. The press release detail page is completely pre-generated (as static HTML) and then transferred to the web server. The same is true for the news overview page, on which the new notification should appear as the first message.

b. The data relevant for dynamic display of the press release (here, header for the news item) is generated by FirstSpirit and transferred directly to the live repository without generating a file. The data format can be any one desired, while the reference implementation uses XML, which can contain HTML fragments.

2.1.2 Output content

After updating the content, a website user calls up the press overview page, which contains the latest message as the first link. Clicking on this link opens the detail page just updated (see Figure 3). To the right, in the sidebar of the page, is a dynamic WebApp, which takes up contact to the live repository at the time of the request and reads out and shows the Top News.

The architecture does not prescribe the technical mechanisms (for example, via which web framework) with which the WebApp reads out its content from the live repository. Here, any fitting frameworks and protocols can be used.

2.1.3 UX-Bridge architecture in detail

Now that the mechanism of the content update with UX-Bridge has been outlined, this chapter will explain the technical details of the module more closely (see Figure 4: UX-Bridge in detail).

On the live side (the presentation layer), UX-Bridge consists of three components:

- UX-Bus (or Content-Bus):
  UX-Bus forms the central infrastructure component in order to distribute content from FirstSpirit into one or more live repositories. The reference implementation of UX-Bus is technically based on ActiveMQ (see http://activemq.apache.org/). At the same time, UX-Bus can also be used to provide third-party systems with relevant FirstSpirit content (for example, a search index or a recommendation engine). A third use for the UX-Bus is the construction of a back channel from the website in the direction of the FirstSpirit backend (see Chapter 1.4).
  UX-Bus forms the central integration component on the live website (see Chapter 1.8), with which FirstSpirit can interact with the repositories and WebApps, but also with which the WebApps can interact with each other. With the aid of UX-Bus, all participating components can exchange data and events.
  Technically, UX-Bus is based on Message Oriented Middleware (MOM). In this way, the UX-Bridge components are only loosely coupled to one another.

Simultaneously, scalability is guaranteed through asynchronous communication and the use of queues. To do so, the UX-Bus reference implementation uses ActiveMQ (see http://activemq.apache.org) as a message broker and Apache Camel (http://camel.apache.org/) for the routing and conversion of the messages.

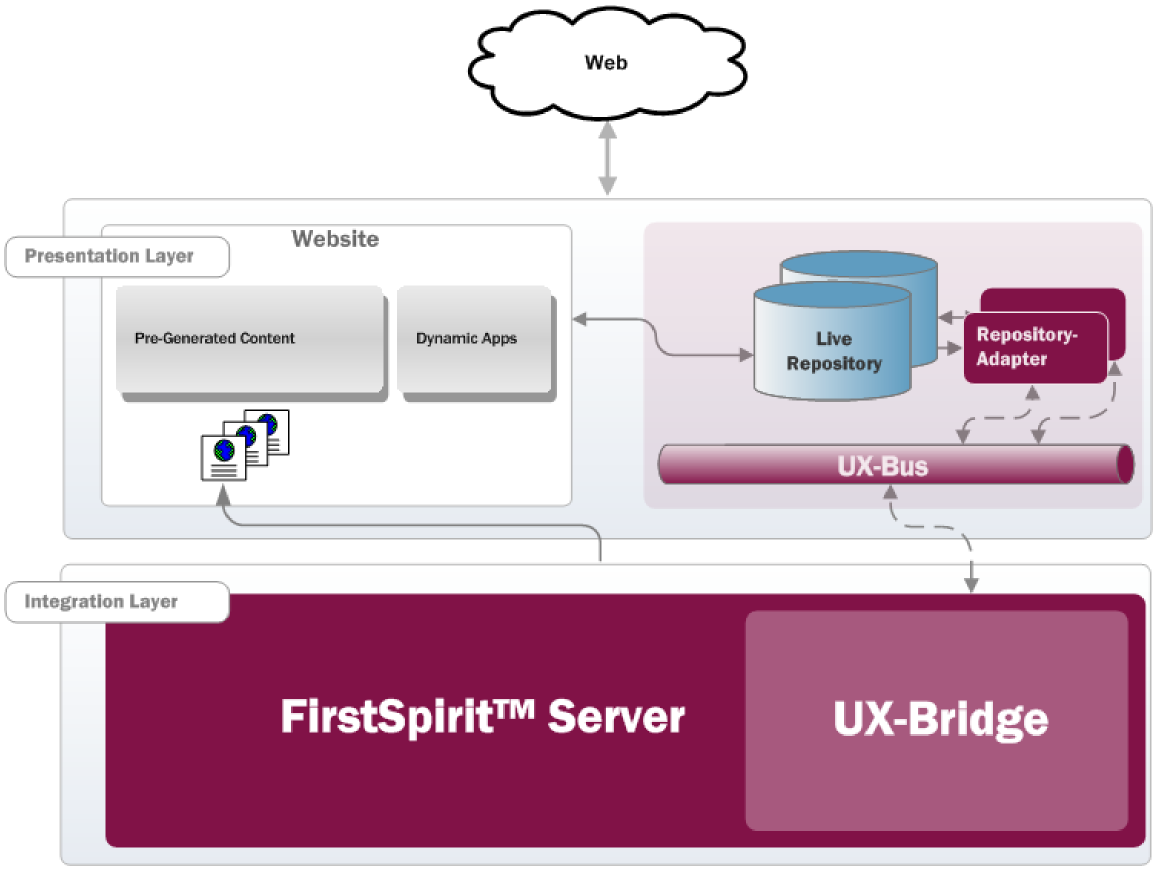


Figure 4: UX-Bridge in detail

- Live repository:
  The live repository is a data storage component which is filled by FirstSpirit and read out by web applications. Here, an important paradigm of the UX-Bridge architecture is: "The type and number of repositories is not prescribed", because different repositories are differently suited based on task type (see Chapter 2.1.3.2).
  In many cases, it can also make sense to fill the live repository not exclusively from FirstSpirit, rather also from the website. This is, for example, relevant when administrating user-generated content (UGC). In the live repository, FirstSpirit Content (for example, press releases) is linked with UGC (for example,

evaluation of the press releases).

- Repository Adapter:
  The repository adapter takes over inserting the FirstSpirit content in the content repository. Simultaneously, the project-specific FirstSpirit data model of a content object (such as an XML construct with the press release) is mapped to the data model in the repository, which is usually determined by the web application (for example, a relational or NoSQL database). Then the repository adapter always has a project-specific part, i.e., a suitable mapping logic has to be implemented for every kind of data.

2.1.3.1 Choosing the live repository

Choosing the "correct" live repositories is an important architecture decision within the project. There are always two important questions to answer in the planning phase of a project:

a) What editorial data should/has to be transferred into a live repository?

b) What kind of repository is optimal for this kind of data and (web) application?

In general, UX-Bridge does not make any firm specifications here, since various criteria have to be considered for the purpose of a delivery infrastructure suitable for the task:

- The type of web application and data:
  What specific type of web application should be used in regards to the data persistence? What does the data model of the application look like, and in what type of persistence is this data model best mapped? Is a static data model suitable, or is dynamic extensibility necessary? For example, is transactionality important, or is the focus on availability[1]?

- Performance and scaling requirements:
  What are the requirements like in the direction of performance and scalability of the repository? Is horizontal scalability important? How relevant is smoothing load peaks? To what extent does the performance have to be elastic and adjustable?

[1] The described trade-off in the repository properties is also called the "CAP theorem" (http://en.wikipedia.org/wiki/CAP_theorem)

- Third-party systems:
  What additional systems must have access to the data? What type of persistence do the third-party systems work with best?

- Tool and framework support:
  What kind of repository does the web framework you are using work with particularly well? For what do plugins already exist? Are there additional tools (for example, for import/export/reporting) that have to access the repository?

- Expertise in the team:
  What type of persistence are the developers most familiar with? With which system is the best development performance to be expected?

These are all questions that have to be asked before selecting a repository. They should be answered in order to make a decision on this basis.

For an initial overview, we list some common kinds of repositories here:

- Relational databases (Oracle, MySQL, PostgreSQL, etc.)

- NoSQL databases (MongoDB, Cassandra, Redis, SimpleDB, DynamoDB, etc.)

- Index based storage (Lucene, Solr, Elasticsearch, Sphinxsearch, etc.)

2.1.3.2 Polyglot persistence

With the described selection of a suitable repository for specific use cases, it is entirely possible that even within a project, multiple, different repositories are used in order to implement a solution suitable for the task. This architectural approach is also called the "polyglot persistence" strategy. This term stands for a simple approach:

"Select the suitable persistence for the respective use case"

Simultaneous use of different kinds of persistence is expressly permitted here for data, as long as it makes sense technically. For more details on the topic of polyglot persistence, refer to: http://martinfowler.com/articles/nosql-intro.pdf

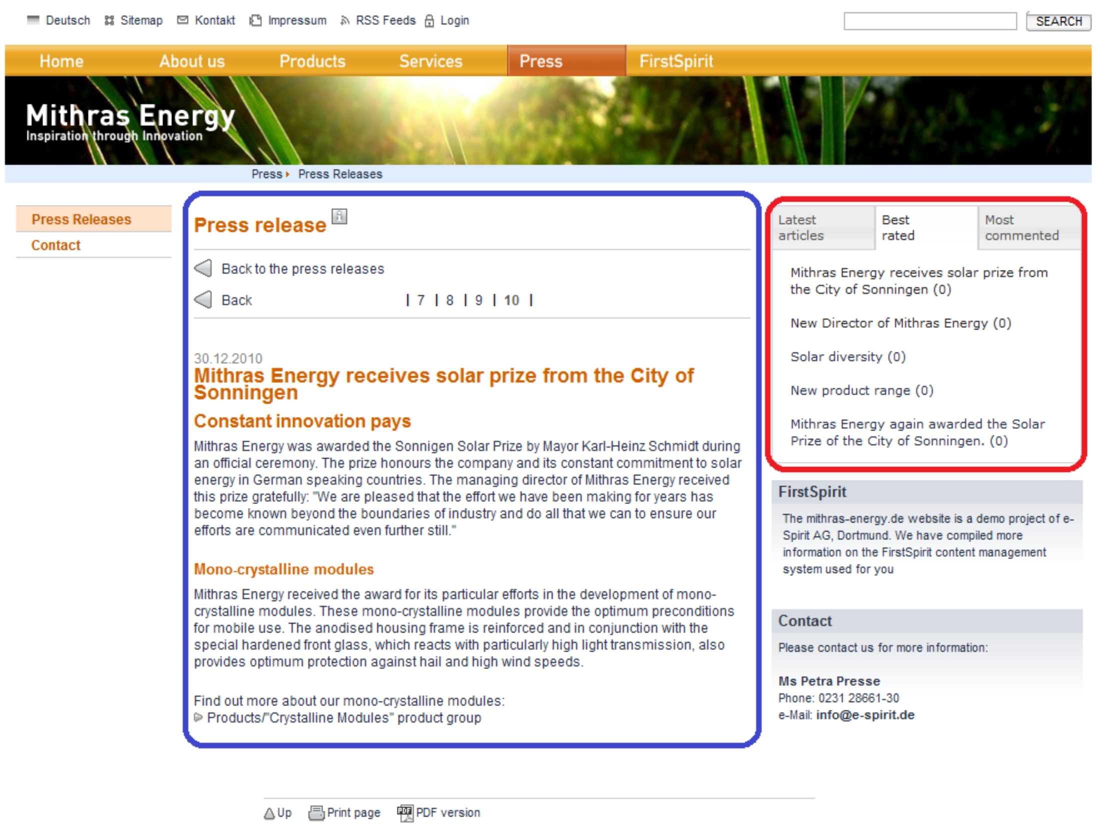
The UX-Bridge follows precisely this approach, since the technical requirements determine the selection of the specific types of persistence (also refer to Chapter 1.5).

2.2 Examples for operational scenarios

To answer the question of which scenarios it makes sense to use the UX-Bridge in, following are some practical examples. To support viable architectural decisions, close attention is paid in this relation to which aspects are dynamized and which are logically pre-generated. Selection of a suitable repository is also discussed for each use case.

2.2.1 News scenario including User-generated Content (UGC)

Let us reconsider the news scenario from Chapter 2.1. A website has numerous messages that are administered via the FirstSpirit data sources. The website has news overview pages (with the latest 10 news items on the first page) and news detail pages (see Figure 5: News scenario). The detail pages have not only the press release (blue border), but also a top news widget (red border), which dynamically depicts the top news based on certain criteria (highest-rated, etc.).



Press release

Back to the press releases

Back | 7 | 8 | 9 | 10 |

30.12.2010

Mithras Energy receives solar prize from the City of Sonningen

Constant innovation pays

Mithras Energy was awarded the Sonnigen Solar Prize by Mayor Karl-Heinz Schmidt during an official ceremony. The prize honours the company and its constant commitment to solar energy in German speaking countries. The managing director of Mithras Energy received this prize gratefully: "We are pleased that the effort we have been making for years has become known beyond the boundaries of industry and do all that we can to ensure our efforts are communicated even further still."

Mono-crystalline modules

Mithras Energy received the award for its particular efforts in the development of mono-crystalline modules. These mono-crystalline modules provide the optimum preconditions for mobile use. The anodised housing frame is reinforced and in conjunction with the special hardened front glass, which reacts with particularly high light transmission, also provides optimum protection against hail and high wind speeds.

Find out more about our mono-crystalline modules:
Products/"Crystalline Modules" product group

Latest articles | Best rated | Most commented

Mithras Energy receives solar prize from the City of Sonningen (0)

New Director of Mithras Energy (0)

Solar diversity (0)

New product range (0)

Mithras Energy again awarded the Solar Prize of the City of Sonningen. (0)

FirstSpirit

The mithras-energy.de website is a demo project of e-Spirit AG, Dortmund. We have compiled more information on the FirstSpirit content management system used for you

Contact

Please contact us for more information:

Ms Petra Presse
Phone: 0231 28661-30
e-Mail: **info@e-spirit.de**

Up | Print page | PDF version

Figure 5: News scenario

The website requirements are:

a) A new or changed news item has to appear on the website within 1 minute

b) Very high delivery speed of the news overview page and news detail page

c) The widget update (highest-rated news) has to take place immediately if new UGC is added

Requirements a) and b) can be met by generating static HTML pages from FirstSpirit via the standard mechanisms. The UX-Bridge is very well suited for requirement c) (see Figure 6: Hybrid news scenario)

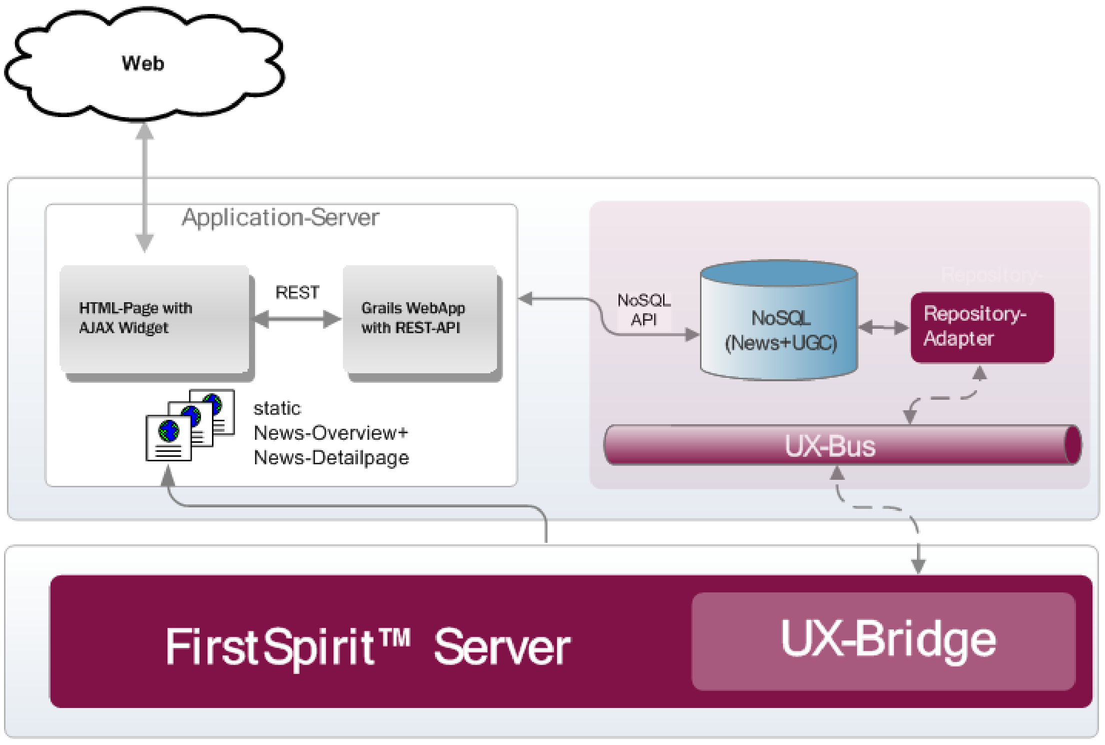


Figure 6: Hybrid news scenario

In this case, a NoSQL database (for example, MongoDB) is selected, in which the relevant (that is, dynamically displayed in the teaser) part of the news is saved, as well as the UGC for comments and ratings. Here, the NoSQL-DB is preferred to a classic relational database, since the UGC data has a very simple structure and the database offers particularly easy access via REST-API. Without the UGC part, a relational database would also be a good option.

A stand-alone web application is developed here with an MVC web framework (here, Grails) in order to read the news and manage (read+write) the UGC. The choice in this example was Grails (see http://grails.org), since it supplies finished plugins for access to NoSQL DBs and very efficient application development is possible for the described use case. Additionally, with Grails it is easy to make JSONP available via a REST interface.

The dynamic news teaser box is directly linked into the static news pages via JavaScript. Each time a page is called up with AJAX, the box calls up the Grails web app and incorporates the results into the website.

2.2.2 Real-time update of the website

State-of-the-art websites frequently have to be equipped with a real-time update. This means the contents are updated without the website user having to call up the page again (manually).

Technically, the same architectural approach can be used as in the previous chapter. The only addition needed is a polling mechanism in the news widget. In this case, the widget regularly (every x seconds) asks the news web application via AJAX whether new data for the current view is present. If it is, the widget downloads the new data via AJAX and updates itself. Alternatively, a pull mechanism can be used, for example, via web sockets.

2.2.3 Product configurator

In the "product configurator" example, a complex web application has to be developed that enables a website user to personally configure a product consisting of various versions and features (such as a car). The website essentially consists only of this application. The complete "frame", that is, the layout and the content from the header and footer fields, should come from CMS. It should also be possible to fully maintain all product information (versions, features, etc.) by editing it with FirstSpirit.

Thus a central element of the website is the configurator web application, which was initially developed independently of FirstSpirit. Since the web application is based on a more complex data model, a relational database is selected as the repository in this case. Also relevant for this selection is the fact that state-of-the-art web frameworks automatically generate a relational model suitable for the object model of the web application, which enables very efficient development.

However, coordination with the FirstSpirit developer is necessary when defining the data model in the live repository, so that the structure of the FirstSpirit data sources can also be mapped to the desired repository model. The mapping, in turn, is implemented in the repository adapter, which either a web developer or a FirstSpirit developer can take over.

This yields the following architecture (see Figure 7: Product configurator).

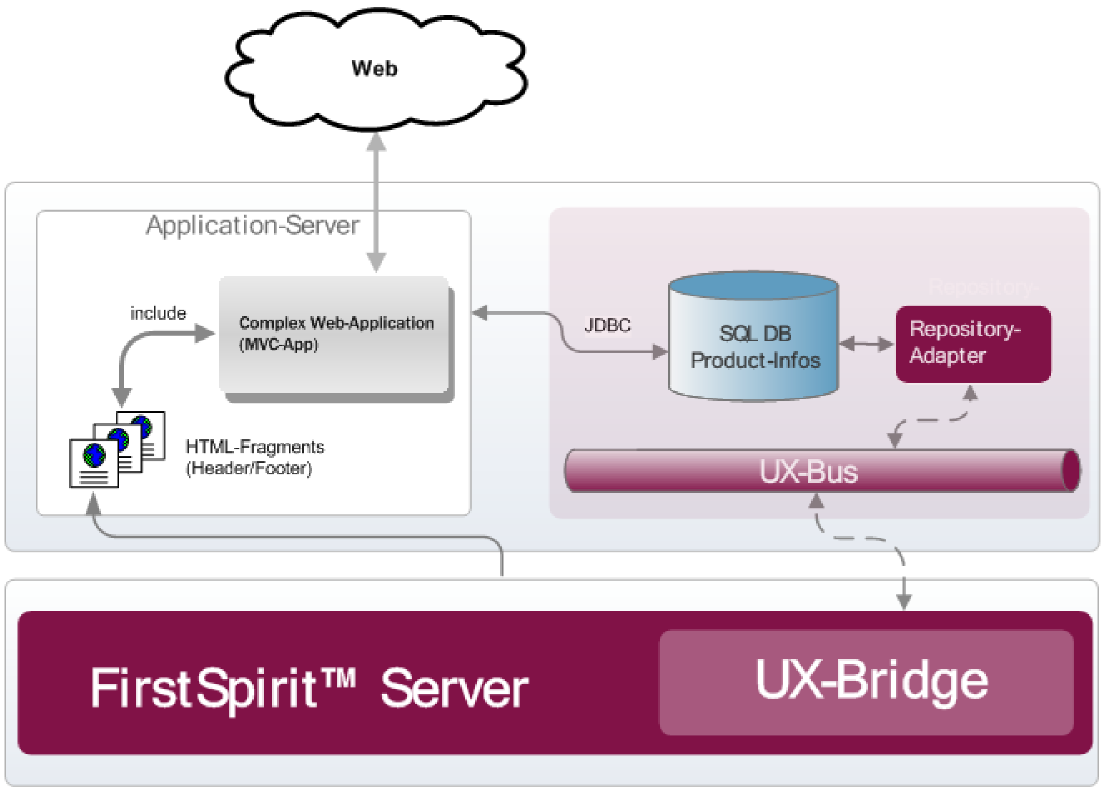


Figure 7: Product configurator

FirstSpirit generates the suitable HTML fragments for the "frame" of the web application (header/footer) either once or cyclically. The web app includes the generated fragments whenever called up; otherwise it works only on the live repository (here, a relational SQL database). Each time product data is changed in FirstSpirit, the UX-Bridge generates the appropriate changes in the product database.

2.2.4 Branch search

A branch search is a popular use case for an integrated web application that is to be linked directly into a website maintained by editors. The input form is rather simple (for example, zip code or address), as is the results list (list of the nearest branches).

To be able to embed the web application at any location in the website (for example, in a sidebar on the homepage), a useful development tool is a JSP-Taglib for visualizing results and a suitable Java servlet, which maps the actual search logic (see Figure 8: Embedded web app).

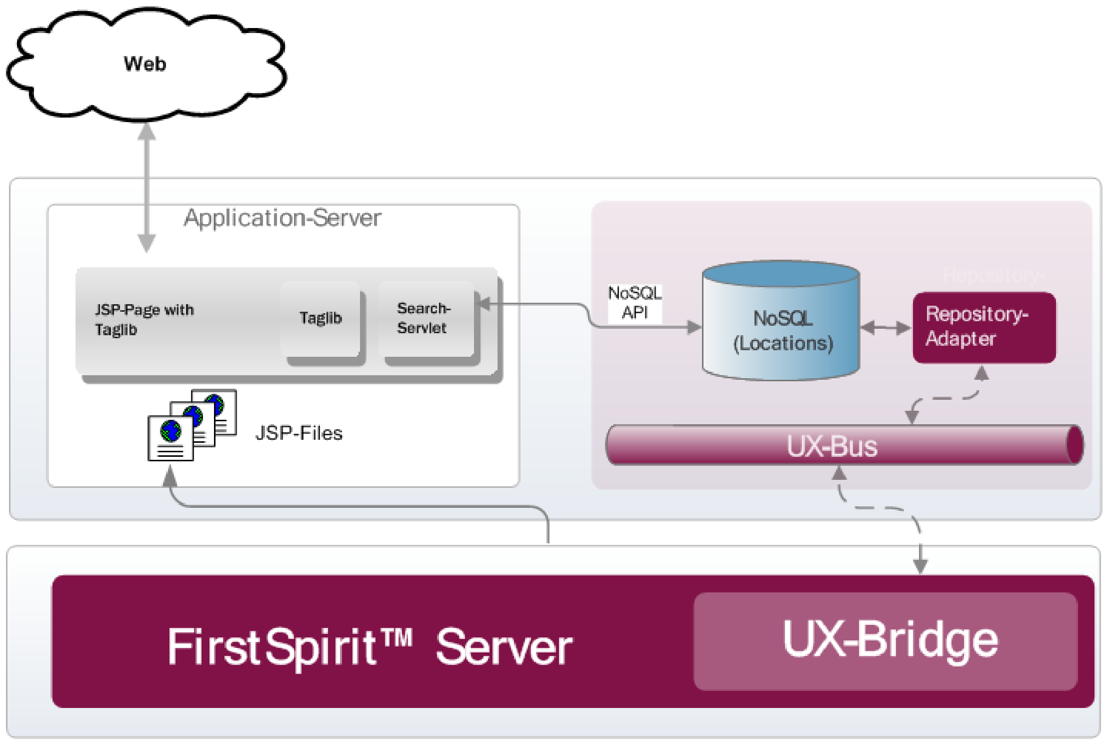


Figure 8: Embedded web app

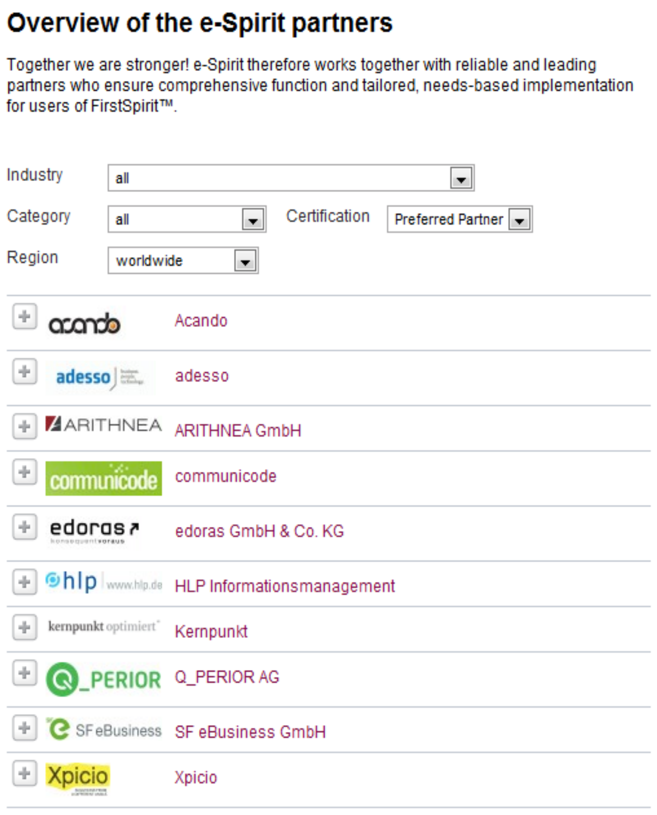
In this case, FirstSpirit generates JSP files that contain editorial content, the search form, and the presentation of results as JSP logic.

A NoSQL database (MongoDB) is selected as the repository here, since this has ready-made mechanisms for searching the surrounding area (geospatial). The search servlet addresses this geographic service directly via the MongeDB-API, while the necessary geographic data is populated by editors directly via FirstSpirit (for example, by integration of Google Maps).

2.2.5 Structured search for a contact person

The application logic within web applications is very frequently associated with the "Search" topic. In addition to a normal full-text search, there is often the desire for a "structured search", that is, the objects to be searched for have a quantity of attributes that will be used in filtering, possibly even hierarchically for a drill down.

A specific use case is, for example, the search for a contact person on the e-Spirit website (see Figure 9: Search for contact persons). Every contact person has attributes there, such as region, branch, or contact level.



Overview of the e-Spirit partners

Together we are stronger! e-Spirit therefore works together with reliable and leading partners who ensure comprehensive function and tailored, needs-based implementation for users of FirstSpirit™.

Industry	all		
Category	all	Certification	Preferred Partner
Region	worldwide		

Acando
adesso
ARITHNEA GmbH
communicode
edoras GmbH & Co. KG
HLP Informationsmanagement
Kernpunkt
Q_PERIOR AG
SF eBusiness GmbH
Xpicio

Figure 9: Search for contact persons

With this kind of application, it is advisable to use an index as live repository (such as Lucene or Solr), particularly if there are very large amounts of data that have to be searched through extremely quickly (see Figure 10: Index-based repository). If there is new or changed content, FirstSpirit updates the index of the Solr server directly and thereby always keeps this consistent with the content on the website.

In this example, the other architectural components are identical with the branch search from Chapter 2.2.4. The web app "talks" to the live repository via the Solr-API to answer the search requests.

At this point it is already fairly clear that it makes sense to use UX-Bridge as a technical basis for "higher-quality" modules, such as Enterprise Search. In future versions of the FirstSpirit run time modules, this path will be pursued more.

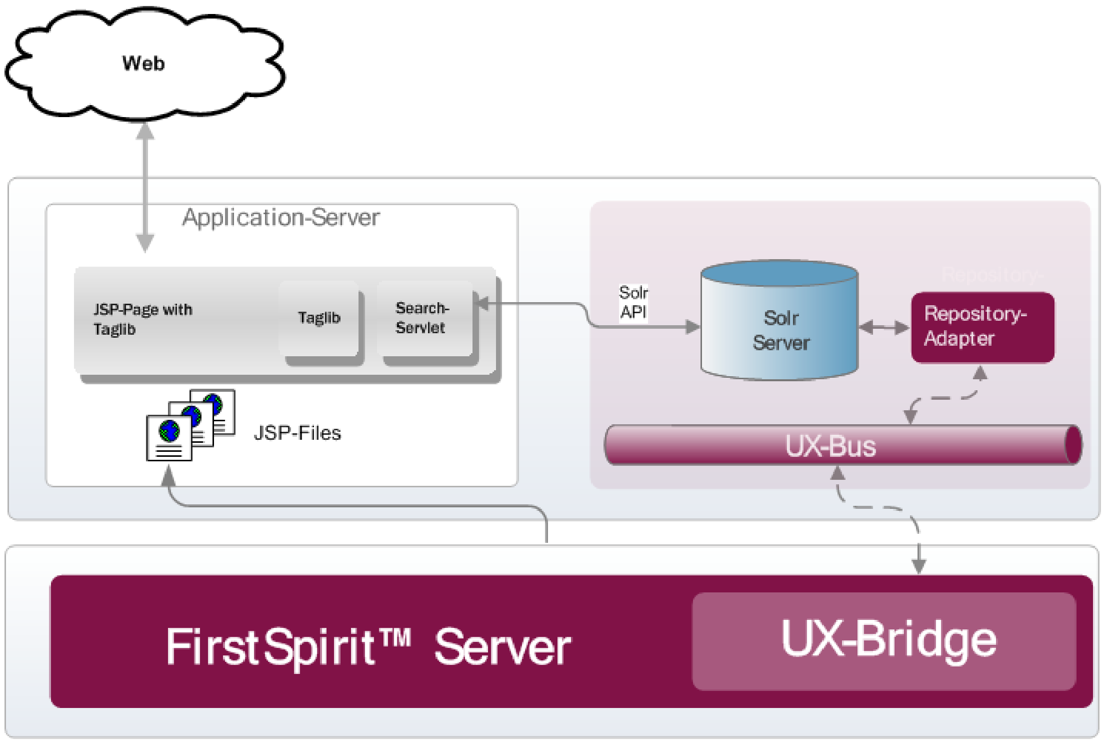


Figure 10: Index-based repository

2.2.6 Mobile news portal

Making content available on mobile devices is a use case that is encountered with increasing frequency. In this example, a mobile news portal has to be drafted that transfers the news maintained by editors to a smartphone app. Important functional requirements here are:

- Offline capability of the app:
  The network connection is repeatedly lost on mobile devices. The user of the app should not notice these temporary losses, that is, the app should be "offline-capable." As soon as a network is present, the local data storage device of the app should automatically synchronize with the central repository.

- Providing the content as "HTML fragment":
  The app developer may want to receive the news in the form of pre-rendered HTML fragments. This permits very high flexibility in case the layout of the content is changed frequently. In this case, the app does not have to be adapted; instead, only the content fragments have to be re-generated.

An intelligent solution for implementing an offline-capable content app can be seen in Figure 11. The NoSQL database CouchDB is used here as content repository (see http://www.couchdb.org/). The great advantage of this architecture is that the content synchronization (which is a complex topic) occurs completely through the CouchDB. For this purpose, a local/mobile version of the CouchDB is used on the smartphone that has appropriate synchronization capabilities. Thus there is no need here for complex programming of the synchronization logic.

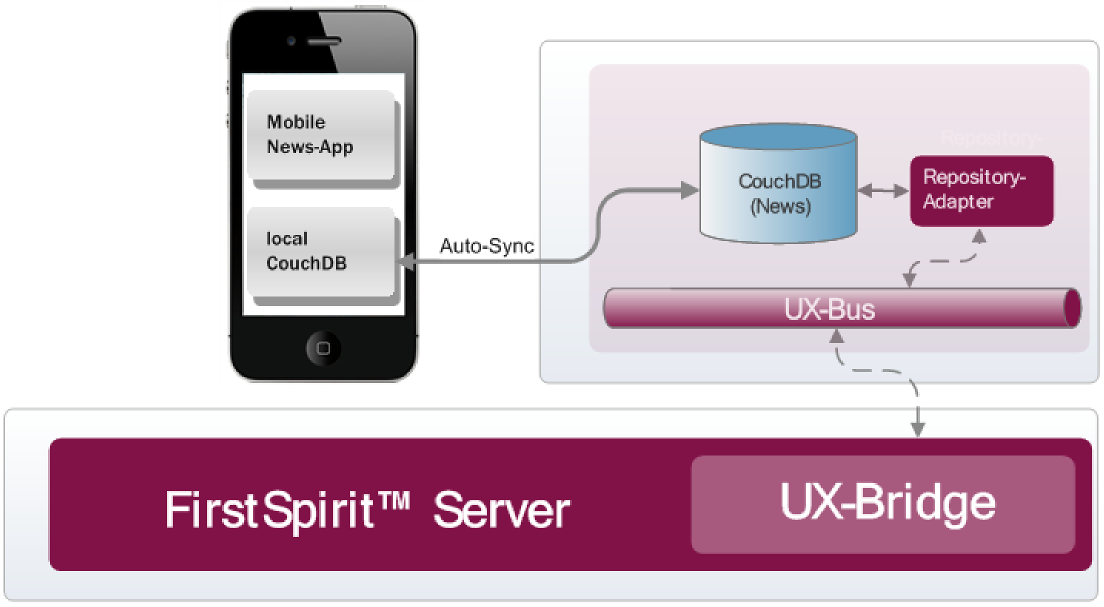


Figure 11: Mobilization via UX-Bridge

This architecture is a good example of how skillful selection of the content repository frequently leads to a reduction of implementation effort.

Finally, when considering these examples of applications, you can see that different applications also mean different architecture variants.

In each case, the UX-Bridge provides the necessary flexibility to be able to implement a suitable solution in very different scenarios.

In the next chapter, possible architecture variants will be classified again in a different way. In doing so, new variants that were not in the examples described will also be mentioned.

2.3 Architecture variants

There are various existing architecture variants involving web applications on which the web server accesses UX-Bridge. The usual variants are summarized again briefly and categorized in this chapter.

2.3.1 Client-side dynamics via widgets

A very lightweight variant to map fairly straightforward web applications is the implementation of a JavaScript logic, which runs entirely within an HTML page in the browser (see application example in Chapter 2.2.1).

Logical operational scenarios are:

- Lightweight widgets
- Manageable complexity / logic
- Simple integration into any location on the website

2.3.2 Server-side dynamics

Server-side dynamics make sense whenever more complex application logic is necessary or when security plays a particularly important role.

There are various sub-variants of this application category, in which additional variants are certainly conceivable, alongside the listed types.

2.3.2.1 JavaServer Pages (JSP)

The seamless integration of logic from the server is done classically via Java Server-Pages (JSP) in a Java context. In the most simple variant (JSP Model 1), the JSP code (or the JSP Taglibs) is integrated directly into the page.

Logical operational scenarios are:

- Simple server-side logic
- Simple integration into any location on the website

2.3.2.2 MVC frameworks

The use of Model-View-Controller (MVC) frameworks lends itself to more complex web applications with complicated server-side logic. This approach makes it possible to develop applications that can be maintained and expanded readily.

Logical operational scenarios are:

- Complex web applications
- Stand-alone development of the web application, completely independent of FirstSpirit as needed

2.3.2.3 Portlets

The portal technology provides a standardized infrastructure in order to integrate individual web applications into a website in the form of portlets, independent of other applications. In addition, any editorial content can be expanded, which likewise is integrated via (content) portlets.

This approach is particularly interesting, as FirstSpirit already provides finished integrations into known portal servers (SAP Netweaver, IBM Websphere Portal, Microsoft Sharepoint or Liferay).

The use of a portal server was a basic architecture decision, which must be made from various viewpoints. If you have decided in favor of a concrete portal server, however, all advantages of a standardized integration platform can be used.

2.3.2.4 Connection of third-party systems

Alongside development of your own applications, the more frequent use case is that existing third-party applications have to be provided with editorial contents, which are to be configured and controlled editorially from FirstSpirit.

Concrete examples for these types of applications are:

- e Commerce applications / Shop systems
- Recommendation engines
- Community software
- Collaboration platforms

For each of these types of applications, it is to be decided on the basis of the technical requirements whether a pre-generation of appropriate HTML/XML fragments is sufficient, or whether UX-Bridge should be used alternatively or supplementally.

3 FAQ

When using UX-Bridge in the concrete project, there are frequently recurring questions, which are answered in the following.

3.1 Questions on dynamization depth

[Q] Can I also move the complete website, with all contents and structures, into the live repository via UX-Bridge?
[A] Theoretically, yes, but this is usually not a good architecture. Only in the very fewest of cases is the entire website fully dynamic. In all other cases, you should use the pre-generation.

3.2 Questions on UX-Bus

[Q] Can you also operate UX-Bus on the FirstSpirit server, in order to have to install as little infrastructure on the live page as possible?
[A] Operating UX-Bus on the FirstSpirit server is technically possible. In doing so, ActiveMQ is operated on the FirstSpirit server in integrated Jetty. To the extent that UX-Bus is only used to fill the live repositories, this scenario should be sufficient for most use cases. If UX-Bus has to be highly available, then UX-Bus is to be decoupled from FirstSpirit. Additional notes on the architecture can be found in our UX-Bridge admin handbook.

[Q] Is ActiveMQ the only supported messaging infrastructure?
[A] Any messaging systems can be used, but only for ActiveMQ e-Spirit can also offer operating experience.

[Q] Whom can I contact if there are problems with UX-Bridge?
[A] In the event of problems with UX-Bridge or other modules, please contact our help desk. Consultation with regard to operational scenarios and architecture is available via our Professional Service.

[Q] How is a sizing for ActiveMQ done?
[A] Sizing for ActiveMQ is largely dependent on the required availability, which can be achieved through suitable clustering and failover mechanisms. An additional factor is presented by the number of expected messages per second. For more reliable evidence, a performance test should be carried out in the project. More precise notes can be found under http://activemq.apache.org/performance.html.

Configuration recommendations can be found in our UX-Bridge admin handbook.

3.3 Questions on selecting a repository

[Q] What is the "Standard" live repository that is recommended by e-Spirit?
[A] e-Spirit does not recommend any standard repository. Similarly to the selection of a concrete database or an operating system, the "correct" repository depends very heavily on the technical requirements.

[Q] I have no particular requirements for a live repository. Which live repository should I choose then?
[A] If technically there are only absolutely minimal requirements (very simple data model, little traffic on the website, few updates, etc.), a repository should be selected in which the implementation partner has as much expertise as possible (in the operation and use area). Generally, the expertise of the implementer should always be taken into account during selection.

3.4 Questions on infrastructure

[Q] How is a 2 or 3-level system landscape (D/Q/P) set up with UX-Bridge?
[A] As with other modules, we recommend installing the necessary components on every system (development, test systems, production) in order to guarantee a strict separation of the individual systems and to be able to test the correct configuration. It is however also possible, through the corresponding routing, to distribute the messages across all systems via a UX-Bus.

[Q] Can UX-Bridge be used in a high availability scenario?
[A] The core of UX-Bridge presents the messaging infrastructure, based on ActiveMQ. ActiveMQ was developed especially for high availability scenarios. More precise information on error tolerances and clustering can be found in our admin handbook.

[Q] What does a cluster operation of UX-Bridge look like?
[A] For cluster operation, a number of topologies are possible. Details can be found in our user admin handbook.

[Q] Where can I get commercial support for ActiveMQ?
[A] There are a number of providers who offer commercial support for ActiveMQ. A list can be found under http://activemq.apache.org/support.html

3.5 Other questions

[Q] How can UX-Bridge be integrated into the FirstSpirit preview?
[A] For integration into the FirstSpirit preview, only the web application is important. How this can be integrated into the preview depends on the technology used. If you have concrete project requirements, please contact the e-Spirit Professional Services department.

[Q] How do you handle media (for example, images in a product catalog)? Should these also be inserted into the live repository?
[A] Generally, it is technically possible to transfer binary files into the repository during this process. In most use cases, however, a pre-generated deployment is logical instead. If, for example, a product detail page is pre-generated, then the media contained is already a part of the normal deployment. If certain resolutions are needed (for example, for different product teasers), then a separate media deployment of all associated product images is preferable.

[Q] Will UX-Bridge provide me with help related to the delta deployment?
[A] Delta deployment refers to a mechanism that only generates and deploys the pages/objects that have changed since the last deployment. UX-Bridge provides no new mechanism for this, but uses the mechanisms provided by FirstSpirit (work flows, revision API, partial generation and, as of FirstSpirit 5, DeltaGeneration as well).

[Q] When using UX-Bridge, do I also need a full-text search?
[A] Here, we differentiate between two scenarios: As long as the content is not generated only via UX-Bridge, but also as a static file, then no special handling is necessary. The content can then be searched completely normally via the BasicSearch or EnterpriseSearch. Refer also to Chapter 2.2.1, News Scenario. If the content is only persistent in the live repository, an additional configuration or implementation is necessary. The EnterpriseSearch module offers a number of connectors to connect external repositories. Alternatively, the Polyglot Persistence approach can be used to write the content via UX-Bridge to a search index as well (see Chapter 2.2.5).